www.css-voorbeelden.nl

Volledig met css aan te passen videospeler(s)

BELANGRIJKE INFORMATIE
Alles op deze site kan vrij worden gebruikt, met twee beperkingen:
* Je gebruikt het materiaal op deze site volledig op eigen risico. Het kan prima zijn dat er fouten in de hier verstrekte info zitten. Voor eventuele schade die door gebruik van materiaal van deze site ontstaat, in welke vorm dan ook, zijn www.css-voorbeelden.nl en medewerkers daarvan op geen enkele manier verantwoordelijk.
Bij dit voorbeeld is dit helemaal belangrijk. Als er ooit iets verandert in browsers en je hebt dit script gebruikt, is het maar helemaal de vraag of het script (snel) wordt aangepast. Deze hele site wordt alleen door mij gerund, voor de lol, en als ik er geen zin meer in heb, of als ik iets niet kan repareren, werkt dit script mogelijk niet meer. Denk daar dus even heel goed over na, voordat je dit script eventueel 'in het echt' gaat gebruiken. Wat mij betreft is het niet meer dan 'n demonstratie van wat er mogelijk is.
* Deze uitleg wordt regelmatig bijgewerkt. Het is daarom niet toegestaan deze uitleg op welke manier dan ook te verspreiden, zonder daarbij duidelijk te vermelden dat de uitleg afkomstig is van www.css-voorbeelden.nl en dat daar altijd de nieuwste versie is te vinden. Dit is om te voorkomen dat er verouderde versies worden verspreid.
Een link naar www.css-voorbeelden.nl wordt trouwens altijd op prijs gesteld.
Meestal staat onderaan dit document de volledige code. Dat is hier maar gedeeltelijk het geval: alleen de html van de eerste pagina met videospelers staat daar. Het is niet goed mogelijk alle code daar neer te zetten, en bovendien zou dat heel veel dubbele code opleveren.
De code die er wel staat, is exact hetzelfde als die van in de download bijgesloten bestanden. Het is de bedoeling dat je die bestanden gebruikt, als je de code wilt bewerken. Kopiëren van de code achteraan dit bestand om die te bewerken – als dat al lukt – levert de wildste problemen op.
De code die te maken heeft met de basis van dit voorbeeld, is in de hele uitleg rood gekleurd. Alle niet-essentiële code staat in een afwijkende `zwarte` lettersoort. (In de inhoudsopgave staat alles in een gewone zwarte letter vanwege de leesbaarheid.)

Inhoudsopgave

Uitleg code:

- Speciaal voor vierde video
- Speciaal voor vijfde video
- Speciaal voor zesde video

Pagina 3 (Alleen verschillen met de css voor pagina 1 hierboven):

- Algemeen
  - Verborgen bedieningselementen
  - Symbolen voor bedieningselementen
  - css voor vensters breder dan 700 px
  - css voor vensters breder dan 1200 px
- Bedieningselementen algemeen
- Speel-pauzeerknop
- Knoppen en percentage volume
- Sleepbalk volume
- Knoppen afspelen
- Sleepbalk afspelen
- Verstreken, resterende en totale speelduur
- Focus
- Speciaal voor tweede video
- Speciaal voor derde video
- Speciaal voor vierde video
- Speciaal voor vijfde video
- Speciaal voor zesde video

Pagina 4 (Alleen verschillen met de css voor pagina 1 hierboven):

- Algemeen
  - Verborgen bedieningselementen
  - Symbolen voor bedieningselementen
  - css voor vensters breder dan 700 px
  - css voor vensters breder dan 1200 px
- Bedieningselementen algemeen
- Speel-pauzeerknop
- Aan-uitknop geluid
- Knoppen voor afspelen, Harder en Zachter
- Percentage volume, verstreken, resterende en totale speelduur
- Focus
- Speciaal voor tweede video
- Speciaal voor derde video
- Speciaal voor vierde video
- Speciaal voor vijfde video

Pagina 5 (Alleen verschillen met de css voor pagina 1 hierboven):

- Algemeen
  - css voor vensters breder dan 700 px
  - css voor vensters breder dan 1200 px
- Speciaal voor eerste video
- Speciaal voor tweede video
- Speciaal voor derde video
- Speciaal voor vierde video
- Speciaal voor vijfde video
- Speciaal voor zesde video

Korte omschrijving

Meerdere videospelers op één pagina. Bij alle spelers gezamenlijk of bij elke speler apart kan het uiterlijk volledig worden aangepast, kunnen bedieningselementen worden verborgen, enz., met behulp van alleen css.

Opmerkingen

Als je iets gaat veranderen in het script (hoe dat moet, staat bij Wijzigingen aanbrengen in het script), **MAAK DAN ALTIJD EERST EEN BACK-UP!!!!!!**
Feitelijk moet je zelfs een hele serie back-ups hebben, want soms wordt het pas na ’n tijd duidelijk dat er een fout in is geslopen. Een kleine typefout is snel gemaakt en kan het hele script compleet onbruikbaar maken.

In deze uitleg zijn de namen van id’s, classes, enz. gebruikt, zoals die zijn opgegeven in het bijgesloten script. De waarin de video staat bijvoorbeeld heeft class=”videobox”. Als je die naam in het script hebt veranderd in bijvoorbeeld ’paashaas’, moet je uiteraard in deze uitleg voor ’videobox’ ’paashaas’ lezen. Dit geldt voor alle id’s, classes, enz.
JavaScript is Engelstalig. Zonder (enige) beheersing van de Engelse taal wordt het wel heel erg moeilijk iets met JavaScript te doen. Omdat Engels feitelijk een vereiste is, heb ik om die reden ook de namen van classes, id’s, in het script ingebouwde foutmeldingen, e.d. Engelstalig gehouden. Anders wordt het een heel verwarrende mengeling van Nederlands en Engels. Bovendien is de code nu ook (enigszins) te lezen door mensen die geen Nederlands kennen.
Het gaat hier trouwens alleen om de code zelf, zeg maar wat de programmeur krijgt te zien. In de videospelers zelf, wat uiteindelijk op het scherm verschijnt, is alles gewoon Nederlandstalig. Ook het commentaar in ’afbeelding-103-met-commentaar.js’ is gewoon Nederlandstalig.
Vrijwel alle namen van id’, classes, teksten, enz. zijn heel simpel te wijzigen. Dat moet in het script zelf gebeuren, maar is echt dodelijk simpel. Hoe je namen e.d. in het script kunt veranderen, staat bij Wijzigingen aanbrengen in het script.

Het is uiteraard ook mij volstrekt duidelijk dat niet elke videospeler even mooi is. Om niet te zeggen dat sommige ronduit afschuwelijk zijn, of niet te bedienen op touchscreens vanwege de te kleine knoppen, of alleen geschikt zijn om mensen acuut een zenuwinzinking te bezorgen. Maar het gaat er alleen om de mogelijkheden te laten zien. De zesde videospeler op pagina vijf bijvoorbeeld zal niemand die niet acuut al z'n bezoekers wil wegjagen ooit echt gebruiken. (Nee toch?)

Ik ben beslist geen JavaScript-expert, en er zijn ongetwijfeld veel verbeteringen in het script mogelijk. Maar alles werkt (min of meer) in alle geteste browsers, en het ging me er vooral om de mogelijkheden van de combinatie van een van de nieuwe html5-API's en css te laten zien.

Als dat is opgegeven, wordt beneden een bepaalde breedte of hoogte (in het voorbeeld is dat 480 px) van het venster van de browser de ingebouwde standaardvideospeler gebruikt, omdat deze op bijvoorbeeld een smartphone veel beter werkt dan een uitgebreide eigen speler. Als bij resizen of kantelen van het venster de maat van 480 px wordt gepasseerd, verandert de speler niet. Om over te schakelen van de standaardspeler naar de eigen, of omgekeerd, moet de pagina worden herladen.
Hierdoor wordt voorkomen dat bij het kantelen van bijvoorbeeld een grote smartphone opeens een totaal andere bediening verschijnt. Bovendien kan eventueel een kleiner formaat video worden opgegeven voor een kleiner venster, en het is ook niet prettig als tijdens het kijken een nieuwe video moet worden gedownload.

Er kan maar één video gelijktijdig spelen. Sommige browsers hebben dat al ingebouwd, andere niet. Zodra je 'n video start, regelt het script dat alle andere video's worden gepauzeerd.

De eerste seconden van de zesde video hebben geen geluid. Omdat de grote versie van deze video is ingekort tot drie seconden, heeft deze helemaal geen geluid. De kleine versie, die tien seconden duurt, heeft aan het eind wel geluid. (Ik vermeld dit even omdat ik zelf enkele uren bezig ben geweest te onderzoeken waarom die %)(*&)-video geen geluid had, denkend dat het om een of andere obscure fout in het script ging of zo.)
Eveneens ter voorkoming van acute woedekolieken: het kleine kiertje aan de bovenkant van de vijfde video zit in de video zelf. Het heeft dus geen nut om, net als ik heb gedaan, uren te zoeken waar die)*)(*$#-kier vandaan komt: het heeft niets met de css te maken.

Niet elke animatie werkt in elke videospeler. Dit geldt vooral voor de zesde videospeler op pagina vijf. Dat maakt niet uit: als een animatie het niet doet, blijven alle knoppen en zo het gewoon doen, dus de speler blijft gewoon werken. Bovendien gaat geen normaal mens zo'n krankzinnige speler ooit echt gebruiken, mag ik hopen.

Als je in een desktopbrowser met behulp van zoomen het beeld vergroot, heeft dit hetzelfde effect als wanneer de pagina in een kleiner browservenster wordt getoond. Je kunt hiermee dus kleinere apparaten zoals een tablet of een smartphone simuleren. Maar het blijft natuurlijk wel een simulatie, dit is nooit hetzelfde als testen op een écht apparaat. Zo kun je bijvoorbeeld aanrakingen alleen testen op een echt touchscreen.
De pagina moet wel worden herladen als de grens van 480 px wordt gepasseerd, omdat het script zich anders niet aanpast.

De gebruikte video's zijn afkomstig van het Internet Archive. Boven elke pagina staat een link naar de precieze pagina met de volledige, originele video. De bijbehorende licenties staan onder Inhoud van de download en licenties.

De video's zijn geconverteerd op online-convert.com.

Voor het bedienen van de videospelers is gebruik gemaakt van hand.js, een door Microsoft ontwikkelde polyfill voor pointer events. Pointer events is ook door Microsoft bedacht, maar inmiddels door w3c geaccepteerd als standaard. Met behulp van pointer events worden touch events (aanrakingen op touchscreens) op ongeveer dezelfde manier afgehandeld als mouse events (muisklikken e.d.). Een polyfill is een stukje JavaScript dat zorgt dat zoiets ook werkt in browsers die dit (nog) niet hebben ingebouwd.
Om te leren werken met alle verschillende systemen touch events moet je ongeveer 361 jaar studeren. Hard studeren. Pointer events is een fantastische oplossing voor dit probleem. Helaas is Apple niet van plan dit te gaan gebruiken. Microsoft gedraagt zich tegenwoordig uiterst netjes, als het om standaarden gaat. Maar Safari is inmiddels echt de nieuwe Internet Explorer 6, als het om standaarden gaat.

De in sommige spelers gebruikte afbeeldingen, zoals luidsprekertjes, zijn afkomstig van iconfinder.com.

Het op de tweede pagina gebruikte webfont is afkomstig van icomoon.io. Het is samengesteld met behulp van de app op genoemde site.

Het vraagteken waaronder de hulp zit is afkomstig van clker.com. Net als de in sommige spelers gebruikte schildpad, hamster en paard.

Het in sommige spelers gebruikte jachtluipaard is afkomstig van i2Cclipart.

De in sommige spelers gebruikte data-uri's zijn gemaakt op dopiaza.org.

Links in deze uitleg, vooral links naar andere sites, kunnen verouderd zijn. Op www.css-voorbeelden.nl/links vind je steeds de meest recente links.

Alles op deze site is gemaakt op een systeem met Linux (Kubuntu). Daarbij is vooral gebruik gemaakt van Komodo Edit, GIMP en Firefox met extensies. De pdf-bestanden zijn gemaakt met LibreOffice.

Vragen of opmerkingen? Fout gevonden? Ga naar het forum.

Iets gevonden waar je wat aan hebt? Mooi. Als je je waardering wilt uiten, maak dan een donatie over aan War Child Nederland, een organisatie die kinderen uit oorlogsgebieden helpt hun trauma's te verwerken. Of - nog beter - wordt donateur:

Achterliggend idee

Deze hele constructie is ontstaan naar aanleiding van een simpele vraag. Iemand wilde video's afspelen op thumbnail-formaat, waarbij het nogal storend zou zijn als de knoppen van de videospeler boven de video zouden staan, omdat deze dan ongeveer de helft van de video zouden bedekken. Bovendien moest het uiterlijk van de knoppen kunnen worden aangepast.
Voor zover ik weet, bestaat er geen enkele videospeler die de knoppen buiten de video plaatst. Bovendien is bij de videospelers die ik heb bekeken, hoe goed die verder ook zijn, het uiterlijk maar heel beperkt aan te passen.

Omdat het ook ’n leuke mogelijkheid was de nieuwe media API (Application Programming Interface) van html5 uitgebreid te bekijken, is uiteindelijk dit resultaat ontstaan. Inderdaad ietwat uit de hand gelopen, want de oorspronkelijke vraag was tamelijk simpel.
Uitgangspunten waren:
* De bedieningselementen van de videospeler moeten overal neergezet kunnen worden: boven de videospeler (zoals in de tweede videospeler op pagina vier), maar vooral ook buiten de video.
* Het uiterlijk van de bedieningselementen moet kunnen worden aangepast met behulp van css, zonder te sleutelen aan het script, zodat ook mensen die geen JavaScript kennen het uiterlijk kunnen aanpassen.
* Je moet kunnen kiezen welke bedieningselementen je wel of niet wilt gebruiken, zonder dat je aan het script moet gaan lopen sleutelen.
* De videospeler moest volledig zijn te bedienen met behulp van het toetsenbord en toegankelijk zijn voor schermlezers.
* Zonder JavaScript moet het ook werken. (Veel ingebouwde videospelers werken niet, als JavaScript uitstaat, daar is weinig aan te doen, maar ze moeten in ieder geval niet geblokkeerd worden door dit voorbeeld.)
* Het moet in zowel grote als kleine browservensters werken

De bedieningselementen kunnen overal worden neergezet, volledig los van elkaar, in willekeurige volgorde. Elke knop is een zelfstandige <button> of zoiets. De sleepbalken bestaan uit een , waarbinnen weer een voor de balk en een voor de knop zitten.
Je kunt de bedieningselementen desnoods volledig verspreid over de pagina neerzetten, helemaal los van elkaar, en ’n heel eind van de video vandaan. De Speel-pauzeerknop in de linkerbovenhoek, en de Geluid aan-uitknop in de rechteronderhoek. In de voorbeelden is dat nergens gedaan, maar het zou kunnen.
In de tweede videospeler op pagina vier staan de knoppen boven de video, als je ze zichtbaar maakt, dus ook dat kan. (Met enige moeite zou je de bediening zichtbaar kunnen maken bij hoveren over de video, maar dat heb ik verder niet uitgewerkt, omdat je dan voor touchscreens weer iets anders zou moeten maken.)
De knoppen staan grotendeels in twee groepen: eentje voor de geluidsregeling en eentje voor de weergave. Dit maakt het mogelijk WAI-ARIA-roles te gebruiken, zodat schermlezers geluids- en weergaveregeling als groep herkennen. (In de spelers waar de knoppen echt volledig door elkaar staan, kunnen deze roles simpel worden verwijderd, zoals staat beschreven bij Het JavaScript onderaan de html-bestanden.)

Op de afbeelding links staat een video met bijbehorende knoppen, zoals die eruit ziet zonder css. Behalve de sleepbalken zijn alle knoppen aanwezig en werkt alles gewoon. Ik zal niet zeggen dat ze makkelijk werken, want je ziet niet welke knop wat doet, maar het werkt.
Als je door de pagina’s met video’s bladert, zie je dat de videospelers er nogal verschillend uitzien. Dat verschillende uiterlijk wordt volledig met behulp van css bereikt, inclusief de volgorde waarin de knoppen staan. Zonder css ziet elke videospeler er exact zo uit als op de afbeelding hiernaast.

Alle knoppen worden ingevoegd door het script. Om dit te kunnen doen, zoekt dit script allereerst naar elk <video>-element dat binnen een element met class=”videobox” staat. <video>’s die niet binnen een element met class=”videobox” staan, worden genegeerd. Vervolgens wordt bij de gevonden video’s een aantal aanpassingen gedaan, zoals het verbergen van de standaardbediening, de volumesterkte instellen op de opgegeven beginstand, e.d.
Het maakt niet uit hoeveel <video>’s er op een pagina staan, als ze maar binnen een element met class=”videobox” staan.
Binnen elk element met class=”videobox” wordt vervolgens, gelijk na het <video>-element, een aantal knoppen, ’s, enz. ingevoegd. Dat zijn de knoppen zoals die op de afbeelding hierboven zijn te zien. De sleepbalken missen, want die bestaan uit lege ’s, die volledig met css worden vormgegeven.
Voor de css kunnen gewone standaardselectors worden gebruikt, zoals `div p`. Elk element met class=”videobox”, alle daarbinnen liggende <video>-elementen en alle door het script ingevoegde bedieningselementen krijgen een unieke id. Dat is een bepaalde naam, gevolgd door een koppelteken gevolgd door een volgnummer. De eerste videospeler heeft volgnummer 1, de tweede 2, enz. De afspeelknop van de eerste video heeft als id ’play-1’, de afspeelknop van de tweede video ’play-2’, enz.
Dit geeft de mogelijkheid om elk bedieningselement een eigen opmaak te geven.
Daarnaast krijgen alle bedieningselement ook nog een class. Alle afspeelknoppen bijvoorbeeld hebben een class=”play”, zodat je ook alle afspeelknoppen gezamenlijk een bepaalde opmaak kunt geven.
Dit geeft een groot aantal aanknopingspunten voor de css om het uiterlijk aan te kunnen passen.
Tijdens het afspelen e.d. worden ook nog wijzigingen aangebracht, die door de css kunnen worden gebruikt. Als een video bijvoorbeeld niet afspeelt, heeft deze de class ”not-playing”. Zodra de video afspeelt, wordt deze class verwijderd. Op deze manier kun je de Speel-pauzeerknop aanpassen naar gelang de video wel of niet afspeelt.
Daarnaast is er nog een aantal data-attributen die door het script worden bijgehouden, en die ook door css-selectors kunnen worden gebruikt. Hierin wordt bijvoorbeeld de verstreken speelduur bijgehouden. Een lijstje hiervan is te vinden bij Door het script aangestuurde data-attributen (data-=”...”).

Alle namen van classes, id’s, names, teksten, titles, enz. zijn opgegeven in het script. In deze uitleg wordt uitgegaan van die namen. Omdat deze allemaal bovenin het script staan, zijn ze heel eenvoudig te wijzigen. Hoe je dat kunt doen staat bij Wijzigingen aanbrengen in het script.

Als een bedieningselement niet wordt gebruikt, hoeft dit alleen maar verborgen te worden met `display: none;`. Elk bedieningselement dat ergens niet wordt gebruikt in de voorbeeldspelers, is op deze manier verborgen. Dit geldt ook voor de elementen die zorgen voor het voorlezen van geluidssterkte en verstreken speelduur: verberg ze met `display: none;` en schermlezers negeren ze.
Achter de schermen blijft alles gewoon werken, alleen zie je het niet meer. Knoppen e.d. die op deze manier zijn verborgen, kunnen uiteraard niet worden gebruikt. Je kunt ’n verdwenen knop nou eenmaal niet aanraken of aanklikken of wat dan ook (mogelijk werkt telepathie, maar dat heb ik niet geprobeerd).

De spelers zijn volledig met het toetsenbord te bedienen. Knoppen werken met behulp van klikken, aanraking, Enter en/of de spatiebalk, afhankelijk van systeem en browser. De sleepbalken kunnen worden bediend met behulp van de pijltjes, PgUp en PgDn, en Home en

End. Hierbij worden toetscombinaties als Alt+← genegeerd, zodat de normale werking van dit soort sneltoetsen behouden blijft.
De breedte, hoogte, border, e.d. van de sleepbalk en van de sleepknop kunnen onafhankelijk van elkaar worden ingesteld met behulp van css. Bij het berekenen van de plaats van de knop past het script de positie aan breedte, borders, e.d. aan.
Alle bedieningselementen zijn te bereiken met de Tab-toets. Indien nodig kan op een simpele manier de tabindex worden aangepast. Als de tabindex wordt gebruikt, krijgen alle bedieningselementen van alle spelers automatisch een unieke tabindex van het script. .
Tussen de tabindexen van de spelers in is ruimte voor eventueel tussenliggende tabindexen. Dit staat verder beschreven bij Tabindex.
Ook bedoeld voor toetsenbordgebruikers is de sneltoets Control+→ of Control+← (in sommige browsers Control+Alt+→ of Control+Alt+←). Hiermee ga je naar de volgende of vorige video. Deze sneltoets wordt door het script aangebracht.

Zonder JavaScript worden geen bedieningselementen e.d. ingevoegd, en worden de bedieningsknoppen van de in de browser ingebouwde standaardvideospeler niet verwijderd. Zonder JavaScript werkt de standaardvideospeler dus precies zoals anders. (Overigens kun je in sommige browsers gewoon helemaal geen video afspelen zonder JavaScript, maar dat heeft niets met dit voorbeeld te maken.)

Als in de html bij het <video>-element een attribuut `data-smscr` met een bepaalde waarde is opgegeven, wordt bij een kleinere hoogte of breedte van het venster van de browser dan die waarde het hele JavaScript niet gebruikt. In de voorbeelden is `data-smscr="480"` gebruikt: in vensters smaller of lager dan 480 px wordt de in de browser ingebouwde standaardvideospeler gebruikt. Op een smartphone bijvoorbeeld werkt deze veel beter dan ’n videospeler met allerlei toeters en bellen, waar feitelijk geen ruimte voor is op zo’n klein schermpje. De waarde van 480 kan worden aangepast in de html, maar alleen de eerste bij `data-smscr` opgegeven waarde wordt gebruikt. Het heeft dus geen nut meerdere verschillende waardes op te geven.

Eigenlijk jok ik hierboven een beetje.. Als een maximale breedte of hoogte is opgegeven met behulp van `data-smscr`, wordt ’n heel klein stukje van het script wel gebruikt.
Als er zo’n maximale breedte of hoogte is opgegeven met behulp van `data-smscr`, wordt door het script bij elk <video>-element gekeken, of bij dat <video>-element ook een maximale breedte en hoogte is opgegeven met behulp van `data-smscr`.
Als dat zo is, worden van deze <video> alle <source>’s bekeken. Als ook een van de <source>’s een `data-smscr` heeft, wordt deze <source> gebruikt. Omdat dit alleen bij browservensters beneden een opgegeven breedte of hoogte gebeurt, geeft dit de mogelijkheid om voor kleinere schermen een kleinere video te gebruiken. Een smartphone heeft weinig aan een video van 1500 px breed. Meer hierover vind je bij data-smscr=”480”.

Hogeresolutieschermen

Vaak wordt dit soort schermen Retina-schermen genoemd, maar Retina is gewoon de merknaam van Apple voor een hogeresolutiescherm.
Er komen steeds meer apparaten met een hogere resolutie, dan op de desktop gebruikelijk is: meer dpi, dots per inch. Van oudsher wordt helaas ’inch’ gebruikt als eenheid. Een nieuwere eenheid is dpcm: dots per centimeter. De beste eenheid voor het meten van de resolutie van een computerscherm is dppx: dots per px unit, maar deze wordt nog niet lang niet overal ondersteund. 1 dppx is even groot als 96 dpi, de standaardresolutie van een gewone desktop monitor (Apple had een iets andere dpi).

Als je meer pixels in een inch stopt, waardoor de pixels dichter op elkaar staan, krijg je fijnere afbeeldingen. Vooral bij ronde lijnen e.d. is dit goed te merken. Een ronde lijn die wordt weergegeven met 72 px per inch is grover, dan diezelfde lijn die met 300 px per inch wordt weergegeven.
Een lijn die op een normale desktopmonitor 4 px breed is, zou op zo'n hogeresolutiescherm van 300 dpi maar 1 px breed zijn. Superduidelijk, dat wel, maar zonder vergrootglas niet te zien. Omdat de pixels vier keer zo dicht op elkaar staan. Bestaande sites die vier (of meer) keer zo klein worden weergegeven, als waar ze voor bedoeld zijn, maken alleen opticiens vrolijk.
Althans: je zou die kleinere weergave op een hogeresolutiescherm verwachten. Maar gelukkig wordt dat voorkomen. Apparaten met een hogeresolutiescherm geven een 'valse' resolutie op. Een iPad met een hogeresolutiescherm geeft niet het aantal 'scherm-pixels' op, maar het aantal 'css-pixels'. En dat is hetzelfde als bij een 'normale' desktopmonitor. Hierdoor ziet een site er op een iPad hetzelfde uit als op een desktopmonitor.
Niet alleen de iPad heeft dit handigheidje, alle hogeresolutieschermen doen dit. Anders zouden ze volstrekt onbruikbaar zijn voor vrijwel alle sites.

Met behulp van media queries kun je testen op resolutiedichtheid. Maar de ondersteuning hiervan is is nog sterk in ontwikkeling, daarom test ik hier (nog) niet op.
Er zijn nogal wat sites die alleen rekening houden met de iPad/iPhone en Safari. Door dat te doen, werk je mee aan de kans op net zo'n monopolie, als Internet Explorer ooit had. Dat heeft ons jarenlang grote ellende opgeleverd. Het lijkt me niet handig om dat nog eens te gaan herhalen met Apple, temeer niet omdat Apple zich steeds meer als een gediplomeerd patenttrol begint te gedragen en veelbelovende pogingen doet Microsoft van de troon te stoten als meest onsympathiek bedrijf in de ICT.
Persoonlijk zou ik gewoon wachten, tot dit in een standaard is opgenomen en voldoende wordt ondersteund door de diverse browsers. Mocht je het toch willen gebruiken, dan zou je in ieder geval álle vormen moeten gebruiken, en niet alleen die met `-webkit-`.
Op de pagina met links kun je onder CSS → Media Queries en Responsive Web Design links over dit onderwerp vinden.

Het <source>-element van <video> had tot voor kort een uiterst handig `media`-attribuut. Hiermee kon je o.a. testen op de resolutie van een scherm, de grootte, enz. Aan de hand daarvan werd dan bepaald, welke video moest worden gebruikt. Hiermee kon je op een heel eenvoudige manier voorkomen dat bijvoorbeeld een smartphone een video voor een breedbeeld-tv zou downloaden.
Tijdens het ontwikkelen van dit voorbeeld vond w3c – de club die de standaard voor html, css, e.d. vaststelt – het wel geestig om het `media`-attribuut te verwijderen uit het <source>-element. Terwijl dit al in alle browsers was ingebouwd en prima werkte. In plaats van deze perfect werkende simpele oplossing is nu een ingewikkelde JavaScript-oplossing nodig, die ook nog 'ns veel slechter werkt. Voor elke extra test moet je namelijk weer 'n extra stuk JavaScript inbouwen.
Dit is des te krankzinniger, omdat vrijwel tegelijkertijd bij <picture> een soort `media`-attribuut is toegevoegd. Niet alleen de wegen des Heeren blijken ondoorgrondelijk te zijn. Gevolg hiervan is dat je zonder JavaScript altijd de grote video ziet. Nu zullen er weinig smartphones zijn zonder JavaScript, maar het blijft vreemd om iets wat al overal is ingebouwd en prima werkt zonder goede reden te verwijderen.

De enige mogelijkheid om hogeresolutievideo's af te spelen is op dit ogenblik het gebruik van JavaScript. Wat nog eens extra ingewikkeld zou worden, omdat het testen op resolutie niet overal op dezelfde manier gebeurt (Apple ligt weer 'ns dwars en Microsoft loopt achter).

Op dit moment is eigenlijk de enige reële mogelijkheid om iedereen een hogeresolutievideo te sturen. Persoonlijk kies ik er dan voor om iedereen een eenvoudiger video te sturen, want je zadelt anders mensen met een simpel draadloos toestel ook op met een onwijs grote download.
Hopelijk komt er ooit een soortgelijke oplossing als voor afbeeldingen met <picture> is gevonden.

De voorvoegsels -moz-, -ms- en -webkit-

Voordat een nieuwe css-eigenschap wordt ingevoerd, is er in de regel een experimentele fase. Browsers passen het dan al toe, maar met een aangepaste naam. Tijdens deze fase kunnen problemen worden opgelost en worden veldslagen uitgevochten over hoe de standaard precies moet worden toegepast.
Als iedereen het overal over eens is en alle problemen zijn opgelost, wordt de officiële naam uit de standaard gebruikt.
De belangrijkste browsers hebben elk een eigen voorvoegsel:
Firefox: `-moz-`, naar de maker: Mozilla.
Op webkit gebaseerde browsers, zoals Google Chrome en Safari: `-webkit-`.
(Google Chrome is van webkit overgestapt op een eigen weergave-machine: blink. Blink zou geen voorvoegsels gaan gebruiken. Het is echter een aftakking van webkit, dus het zal nog wel even duren voor `-webkit-` hier helemaal uit is verdwenen. Ook Opera gebruikt inmiddels Blink, inclusief het daar nog in gebruikte `-webkit-`. Hierdoor is het tot voor kort door Opera gebruikte voorvoegsel `-o-` niet meer nodig.)
Internet Explorer: `-ms-`, naar de maker: Microsoft.

In dit voorbeeld worden `radial-gradient`, `column-count`, `column-gap`, `@keyframes`, `animation` en `transform` gebruikt.
Zodra de experimentele fase voorbij is, wordt het voorvoegsel weggelaten. Omdat dat moment niet bij alle browsers hetzelfde is, zet je nu ook al de officiële naam erbij. Deze wordt als laatste opgegeven. Bijvoorbeeld Android browser herkent `-webkit-radial-gradient`. Zodra Android browser `radial-gradient` gaat herkennen, zal dit `-webkit-radial-gradient` overrulen, omdat het er later in staat. Dat ze er beide in staan, is dus geen enkel probleem.

`radial-gradient`:

Op dit moment moet je nog het volgende schrijven:

```
{-webkit-radial-gradient: ...; radial
     gradient: ...;}
```

In de toekomst kun je volstaan met:

```
{radial-gradient: ...;}
```

Internet Explorer 9 kent `radial-gradient` helemaal niet, die negeert deze regel gewoon.
(De syntax van `radial-gradient` is nogal fors gewijzigd tijdens de proefperiode. Voor oudere op webkit gebaseerde browsers was er een syntax die begon met `-webkit-gradient(radial`. Die gebruik ik niet meer, want er zijn steeds minder browsers die die syntax gebruiken. Die krijgen gewoon geen gradiënt.)

`column-count en column-gap`:

Op dit moment moet je nog het volgende schrijven:

```
{-moz-column-count: ...; webkit-column-count: ...;
     column-count: ...; -moz-column-gap: ...;
     -webkit-column-gap: ...; column-gap: ...;}
```

In de toekomst kun je volstaan met:

```
{column-count: ...; column-gap: ...;}
```

`@keyframes`:

Op dit moment moet je nog het volgende schrijven:

```
@-webkit-keyframes pulse {...}
@keyframes pulse {...}
```

In de toekomst kun je volstaan met:

```
@keyframes pulse {…}
```

(In deze uitleg is `@-webkit-keyframes` vaak weggelaten, omdat dat tientallen pagina's scheelt. In de stylesheet op de site en 'afbeelding-103-5-dl.css' in de download is wel alles aanwezig.)

`animation`:

Op dit moment moet je nog het volgende schrijven:

```
{-webkit-animation: ...; animation: ...;}
```

In de toekomst kun je volstaan met:

```
{animation: ...;}
```

`transform`:

Op dit moment moet je nog het volgende schrijven:

```
{-ms-transform: ...; -webkit-transform: ...;
transform: ...;}
```

In de toekomst kun je volstaan met:

```
{transform: ...;
```

`hyphens`:

Op dit moment moet je nog het volgende schrijven:

```
{-moz-hyphens: ...; -ms-hyphens: ...; -webkit-
     hyphens: ...; hyphens: ...;}
```

In de toekomst kun je volstaan met:

```
{hyphens: ...;}
```

Het script voegt ook nog wat css in, zoals beschreven bij Door het script ingevoegde css:

```
{-ms-touch-action: none; touch-action: none;}
```

Hier is alleen het voorvoegsel `-ms-` nog nodig.

```
{-moz-user-select: none; -ms-user-select: none;
     -webkit-user-select: none; -user-select: none;}
```

Hier zijn de voorvoegsels `-moz-`, `-ms-` en `-webkit-` nog nodig.

Gegenereerde code

Het onderstaande geldt alleen voor desktopbrowsers. In browsers op mobiele systemen is het vaak ook mogelijk gegenereerde code te bekijken, maar dat is veel ingewikkelder. Bovendien verandert de manier, waarop dat kan, nogal snel.
Als je html schrijft, kan dat (hopelijk) in de browser worden weergegeven. Vanuit de browser kun je die html bekijken, precies zoals je hem hebt ingevoerd. Alsof je het in een editor ziet. In Firefox bijvoorbeeld kan dat door ergens op de pagina te rechtsklikken en te kiezen voor Paginabron bekijken, of door in het menu te kiezen voor Extra → Webontwikkelaar → Paginabron. (Of door de veel snellere sneltoets Ctrl+U.) Elke browser heeft dit soort mogelijkheden.
Wat je dan te zien krijgt, is exact de code, zoals jij die hebt ingetypt. Inclusief alle fouten, hoofd- en kleine letters, noem maar op. Als je zo slordig bent om een niet af te sluiten, zit er niet plotsklaps een afsluitende in de code. Als er css wordt gebruikt, html wordt ingevoegd via JavaScript, noem maar op, zie je daar niets van, want dat heb jij niet ingetypt. Daar heb je dus eigenlijk vrij weinig aan, want die code kende je al. Die heb je zelf bloedig zitten intypen.

Wat de browser daadwerkelijk gebruikt is iets totaal anders: de gegenereerde code. En die is veel interessanter, want die code blijkt (fors) af te wijken van wat jij heb ingetypt. De browser gebruikt een tijdelijke kopie van de door jou geschreven code, die zo is aangepast dat er voor de browser mee te werken is.

Elke browser heeft inmiddels mogelijkheden om de gegenereerde code te bekijken. In Firefox bijvoorbeeld in het menu Extra → Webontwikkelaar → Hulpmiddelen in-/uitschakelen. In Google Chrome in het menu onder Meer hulpprogramma's → Hulpprogramma's voor ontwikkelaars. In Internet Explorer open je dit door op F12 te drukken, en het kan vast ook via het menu.
Houdt er wel rekening mee dat elke browser de door jou ingetypte code iets zal aanpassen. In Firefox bijvoorbeeld wordt een <P> veranderd in een . Als er 'n mist, is die opeens wel aanwezig in de gegenereerde code. Wat elke browser precies aanpast, zal iets verschillen en kan ook nog veranderen. In het verleden veranderde Internet Explorer bijvoorbeeld een juist in een <P>, nu is dat niet meer zo.
Als je met behulp van JavaScript elementen invoegt (zoals in dit voorbeeld uitbundig gebeurt), zie je die alleen in deze gegenereerde code.
Veel browsers (misschien wel allemaal inmiddels) hebben ook uitbreidingen, waarmee gegenereerde code bekeken kan worden. In Firefox zijn dat bijvoorbeeld Firebug en Web Developer. Op de pagina met links vind je onder Gereedschap → Debuggen links naar allerlei hulpmiddelen, handleidingen, e.d. op dit gebied.

Semantische elementen en WAI-ARIA

Deze twee onderwerpen zijn samengevoegd, omdat ze veel met elkaar te maken hebben.

Semantische elementen

De meeste elementen die in html worden gebruikt, hebben een semantische betekenis. Dat wil zeggen dat je aan de gebruikte tag al (enigszins) kunt zien, wat voor soort inhoud er in het element staat. In een <h1> staat een belangrijke kop. In een <h2> staat een iets minder belangrijke kop. In een staat een alinea. In een <table> staat een tabel (en geen lay-out, als het goed is!). Enz.
Door het op de goede manier gebruiken van semantische elementen, kunnen zoekmachines, schermlezers, enz. de structuur van een pagina begrijpen. De spider van een zoekmachine is redelijk te vergelijken met een blinde. Het is dus ook in je eigen belang, om semantische elementen zo goed mogelijk te gebruiken. Een site die

toegankelijk is voor mensen met een handicap, is in de regel ook goed te verwerken voor een zoekmachine en maakt dus een grotere kans gevonden en bezocht te worden.
Als het goed is, wordt het uiterlijk van de pagina bepaald met behulp van css. Het uiterlijk staat hierdoor (vrijwel) los van de semantische inhoud van de pagina. Met behulp van css kun je een <h1> heel klein weergeven en een <h6> heel groot, terwijl schermlezers, zoekmachines, e.d. nog steeds weten dat de <h1> een belangrijke kop is.
Slechts enkele elementen, zoals en , hebben geen semantische betekenis. Daardoor zijn deze elementen uitstekend geschikt, om met behulp van css het uiterlijk van de pagina aan te passen: de semantische betekenis verandert niet, maar het uiterlijk wel. Voor een schermlezer of zoekmachine verandert er (vrijwel) niets, voor de gemiddelde bezoeker krijgt het door de css een heel ander uiterlijk.
(De derde laag, naast html voor de inhoud en css voor het uiterlijk, is JavaScript. Die zorgt voor de interactie tussen site en bezoeker. De min of meer strikte scheiding tussen css en html aan de ene kant en JavaScript aan de andere kant is met de komst van css3 en html5 veel vager geworden. Je kunt nu bijvoorbeeld ook met css dingen langzaam verplaatsen en met html deels de invoer in formulieren controleren.)

Html5 heeft een aantal nieuwe elementen, die speciaal zijn bedoeld om de opbouw van een pagina aan te geven. In dit voorbeeld worden hiervan <main>, <aside> en <nav> gebruikt. Alle drie gedragen zich als een gewone , maar dan een met een semantische betekenis. Hierdoor kunnen schermlezers, zoekmachines, e.d. beter zien hoe de pagina is samengesteld.

<main>: het deel van de pagina waar de belangrijkste inhoud begint. Meestal zal dit het deel na een inhoud, header, e.d. zijn. Omdat in deze pagina's geen header e.d. staan, staat <main> hier gelijk onder <body>.
(Er is iets voor te zeggen <main> gelijk boven de eerste video te zetten, omdat daar de belangrijkste inhoud begint. Maar dan zou heel makkelijk onbedoeld de pop-up met uitleg kunnen worden gepasseerd.)

<nav>: meestal gebruikt voor de belangrijkste navigatie. In deze pagina's wordt het op de pagina's met links rondom het rijtje met links naar vorige en volgende pagina en overzicht.

<aside>: bedoeld voor inhoud die min of meer los staat van de rest, voor extra informatie, kadertjes, dat soort dingen. Het wordt hier op de pagina met video's gebruikt voor de pop-up met uitleg, die onder het vraagteken zit.

Met behulp van dit soort nieuwe semantische elementen kan bijvoorbeeld een schermlezer in één keer een heel menu passeren en gelijk naar de echte inhoud bij <main> gaan. Alleen hebben deze nieuwe elementen één probleem: ze hebben in de praktijk nog weinig nut, omdat schermlezers e.d. ze nog niet herkennen. En voordat alle schermlezers e.d. ze herkennen, zijn we jaren verder, want veel schermlezers zijn peperduur. (Er zijn gelukkig ook gratis schermlezers, zoals het uitstekende open source NVDA).
Maar er is niets op tegen, om ze nu vast te gaan gebruiken. Op het moment dat schermlezers, zoekmachines, enz. ze gaan herkennen, is de pagina dan al hierop voorbereid. Extra werk is het eigenlijk niet. Het intypen van <footer> gaat vrijwel even snel als het intypen van , en zeker sneller dan het intypen van `<div id="footer">`.

WAI-ARIA-landmarks

Los van semantische elementen is er nog een hulpmiddel om pagina's beter toegankelijk te maken voor schermlezers e.d.: WAI-ARIA-ondersteuning. WAI-ARIA staat voor Web Accessibility Initiative – Accessible Rich Internet Applications. Vaak wordt het alleen ARIA genoemd, of aria-codes, of zoiets.
WAI-ARIA is een hele serie codes die bijvoorbeeld aangeven, welk deel van een pagina een menu, een header, een footer, enz. is. Daarnaast kan WAI-ARIA bij bijvoorbeeld een formulier aangeven, of iets is aangevinkt of niet. Voor blinden zijn dit soort zaken uiterst belangrijk (en dus ook voor zoekmachines).
Schermlezers e.d. kunnen al jaren WAI-ARIA-codes herkennen. Dat is nog niet het geval met de nieuwe semantische elementen uit html5, en dat zal ook nog wel even duren, want veel schermlezers zijn peperduur.
Een van de groepen met codes binnen WAI-ARIA bestaat uit de zogenaamde 'landmarks'. Daarmee worden belangrijke onderdelen van een pagina gemarkeerd. De landmarks komen gedeeltelijk overeen met de nieuwe semantische html5-elementen. Omdat schermlezers deze landmarks al wel herkennen, zet ik ze bij de corresponderende html-elementen. In de toekomst zijn ze niet meer nodig, maar dat zal nog wel even duren.
Een schermlezer kan niet alleen van kop naar kop, maar ook van landmark naar landmark springen. Daardoor kan bijvoorbeeld de navigatie toch snel worden gevonden, ook al herkent een schermlezer <nav> nog niet.
In dit voorbeeld gebruik ik de landmarks `main`, `navigation` en `complementary`. Elke landmark wordt voorafgegaan door `role=`, op dezelfde manier als dat bij classes of id's gebeurt:

```
<main role="main">
<nav role="navigation">
<aside role="complementary">
```

De regels `<main role="main">` en in iets mindere mate `<nav role="navigation">` zien er wat dubbelop uit, en dat is ook zo. Maar dat heeft dus te maken met de nieuwigheid van <main> en <nav>, die nog niet worden herkend, terwijl roles al wel worden herkend. Ooit kun je de roles hier gewoon helemaal weglaten.
Op de pagina met links vind je onder het kopje Toegankelijkheid → Forums, richtlijnen, links, artikelen en dergelijke meer info over WAI-ARIA, o.a. over hier niet gebruikte landmarks zoals <header> en <section>.

WAI-ARIA-codes binnen de videospeler

Voor iemand die niet of slecht kan zien, is een videospeler uiterst moeilijk te bedienen. Een <button> wordt door een schermlezer herkend, maar een waarin met behulp van css een sleepbalk is aangebracht, is zonder hulp niet te herkennen.
Met behulp van WAI-ARIA kunnen de knoppen e.d. voor geluidsregeling en afspelen worden gegroepeerd tot twee groepen: eentje voor geluid en eentje voor het afspelen. Dat maakt het 'n stuk overzichtelijker, als je afhankelijk bent van een schermlezer.
In de videospeler wordt een aantal WAI-ARIA-codes gebruikt. Deze worden automatisch toegevoegd door het script.
Omdat deze codes (en de meeste elementen waar ze in staan) door JavaScript worden ingevoegd, zijn ze niet zichtbaar als je gewoon de html bekijkt. Ze zijn alleen zichtbaar in de gegenereerde code.

Een lijstje van gebruikte codes, in alfabetische volgorde:

ARIA-CHECKED

Een radioknop (of een aankruisvakje) is wel of niet aangevinkt. De snelheid wordt door vier radioknoppen geregeld, waarvan er maar eentje aangevinkt kan zijn. De code voor de eerste radioknop, die voor de halve afspeelsnelheid, is als volgt:

```
<input id="speed-half-1" class="speed-half"
type="radio" name="knop-snelheid-1" value="0.5"
title="Halve snelheid" tabindex="0" aria-
checked="false" aria-controls="video-1" aria-
label="Halve snelheid">
```

In de hierboven staande code staat `aria-checked="false"`. Hierdoor weet de schermlezer, dat deze knop niet is aangevinkt.
Als de knop wel is aangevinkt, staat er `aria-checked="true"`.
Deze waarde wordt door het script ingevuld en kan niet rechtstreeks worden gewijzigd. Maar omdat de waarde wordt aangepast aan de keuze van de gebruiker, kan de waarde wel indirect worden aangepast door het aanvinken van een andere radioknop.

ARIA-CONTROLS

Wordt gebruikt om knoppen e.d. te koppelen aan het element dat ze bedienen. Dit wordt door het script bij elk bedieningselement ingevoegd.
De hieronder staande code hoort bij de Speel-pauzeerknop van de eerste video:

```
<button id="play-1" class="play not-playing"
     type="button" title="Afspelen"
     tabindex="0" aria-controls="video-1" aria-
     label="Afspelen">
```

Het script geeft automatisch een id aan elk <video>-element: 'video' gevolgd door een koppelteken gevolgd door een volgnummer. De eerste <video> heeft id="video-1". Deze id wordt ingevuld als waarde achter `aria-controls`, waardoor nu duidelijk is dat deze knop de eerste video bedient.
De waarde van `aria-controls` kan niet rechtstreeks worden gewijzigd. Je kunt echter wel bovenin het script de naam van de id voor de <video> wijzigen, dus indirect is deze waarde wel te wijzigen. Hoe dat moet, staat bij ClassId.

ARIA-DESCRIBEDBY

Als de speelduur nog onbekend is, wordt bij gebruik van bepaalde knoppen een melding geopend, zoals beschreven bij Speelduur nog onbekend. In de waarin deze melding verschijnt staat o.a. `aria-describedby="duration-par"`. Dit wordt door het script ingevoegd en kan niet (rechtstreeks) worden gewijzigd.
In dezelfde staat ook `role="dialog"`. `aria-describedby` verwijst naar de inhoud, in dit geval de tekst, van het bij `dialog` horende venstertje. 'duration-par' is de id van het element, waarin die inhoud staat. Dat is hier de <p> met de tekst van de melding.

De id kun je bovenin het script eventueel wel wijzigen, zoals beschreven bij Id. Het script zorgt er dan automatisch voor dat de gewijzigde id ook bij `aria-describedby` wordt gebruikt.

ARIA-HIDDEN="TRUE"

Wordt gebruikt om elementen voor schermlezers te verbergen, terwijl ze wel gewoon op het scherm blijven staan. Ze worden alleen door schermlezers genegeerd. Wordt ingevoegd door het script en kan niet worden gewijzigd.
Hier worden ze gebruikt om de twee 's met de knoppen van de sleepbalken voor geluid en afspelen te verbergen. Die sleepbalken werken feitelijk zonder knoppen, de knoppen zijn er alleen voor het uiterlijk. Voor een schermlezer zorgen ze alleen voor nodeloze verwarring. De code van de met de knop van de sleepbalk voor afspelen:

```
<div id="image-slider-button-1" class="image-
    slider-button" title="Sleepbalk afspelen"
    aria-hidden="true" style="position:
    absolute;"></div>
```

Het simpele `aria-hidden="true"` verbergt de hele knop voor schermlezers.
In de tweede video op de vierde pagina is de bediening verborgen, tot deze zichtbaar wordt gemaakt. Omdat dit verbergen voor een schermlezer zinloos is, wordt ook daar `aria-hidden` gebruikt voor enkele elementen.

ARIA-LABEL

Afhankelijk van de instellingen van de schermlezer wordt de tekst hiervan voorgelezen. Het is vergelijkbaar met de title die bij hoveren over een element verschijnt. De meeste knoppen e.d. hebben een `aria-label`. Bij de <button> hier iets boven bij `aria-controls` bijvoorbeeld wordt het ook gebruikt.
Als de speelduur nog onbekend is, wordt bij gebruik van bepaalde knoppen een melding geopend, zoals beschreven bij Speelduur nog onbekend. In de waarin deze melding staat, wordt door het script ook een `aria-label` ingevoegd met een korte omschrijving van de melding.
Het `aria-label` wordt door het script ingevoegd, maar de inhoud ervan kan op een simpele manier bovenin het script worden veranderd. Hoe dat moet, staat bij AriaLabel.

ARIA-LIVE

Wordt gebruikt om een wijziging voor te lezen. In de spelers wordt het gebruikt als de gebruiker volume of positie binnen de video verandert. Wordt ingevoegd door het script en kan niet worden gewijzigd.
Er zijn twee 's, waar dit wordt gebruikt: eentje voor het volume, en eentje voor de verstreken speelduur. Omdat dit alleen voor schermlezers is bedoeld, zijn beide 's met behulp van css links buiten het scherm gepositioneerd, zodat ze de lay-out niet verstoren.
De hieronder staande code hoort bij de voor het volume:

```
<span id="aria-volume-1" class="aria-volume" aria-
    live="polite"></span>
```

Als de gebruiker de geluidssterkte verandert, wordt de nieuwe sterkte door het script ingevuld in de . Waarna deze nieuwe waarde wordt voorgelezen.
`aria-live` kan verschillende waarden hebben. Hier wordt 'polite' gebruikt, in het Nederlands 'beleefd'. Dat wil zeggen dat de nieuwe waarde op een enigszins beleefde manier wordt voorgelezen en niet overal doorheen wordt gebruld, storend of niet.
Voor het voorlezen van de verstreken speelduur worden de woorden 'uur', 'minuut' en 'seconden' gebruikt. Voor het voorlezen van de geluidssterkte wordt het woord 'procent' gebruikt. Deze woorden worden door het script ingevoegd, maar zijn simpel te wijzigen. Hoe dat moet, staat bij Text.
Een verandering van verstreken speelduur wordt alleen voorgelezen, als handmatig voor- of achteruit is gegaan. Een verandering van volume wordt alleen voorgelezen, als de video niet speelt, of als de geluidssterkte op 0% of 100% wordt gezet. (Als de video speelt, hoor je het nieuwe volume zo ook wel. Het voorlezen van 0% en 100% is voor het geval de video geen geluid heeft en iemand wanhopig blijft proberen het geluid harder of zachter te zetten.)

ARIA-VALUEMAX

Het zusje van de hieronder staande `aria-valuemin`. Hiermee kan de hoogste waarde van bijvoorbeeld een sleepbalk worden opgegeven. Wordt gebruikt in de sleepbalken voor geluid en afspelen.
In de code iets hierboven staat `aria-valuemax="100"`: de grootste mogelijke waarde is 100, waarbij het geluid voluit open staat. Wordt ingevoegd door het script en kan niet worden gewijzigd.
Bij de sleepbalk voor het afspelen is de waarde hiervan 'Einde video': `aria-valuemax="Einde video"`. Deze tekst wordt door het script ingevoegd, maar kan simpel worden gewijzigd. Hoe dat moet, staat bij Text.

ARIA-VALUEMIN

Het broertje van de hierboven staande `aria-valuemax`.
Hiermee kan de kleinst mogelijke waarde van bijvoorbeeld een sleepbalk worden opgegeven. Wordt gebruikt in de sleepbalken voor geluid en afspelen.
In de code gelijk hierboven staat `aria-valuemin="0"`: de kleinst mogelijke waarde is 0, waarbij het geluid helemaal uit staat. Wordt ingevoegd door het script en kan niet worden gewijzigd.
Bij de sleepbalk voor het afspelen is de waarde hiervan 'Begin video': `aria-valuemin="Begin video"`. Deze tekst wordt door het script ingevoegd, maar kan simpel worden gewijzigd. Hoe dat moet, staat bij Text.

ARIA-VALUENOW

Voor de volledigheid vermeld ik dit ook, omdat het vaak samen met `aria-valuemin` en `aria-valuemax` wordt gebruikt. Het geeft de huidige waarde van een sleepbalk e.d. aan. Wordt hier niet gebruikt, omdat het voorlezen van de geluidssterkte op een andere manier door het script wordt geregeld. Mede omdat `aria-valuenow` de irritante gewoonte heeft om alle tussenliggende waarden ook voor te lezen, als je geluidssterkte of verstreken speelduur wijzigt.

ROLE="DIALOG"

Als de speelduur nog onbekend is, wordt bij gebruik van bepaalde knoppen een melding geopend, zoals beschreven bij Speelduur nog onbekend. In de waarin deze melding staat, staat ook `role="dialog"`. Dit maakt aan schermlezers duidelijk dat het hier om een pop-up gaat.

ROLE="GROUP"

Wordt gebruikt om elementen in een groep bij elkaar te zetten. Wordt ingevoegd door het script en kan niet worden gewijzigd.
Deze code komt binnen de speler op drie plaatsen voor:
Alle bedieningselementen van de videospeler staan binnen een met class=”controls”:

```
<div id="controls-1" class="controls"
     role="group">
```

(Voor de duidelijkheid is de door het script ingevoegde inline-style bij de hierboven weggelaten.)

Alle elementen die met geluid te maken hebben, staan binnen een met class=”sound”:

```
<div id="sound-1" class="sound" aria-
     label="sound controls" role="group">
```

Alle elementen die met afspelen te maken hebben, staan binnen een met class=”image”:

```
<div id="image-1" class="image" aria-
     label="image controls" role="group">
```

(Beide laatste ’s staan binnen de eerste . Het zijn subgroepen van `div.controls`.)

Afhankelijk van de instellingen zullen sommige schermspelers aankondigen dat er een bepaald soort groep wordt binnengegaan, wat de bediening overzichtelijker kan maken.

Bij een aantal videospelers staan de elementen voor geluid en afspelen niet bij elkaar. Hierdoor kunnen ze beter niet als groep worden gemarkeerd. Bij deze spelers wordt met behulp van een klein stukje JavaScript onderaan de html deze role verwijderd. Meer daarover bij Het JavaScript onderaan de html-bestanden.

ROLE="RADIOGROUP"

`<input type="radio">`, radioknoppen, worden prima herkend door een schermlezer. Maar op een of andere manier moet duidelijk worden gemaakt, welke radioknoppen bij elkaar horen. Wordt ingevoegd door het script en kan niet worden gewijzigd.
In de videospelers wordt de snelheid geregeld met behulp van radioknoppen, die in een staan. De code voor die is als volgt:

```
<div id="speed-1" class="speed" aria-
     label="Snelheid afspelen instellen"
     role="radiogroup">
```

De schermlezer kan nu aankondigen dat hier een groep radioknoppen staat.

ROLE="SLIDER"

Wordt gebruikt om aan te geven dat een element een sleepbalk (in het Engels 'slider') is. Omdat de sleepbalken voor volume en afspelen in een gewone staan, wordt het in de videospelers twee keer gebruikt. Wordt ingevoegd door het script en kan niet worden gewijzigd.
Onderstaande code hoort bij de , waarin de sleepbalk voor het volume staat:

```
<div id="sound-slider-beam-1" class="sound-slider-
     beam" title="Sleepbalk volume" tabindex="0"
     aria-controls="video-1" aria-label="Sleepbalk
     volume" role="slider" aria-valuemin="0" aria-
     valuemax="100">
```

Omdat ik vrees dat bovenstaande code mogelijk tot een hartverzakking van schrik leidt: ook dit wordt volledig door het script ingevuld. Je kunt bovenin het script wel dingen wijzigen, zoals de naam van de id en de title. Hoe dat moet staat bij Wijzigingen aanbrengen in het script.
Het stukje `role="slider"` zorgt ervoor dat de schermlezer weet dat dit een sleepbalk is. (En omdat deze ook met toetsen kan worden bediend, is hij vervolgens ook nog te bedienen door mensen die niet of slecht zien.)

ROLE="TIMER"

Deze role geeft aan dat het element een teller bevat die verstreken of resterende tijd aangeeft. Wordt in de spelers twee keer gebruikt: in een met de verstreken speelduur en in een met de resterende speelduur.
Wordt door het script ingevoegd en kan niet worden gewijzigd.

OVERIGE WAI-ARIA-CODES

ARIA-CHECKED en ROLE="RADIOGROUP"

Bovenaan de pagina's met video's kun je kiezen tussen de in de browser ingebouwde standaardspeler of de aangepaste speler. Dat gebeurt met behulp van twee radioknoppen. Deze radioknoppen staan in een , die als attribuut `role="radiogroup"` heeft.

`<input type="radio">`, radioknoppen, worden prima herkend door een schermlezer. Maar op een of andere manier moet duidelijk worden gemaakt, welke radioknoppen bij elkaar horen. Daar dient `role="radiogroup"` voor.
Een radioknop (of een aankruisvakje) is wel of niet aangevinkt. De keuze voor de speler wordt door twee radioknoppen geregeld, waarvan er maar eentje aangevinkt kan zijn.
De radioknoppen hebben als attribuut `aria-checked="true"` als de knop is geselecteerd, `aria-checked="false"` als de knop niet is geselecteerd. ('false' en 'true' worden door het script bijgehouden.)
Deze mogelijkheid om een speler te kiezen wordt ingevoegd door het script en kan daarom niet rechtstreeks worden gewijzigd. Je kunt hem wel volledig verbergen met behulp van css en ook kun je de getoonde tekst aanpassen. Hoe dat aanpassen moet, staat bij Text.

ARIA-DESCRIBEDBY

In de html staat bij de eerste video de volgende code:

```
<video data-smscr="480" aria-describedby=
    "titel-1" controls preload="metadata"
    poster="../103-images/s_brock_cellular_
    respiration.jpg">
```

Het stukje `aria-describedby="titel-1"` is gewoon in de html aanwezig, het wordt niet door het script aangebracht.
'titel-1' verwijst naar de id van een ander element, waar een beschrijving van de video, een titel, of iets soortgelijks staat. In dit geval verwijst het naar de <h2> gelijk boven de video, die als id 'titel-1' heeft. Op deze manier weet een schermlezer de titel van de video.
Zonder dit attribuut moet je maar raden wat voor video het is. Omdat dit attribuut dus nogal belangrijk is voor schermlezers (en dus ook zoekmachines), controleert het script of in het <video>-element een van de attributen `aria-describedby`, `aria-labelledby` of `aria-label` aanwezig is.
(`aria-labelledby` is een soortgelijk attribuut als `aria-describedby`, bij `aria-label` wordt de omschrijving van de video in de het <video>-element zelf gezet.)
Als geen van deze drie attributen aanwezig is, voegt het script een `aria-label` in. Dit krijgt als waarde 'video-1': `aria-label="video-1".` Bij de tweede video is de waarde 'video-2', enz. Dit is natuurlijk volstrekt waardeloos als informatie, maar in ieder geval weet de gebruiker dan het nummer van de video.
Veel beter is het zelf aanbrengen van een fatsoenlijke omschrijving van de video, die iets meer vertelt dan alleen het volgnummer.

ARIA-HIDDEN="TRUE"

Op de pagina's met video's kan een pop-up met uitleg worden geopend door aanraken van, klikken op of hoveren over het vraagteken. Ook kan de pop-up worden geopend door er met de Tab-toets naar toe te gaan. De hieronder staande code regelt dat:

```
<span id="focus-for-tab" aria-hidden="true"
    tabindex="0"></span>
<input id="checkbox-help" type="checkbox" aria-
    hidden="true">
<div id="wrapper-icons" aria-hidden="true">
    <label id="icons" for="checkbox-help"
    title="Openen of sluiten van het
    hulpschermpje" accesskey="8"></label>
</div>
```

Schermlezers hebben niets aan al dit gedoe. Daarom wordt de hele handel voor schermlezers verborgen met behulp van drie keer `aria-hidden="true"`. De hele code blijft nu gewoon zichtbaar en werken, behalve voor schermlezers.
In schermlezers wordt de uitleg nu altijd voorgelezen, maar met enkele toetsaanslagen kan deze worden gepasseerd.
(Je zou het passeren van de uitleg nog makkelijker kunnen maken door

<main> gelijk boven de eerste video te zetten, waar de belangrijkste inhoud feitelijk begint. Alleen zou dan heel makkelijk onbedoeld de pop-up met uitleg kunnen worden gepasseerd. Maar hier is zeker iets voor te zeggen.)

Tabindex

Links, invoervelden in formulieren, e.d. kunnen met behulp van de Tab-toets (of een soortgelijke toets) één voor één worden bezocht, in de volgorde waarin ze in de html voorkomen. Shift+Tab-toets keert de volgorde van de Tab-toets om. Dit is een belangrijk hulpmiddel voor mensen die om een of andere reden de muis niet kunnen of willen gebruiken. (En het is vaak ook veel sneller dan de muis, vooral in formulieren.)
In sommige browsers en/of besturingssystemen is dit vreemd genoeg standaard uitgeschakeld en is een zoektocht in de instellingen nodig om dit aan te zetten.
Als je met behulp van de Tab-toets een element hebt bereikt, heeft dit 'focus': als het een link is en je drukt op Enter, wordt de link gevolgd. Bij een tekstveld kun je tekst gaan invoeren. Enz. Normaal genomen wordt focus aangegeven door een lijntje rondom het element dat focus heeft. In de videospelers binnen dit voorbeeld is meestal op een andere manier aangegeven dat een bedieningselement focus heeft, bijvoorbeeld door een afwijkende achtergrondkleur.
De Tab-toets volgt de volgorde van de elementen in de html. Het maakt niet uit, hoe ze op het scherm staan. Als je met behulp van css de elementen van plaats verwisselt op het scherm, wordt toch gewoon de volgorde in de html gevolgd.
De werking van de Tab-toets kan worden veranderd met behulp van het attribuut `tabindex`: `<div tabindex="3">`. Deze zal nu als derde worden bezocht, ook al krijgt een simpele normaal genomen nooit bezoek van de Tab-toets.
Normaal genomen is het gebruik van een tabindex niet nodig. Het is zeker niet bedoeld om de bezoeker als een kangoeroe op een hindernisbaan van onder via links over rechts naar boven te laten springen. Maar soms kan het handig zijn voor kleinere correcties, als de normale volgorde in de html niet optimaal is.
Schermlezers blijven altijd de volgorde van de html volgen, dus als de tabindex sterkt afwijkt van de volgorde in de html, kan dat behoorlijk verwarrend zijn.

Tabindex kan drie verschillende waarden hebben: -1, 0 of een positief getal.
Op de pagina's met video's wordt tabindex behoorlijk vaak gebruikt. 0 en -1 op alle pagina's, een positief getal op enkele pagina's.

In principe is de volgorde bij gebruik van de Tab-toets als volgt: eerst worden alle positieve getallen in volgorde afgewerkt. Als twee tabindexen dezelfde waarde hebben, wordt de volgorde in de html aangehouden. Daarna worden alle tabindexen met '0' bezocht.
Als de bedieningselementen van de videospelers een `tabindex="0"` krijgen, en gewone links op de pagina krijgen helemaal geen tabindex, is het niet helemaal duidelijk welke volgorde gevolgd moet worden. Om die reden hebben op de eerste en tweede pagina, waar de bedieningselementen van de videospelers allemaal `tabindex="0"` hebben, alle links toch een `tabindex="0"`. Omdat de nummers van de tabindexen hetzelfde zijn, wordt nu gewoon de volgorde van de html aangehouden.

TABINDEX="-1"

Een negatieve waarde van -1 zorgt ervoor dat het element volledig wordt genegeerd door de Tab-toets. Zelfs een link met een negatieve tabindex wordt volledig genegeerd. Normaal genomen heeft een `tabindex="-1"` maar één nut: je kunt dan met behulp van JavaScript toch focus aan het element geven, zonder dat gebruikers van de Tab-toets erdoor worden gehinderd.

Op de pagina's met voorbeelden wordt dit gebruikt bij de pop-up met uitleg:

```
<aside id="wrapper-help" role="complementary"
    tabindex="-1">
```

Eigenlijk zou dit niet nodig moeten zijn, maar om een of andere reden sluit op Android 4.4.2 de pop-up niet na aanraken van het sluitkruisje. Daarvoor moet je eerst het scherm buiten de pop-up aanraken. Deze tabindex zorgt dat de pop-up gewoon met één aanraking sluit. Omdat het een negatieve tabindex is, heeft deze verder geen invloed op de volgorde van de Tab-toets.
Kennelijk krijgt op Android 4.4.2 de <aside> op een of andere manier ook focus, terwijl dat niet zo hoort te zijn.

Als de speelduur nog onbekend is en bepaalde knoppen worden gebruikt, verschijnt een melding dat die knoppen nog niet kunnen worden gebruikt, zoals beschreven bij Speelduur nog onbekend. Die melding moet Focus krijgen, maar dat kan niet zonder meer, omdat de melding in een gewone <p> staat. En alleen links, <button>'s, e.d. kunnen focus krijgen. Door aan de <p> het attribuut `tabindex="-1"` te geven, kan de <p> wel focus krijgen.

TABINDEX="0"

Veel elementen zijn niet te bereiken met behulp van de Tab-toets, omdat deze standaard maar een beperkt aantal elementen bezoekt, zoals links, knoppen en tekstvelden. In de videospeler zitten echter ook bedieningselementen in 's en 's. Die zouden dus nooit bereikt kunnen worden met behulp van de Tab-toets. Door deze elementen een `tabindex="0"` te geven, gaan ze gewoon meedoen. De , , of wat dan ook, wordt als het ware veranderd in 'n link, voor wat de Tab-toets betreft.
Op de pagina's met video's komt deze tabindex behoorlijk vaak voor. In de html staat de volgende code:

```
<span id="focus-for-tab" aria-hidden="true"
    tabindex="0"></span>
```

De tabindex zorgt er hier voor dat deze met behulp van de Tab-toets is te bereiken, focus kan krijgen. Verder is de volledig onzichtbaar, omdat hij leeg is. Door in de css op `#focus-for-tab:focus` te testen, kan de pop-up worden geopend zodra deze focus heeft. En omdat hij verder volledig leeg is, heeft hij geen enkele invloed op muis, touchscreen, enz.

In de videospeler zelf zit ook een aantal 's en 's met daarin bedieningselementen. Deze zijn voorzien van `tabindex="0"`, waardoor ze met de Tab-toets zijn te bereiken. Deze tabindexen worden automatisch door het script gegeven.
Niet alle bedieningselementen hebben dit feitelijk nodig, maar voor de duidelijkheid krijgen ze er allemaal eentje. Je vergeet er anders heel makkelijk eentje. Bovendien is niet helemaal duidelijk, hoe een combinatie van `tabindex="0"` en helemaal geen tabindex moet worden afgehandeld. Om alle problemen te voorkomen, worden daarom alle bedieningselementen en ook alle gewone links van een `tabindex="0"` voorzien, voor zover ze al geen andere tabindex hadden. (Dit geldt alleen voor de eerste en tweede pagina, de andere pagina's hebben een volledig aangepaste tabindex.)

TABINDEX="..."

Op de plaats van de puntjes moet een positief getal worden ingevuld: het volgnummer. Er is ruimte voor, afhankelijk van de browser, minimaal 32767 tabindexen, dus daar kun je aardig wat links e.d. mee bedienen.
Met behulp van een nummer kan de volgorde van de Tab-toets daadwerkelijk worden gewijzigd. Alle elementen met een tabindex worden in de volgorde van het nummer afgelopen. Elementen zonder tabindex of met `tabindex="0"` worden bezocht, nadat alle elementen met een positief volgnummer zijn afgewerkt.
Als je dat zou willen, zou je de Tab-toets als eerste naar 'n footer onderaan de pagina kunnen laten gaan, daarna helemaal naar boven en vervolgens naar het midden. En als je toevallig neoliberaal bent en alles in financiële waarde vertaalt, moet je dat vooral doen, want dit levert de fabrikanten van kalmerende middelen, psychiaters, enz. bergen extra werk op en stimuleert dus Sint Economie.
Voor alle niet-neoliberalen: niet doen. De tabindex is bedoeld voor kleine correcties, niet om van je bezoeker een door een jager achtervolgd konijn in een doolhof te maken. Een schermlezer volgt trouwens altijd de volgorde van de html, dus ook voor een schermlezer kunnen grote ingrepen uiterst verwarrend zijn.
Als je de tabindex gebruikt om de volgorde te veranderen, let er dan op dat je dat goed doet. Alle velden in een bestelformulier van een tabindex voorzien, maar dat bij de verzendknop vergeten, helpt niet bij je volgende gesprek over loonsverhoging.

De volgnummers van de tabindex hoeven niet op elkaar aan te sluiten. Het is zelfs beter als dat niet zo is. Lang geleden heb ik 'ns handmatig honderden tabindexen in zitten typen. Netjes op elkaar aansluitend, want tellen kan ik prima. Iets later moest er helemaal aan het begin eentje worden tussengevoegd. Aangezien de nummers zo fantastisch foutloos op elkaar aansloten, moest elk hoger nummer dus handmatig worden veranderd. Volgens mijn therapeut is dit de oorzaak van mijn nog steeds terugkerende angstdromen over genummerde schapen, waar zich plotseling een ongenummerde ram tussen probeert de dringen. Oftewel: zoiets doe je maar één keer.

Het beste is, als de volgorde van de html gewoon ook de meest logische volgorde voor de tabindex is. Dan heb je helemaal geen gedoe. Dit is het geval bij de eerste twee pagina's met video's. De bedieningselementen staan op het scherm in dezelfde volgorde als in de html, dus `tabindex="0"` (wat de volgorde van de Tab-toets niet verandert) is voldoende.
Als je de gegenereerde code van de eerste en tweede pagina bekijkt, zie je daarin de door het script ingevoegde tabindexen in de spelers staan. (In de pagina's in de download klopt dit helemaal, maar op pagina's op de site staan wel wat tabindexen bij een aantal links, vanwege de navigatie op de site.)

Omdat de tabindex wordt ingevoegd door het script, moet je eventuele veranderingen ook daar aanbrengen. Hoe dat kan, staat bij TabIndex. Op deze plaats staat alleen, wat het nut is van een tabindex.

Als je 'afbeelding-103.js' bekijkt, zie je daar bovenin vijftien keer iets staan als `chooseRadioTabIndex="0"`. Het eerste deel van dit woord is steeds verschillend, maar ze eindigen alle vijftien op `TabIndex="0"`. (Door te zoeken op 'tabindex' kun je ze snel alle vijftien vinden.) Dit zorgt ervoor dat de keuzemogelijkheid bovenaan de pagina voor de videospeler en de veertien bedieningselementen allemaal een `tabindex="0"` krijgen. (De eerste en tweede pagina met video's gebruiken dit script, de andere pagina's gebruiken andere scripts, zoals gelijk hieronder staat.)

Op de derde, vierde en vijfde pagina is de volgorde van de bedieningselementen op het scherm met behulp van css aangepast. Die volgorde is dus niet meer hetzelfde als die in de html. Zonder tabindex zou de Tab-toets niet de bedieningselementen volgen in de volgorde, zoals ze op het scherm staan, maar in de volgorde van de html. Precies zoals het hoort, maar op het scherm zie je een focus die ogenschijnlijk willekeurig van de ene naar de andere knop springt bij indrukken van de Tab-toets.
Het beste zou nog steeds zijn om de volgorde van de html aan te passen, maar daarvoor moet je nogal fors aan het script gaan sleutelen. En als je dat kunt, heb je vermoedelijk dit hele voorbeeld niet nodig.
Zoals hierboven al staat, wordt boven in het script aan vijftien elementen een tabindex toegekend. In 'afbeelding-103.js' is die tabindex altijd '0'. Maar in 'afbeelding-103-3.js', 'afbeelding-103-4.js' en 'afbeelding-103-5.js' krijgt die tabindex een andere waarde. Dat zijn de scripts voor respectievelijk de derde, vierde en vijfde pagina met video's.
(Dit is gelijk ook het enige verschil tussen die vier scripts: de bovenin ingevulde tabindex.)

Door bovenin achter `...TabIndex="0"` een ander getal in te vullen, kun je de volgorde veranderen, waarin de Tab-toets de bedieningselementen afloopt. Als je in 'afbeelding-103-3.js' zoekt naar 'tabindex', zie je dat de opgegeven tabindex niet 15 keer '0' is, maar 15 – 1 – 2 – 3 – 4 – 11 – 5 – 7 – 9 – 10 – 8 – 6 – 12 – 13 – 14.
Het deel van de naam voor `TabIndex="..."` geeft aan, bij welk element de tabindex hoort: `chooseRadioTabIndex` hoort bij de radioknoppen, waarmee de standaard- of aangepaste speler wordt gekozen. `playPauseTabIndex` hoort bij de Speel-pauzeerknop. Enz.
(Als de Engelse namen problemen opleveren: in 'afbeelding-103-met-commentaar.js' staan deze namen ook, met daarachter in het Nederlands bij welk bedieningselement ze horen.)
In het rijtje volgnummers hier gelijk boven staat op de zesde plaats '11'. Als zesde element wordt de sleepbalk voor het geluid ingevoegd. Maar bij gebruik van de Tab-toets wordt deze pas als elfde bezocht, niet als zesde. Op de elfde plaats staat '8', de knop om vijf procent vooruit te gaan, maar die wordt dus niet als elfde bezocht, maar als achtste bij gebruik van de Tab-toets.
Als je de gegenereerde code van de derde pagina bekijkt, kun je de bij de eerste videospeler de hierboven genoemde tabindexen zien staan. Met één klein verschil: ze zijn met 100 verhoogd. De sleepbalk voor het geluid heeft niet '11', maar '111' als tabindex. De knop om vijf procent vooruit te gaan heeft niet '8', maar '108'. Enz.
Bij de derde videospeler is de tabindex met 300 verhoogd. De de sleepbalk voor het geluid heeft '311' en de tabindex voor de knop om vijf procent vooruit te gaan '308'.
Bij elke videospeler wordt steeds dezelfde volgorde aangehouden, maar dan steeds 100 hoger dan de vorige videospeler. Anders zou elke Speel-pauzeerknop een tabindex van '1' krijgen, elke Zachter-knop '2', enz. Bij gebruik van de Tab-toets zouden dan eerst alle tabindexen met '1'worden afgelopen (alle Speel-pauzeerknoppen), daarna alle tabindexen met '2' (alle Zachter-knoppen), enz. Niet echt handig.
Door bij elke speler de nummering met 100 te verhogen, kan elk bedieningselement een uniek nummer krijgen. En voor het veranderen van de volgorde hoef je alleen maar het nummer van de tabindex bovenin het script te veranderen, zoals beschreven bij TabIndex.

Omdat er maar veertien bedieningselementen zijn en de teller van de tabindex bij elke speler met 100 wordt verhoogd, heb je steeds 86 vrije tabindexen tussen twee opeenvolgende videospelers. Die kunnen worden gebruikt om in de html tabindexen aan te brengen, die passen binnen de volgorde van de door het script aangebrachte tabindexen.

Op de derde pagina met video's staan in de gegenereerde code de volgende tabindexen (in de volgorde zoals ze in de html staan):

15: keuze voor standaard- of aangepaste speler, aangebracht door script
1: link naar de tweede pagina, handmatig aangebracht in de html
2: link naar overzicht, handmatig aangebracht in de html
3: link naar de vierde pagina, handmatig aangebracht in de html
-1: pop-up met uitleg, handmatig aangebracht in de html
4: met id="focus-voor-tab", handmatig aangebracht in de html
20: link naar origineel boven eerste video, handmatig aangebracht in de html
101 t/m 114: bedieningselementen in de eerste video, aangebracht door script
120 t/m 123: download-links onder de eerste video, handmatig aangebracht in de html
190: link naar origineel boven tweede video, handmatig aangebracht in de html
201 t/m 215: bedieningselementen in de tweede video, aangebracht door script
Enz.

Bij gebruik van de Tab-toets worden eerst de links naar vorige pagina, overzicht en volgende pagina bezocht, de '-1' wordt genegeerd, dan wordt de pop-up met uitleg geopend, vervolgens wordt de keuze voor standaard- of aangepast speler bezocht, daarna de link naar de originele video boven de eerste video, dan alle bedieningselementen van de videospeler, en ten slotte de download-links onder de eerste video. Waarna hetzelfde gebeurt met de tweede, derde, enz. video.
De handmatig aangebrachte tabindexen in de html staan, wat de volgnummers van de tabindexen betreft, tussen de door het script aangebrachte tabindexen, waardoor het voor de bezoeker één geheel lijkt. Bij gebruik van de Tab-toets worden de bedieningselementen van de videospelers nu in een logische volgorde afgelopen.

Normaal genomen is het het beste om de volgorde van de html aan te houden, zoals eerder uitgelegd. En als dat niet kan, zou je normaal genomen met de hierboven beschreven verandering van de tabindex kunnen volstaan. Of helemaal geen verandering, als de volgorde van de bedieningselementen niet al te vreemd is bij gebruik van de Tab-toets. Maar op de pagina's drie, vier en vijf met video's zijn allerlei verschillende modellen opgenomen. Waarbij de volgorde van de bedieningselementen ook binnen dezelfde pagina niet bij alle spelers hetzelfde is. Nogmaals: dit zal normaal genomen natuurlijk niet voorkomen, omdat het uiterst verwarrend is als videospelers op dezelfde pagina, of zelfs op dezelfde site, er allemaal anders uitzien.
Hier is dit echter wel het geval. Daarom wordt op de pagina's drie, vier en vijf onderaan de html met een klein scriptje de volgorde van de tabindex van sommige spelers nog eens aangepast.
Hoe dat scriptje werkt, en hoe je het eventueel kunt aanpassen, staat bij Het JavaScript onderaan de html-bestanden.
Dit is trouwens ook de reden dat afbeelding-103-5.js, het script voor de vijfde pagina, tabindexen met positieve nummers invoegt. Onderaan de html van de vijfde pagina wordt een aantal tabindexen veranderd. Daarom moeten álle met de Tab-toets te bezoeken elementen nu een positieve tabindex hebben, want elementen zonder tabindex of met `tabindex="0"` worden pas bezocht, nadat alle elementen met een positieve tabindex zijn afgelopen. Dat zou op deze pagina betekenen dat eerst een aantal knoppen van de derde, vierde, vijfde en zesde speler wordt bezocht met de Tab-toets, en daarna de rest van de pagina.

Muis, toetsenbord, touchpad en touchscreen

Vroeger, toen het leven nog mooi was en alles beter, waren er alleen monitors. Omdat kinderen daar niet af konden blijven met hun tengels, besloten fabrikanten dan maar

touchscreens te gaan maken, omdat je die mag aanraken. Het bleek makkelijker te zijn om volwassenen ook te leren hoe je ’n scherm vies maakt, dan om kinderen te leren met hun vingers van de monitor af te blijven.
Zo ontstonden touchscreens en kreeg het begrip ellende een geheel nieuwe lading. In de perfecte wereld van vroeger, waarin alleen desktops bestonden, werkten dingen als hoveren, klikken en slepen gewoon. Zonder dat je eerst ’n cursus hogere magie op Zweinstein hoefde te volgen. Zelfs in JavaScript was het nog wel te behappen, ook voor mensen zoals ik die toevallig niet Einstein heten.
Op dit moment kun je computerschermen ruwweg in twee soorten indelen: schermen die worden aangeraakt, en schermen die worden bediend met hulpmiddelen als een toetsenbord, muis of touchpad. Omdat ook computer-schermen zich kennelijk vermengen, bestaan er inmiddels ook schermen die zowel van aanraken als van muizen houden.
Hieronder staat een lijstje met dingen die zijn aangepast voor de verschillende soorten schermen, zodat dit voorbeeld overal werkt.

Vroeger...
Mocht je ooit ’n fossiel van mijn leeftijd tegenkomen die echt gelooft dat vroeger alles beter was, begin dan even gezellig over de tandarts in de zestiger jaren, waar je ’n klap voor je kop kon krijgen als je als kind bang was en spuiten ’n inhoud van twee liter en ’n botte naald van ’n halve centimeter dik hadden. Minstens. Als je geluk had, want verdoofd werd er alleen bij trekken. Als alleen de buitenmaatse klopboor voor beton werd gebruikt, werd er niet verdoofd. Ja, vroeger was alles veel beter... Maar ik dwaal mogelijk ’n heel erg klein ietsepietsie af.

:HOVER

Omdat `:hover` mogelijk niet werkt, als css uitstaat, ontbreekt of onvolledig is geïmplementeerd, moet je nooit belangrijke informatie alleen met behulp van `:hover` tonen.
Je hovert over een element, als je met behulp van muis of touchpad de cursor boven dat element brengt. Hoveren kan over álle elementen. Het wordt veel gebruikt om iets van uiterlijk te laten veranderen, een pop-up te laten verschijnen, e.d.
In dit voorbeeld verschijnen kleine venstertjes bij hoveren over een van de knoppen van de videospeler, sommige links, e.d., waarin de inhoud van het attribuut `title` zichtbaar is. In deze venstertjes staan hele korte tekstjes, zoals ’Afspelen’. Het zijn aanvullende tekstjes, als ze niet verschijnen is dat geen ramp. Afhankelijk van de browser kunnen ze ook op ’n touchscreen verschijnen.
De title wordt opgegeven in de html. In de bedieningselementen van de videospeler wordt de title ingevoegd door het script, maar je kunt de title wel wijzigen. Hoe dat kan, staat bij Title.

Belangrijker is de pop-up met uitleg die verschijnt, als je over het vraagteken op de pagina’s met video’s hovert. Om zoiets werkend te krijgen in iOS, Android en Windows Phone, wordt meestal JavaScript gebruikt. Waarbij er nogal wat verschillen in het benodigde JavaScript blijken te bestaan.
Gewoon hoveren kan uiteraard niet op een touchscreen. Je kunt natuurlijk met je neus boven het scherm gaan hangen en hopen dat het touchscreen denkt dat het ’n muis is (sommige neuzen...), maar ik geef je weinig kans op succes. Bovendien blijft het scherm ook nog steeds op aanraking reageren.
(Dit vraagteken is alleen zichtbaar in browservensters breder en hoger dan 480 px, want alleen daar verschijnen de aangepaste videospelers. De maat van 480 px is te wijzigen, meer hierover bij data-smscr=”480”.)

De pop-up blijft geopend, zodra je er met de muis op klikt. Bij gebruik van de Tab-toets opent de pop-up ook, als deze aan de beurt is. (Meer over de Tab-toets bij Tabindex.) Maar een touchscreen reageert ook op klikken: als je een touchscreen aanraakt, werkt dat ongeveer zoals een klik met de muis.
Kommer, kwel, levertraan en ellende.
Ik heb geen zin om JavaScript voor al die verschillende invoermethoden van allerlei systemen te leren. Inclusief de bergen bugs die er blijkbaar ook in zitten, verouderde versies, noem maar op.
Gelukkig blijkt het ook mogelijk met alleen css en html iets te construeren. Hoe het precies werkt, is te vinden bij Pop-up met uitleg, maar het komt neer op een kleine waarover je kunt hoveren, waarin ook checkboxen staan die uit- en aangevinkt kunnen worden. Alsmede een die zorgt dat de juiste afbeeldingen en teksten verschijnen, zoals het vraagteken.
De reageert op hoveren, de checkboxen reageren op klikken en aanraken. De css is ietwat ingewikkeld, maar het blijkt te werken. Op deze manier wordt hoveren en klikken uit elkaar gehaald, waardoor het overal werkt.
(Op de site zelf is dit inmiddels nog iets verder uitgewerkt, waardoor je inmiddels ook met het toetsenbord de pop-up geopend kunt laten en kunt sluiten. Maar dat zou dit voorbeeld nog uitgebreider maken, dus dat komt ooit wel 'ns in 'n apart iets.
Mocht je belangstelling hebben: o.a. de inhoudsopgave van de uitleg bij dit voorbeeld werkt op deze verder uitgewerkte manier.)

:FOCUS

Omdat `:focus` mogelijk niet werkt, als css uitstaat, ontbreekt of onvolledig is geïmplementeerd, moet je nooit belangrijke informatie alleen met behulp van `:focus` tonen.
De meeste mensen gaan met een muis naar een link, invoerveld, e.d. Waarna ze vervolgens klikken om de link te volgen, tekst in te voeren, of wat ze ook maar willen doen.
Er is echter 'n tweede manier om naar links, invoervelden, e.d. te gaan: met behulp van de Tab-toets (sommige browsers gebruiken andere toetsen, maar het principe is hetzelfde). Met behulp van de Tab-toets kun je naar links, invoervelden, e.d. (Over het gebruik van de Tab-toets staat meer bij Tabindex.) 'springen'. Waar je staat, wordt door alle browsers aangegeven met een of ander kadertje. De link e.d. met het kadertje eromheen heeft focus. Dat wil zeggen dat je die link volgt, als je op de Enter-toets drukt, of dat je in dat veld tekst kunt gaan invoeren, enz.
Eigenlijk is de focus alleen nuttig bij gebruik van een toetsenbord, maar om een of andere mij niet geheel duidelijke reden reageert iOS er ook op. Als je op 'n element 'n `:focus` én 'n `:hover` zet, blijft die `:focus` op iOS van kracht, tot je 'n andere link, knop, of zoiets aanraakt.
Dat leverde een probleem op bij de pop-up met de uitleg, want deze opent bij hoveren over het vraagteken én als de Tab-toets wordt gebruikt om het vraagteken focus te geven. Op iOS sluit deze pop-up dus pas, als een andere link, knop, of zoiets wordt aangeraakt. Om dat te voorkomen is het element dat voor `:focus` wordt gebruikt in een aparte gezet, die verder leeg is. iOS negeert die gewoon, waardoor het ook op iOS werkt zoals bedoeld. Hoe het precies werkt is te vinden bij Pop-up met uitleg.
Bij gewone links e.d. speelt dit probleem niet, want daar zit geen `:hover` op. Die werken dus gewoon op de normale standaardmanier op alle soorten schermen.

Verder wordt in dit voorbeeld `:focus` alleen gebruikt bij de knoppen van de videospeler (buiten de standaardwerking bij links e.d.) om aan te geven, welke knop `:focus` heeft. Dat is belangrijk voor gebruikers van de Tab-toets, die anders niet weten waar ze zitten. Als een bedieningselement van een videospeler focus heeft, krijgt het element een andere achtergrondkleur of zoiets, met behulp van `:focus`. Sommige browsers geven ook op een touchscreen een andere kleur aan de knop die als laatste is ingedrukt, maar dat is verder niet storend, dus daar hoeft niets aan te gebeuren.

Slepen

In een deel van de videospelers zitten twee sleepbalken: eentje voor de geluidssterkte en eentje voor het afspelen. (Ze zijn niet in alle spelers zichtbaar, omdat ze bij een aantal zijn verborgen met `display: none;`.)
Deze sleepbalken worden ingevoegd door het script, het zijn geen standaardsleepbalken. De bestaande sleepbalken zijn niet of nauwelijks aan te passen met css, en daar ging het juist om. Overigens werken deze balken (vrijwel) even goed als standaardbalken. (Eventuele problemen staan bij Bekende problemen (en oplossingen).)
Op een touchscreen sleep je met behulp van je vinger. Op andere schermen druk je de linkerknop van de muis in, sleept, en laat die knop los. Je zou zeggen dat dat enigszins op elkaar lijkt en enigszins op dezelfde manier zou moeten kunnen werken. Dat is niet zo. Slepen op een touchscreen werkt totaal anders dan slepen op andere schermen, en er zijn ook nog 'ns verschillen tussen diverse systemen. En bugs. En verouderde mobiele apparaten.
Er zijn twee mogelijkheden om dit op te lossen: bergen nieuw JavaScript (inclusief 37.389 bugs en afwijkingen in de verschillende browsers en systemen) leren, aan de whisky raken en totaal vereenzamen, of een door Microsoft bedacht systeem gebruiken: pointer events. Pointer events is bedacht door Microsoft, maar daarna aangemeld bij w3c, de club die over de standaarden voor css en html gaat. Inmiddels is het een standaard, wat inhoudt dat het vermoedelijk niet al te veel meer verandert. Pointer events maken geen onderscheid tussen muis, aanraken, e.d. In plaats van iets als `mousedown` of `touchbegin` gebruik je `pointerdown`. In Internet Explorer zijn pointer events standaard ingebouwd. Firefox is bezig ze te implementeren. Apple heeft aangekondigd ze niet te gaan gebruiken, die houden liever vast aan hun zelf ontwikkelde hopeloos ingewikkelde systeem. Omdat Apple ze niet gaat gebruiken in Safari, gaat Chrome (en dus ook Opera) dit ook niet doen, maar als genoeg mensen pointer events gewoon gaan gebruiken, is het goed mogelijk dat Chrome alsnog overstag gaat. En dan zal Apple ook wel moeten, vroeger of later, en ophouden met zich als de nieuwe Internet Explorer 6 te gedragen. De overgrote meerderheid van sitebouwers lijkt er in ieder geval enthousiast over te zijn.
Hoe dan ook: voor de videospelers heb ik gebruik gemaakt van pointer events. Microsoft heeft een zogenaamde 'polyfill' gemaakt: hand.js. Met behulp van deze polyfill zijn pointer events te gebruiken in alle geteste browsers (behalve Opera Mini, maar dat is een geval apart.)
(Als iemand mij drie jaar geleden had verteld dat ik iets van Microsoft zou aanbevelen, en dat Microsoft dat had aangeboden als standaard aan w3c, had ik die persoon absoluut voor gek verklaard. In eerdere voorbeelden heb ik 'n paar keer gebruik gemaakt van modernizr om te proberen touchscreens te herkennen. Maar dat is niet een echt heel erg betrouwbare manier en levert bergen extra css op. Pointer events werkt gewoon veel beter.)

Aangezien ik me niet rechtstreeks heb verdiept in hand.js, kan ik daar verder weinig over melden. In het script zelf worden gewoon eventlisteners als `pointerdown` gebruikt. hand.js wordt aan de pagina gelinkt en loopt door dit script heen, waarbij het deze pointer events vervangt door code, die andere browsers wel kennen. Waardoor die ook met pointer events overweg kunnen (maar in werkelijkheid dus heel andere JavaScript te eten krijgen). Meer weet ik er eigenlijk niet van, en ik weet niet eens honderd procent zeker of dit laatste stukje helemaal correct is.

Repeterende toetsen en aanrakingen

Als je op een gewoon toetsenbord een toets ingedrukt blijft houden, gaat deze repeteren. Hetzelfde kan gebeuren als je de muis ingedrukt blijft houden boven ’n knop. In de videospelers zit een aantal knoppen, die gaan repeteren als je ’n bepaalde toets of de muis ingedrukt blijft houden, zoals de knoppen voor Zachter of Harder. Het vaststellen van een eerste toetsaanslag is niet zo moeilijk, maar vaststellen of een toets (of de muis) ingedrukt blijft, is wat lastiger. En helemaal als je ook met touchscreens te maken hebt. Ook de sleepbalken werken met toetsbediening, wat weer anders behandeld moet worden, dan een knop die simpel ingedrukt blijft worden.

De hele bediening van de videospelers wordt afgehandeld met behulp van JavaScript. Gedeeltelijk met ’ouderwetse’ eventlisteners als `keydown`, gedeeltelijk met behulp van eventlisteners als `pointerdown`. Voor het afhandelen van pointer events wordt weer gebruik gemaakt van hand.js, op de manier zoals gelijk hierboven is beschreven bij Slepen.

Omdat niet elk touchscreen even goed registreert als een knop weer wordt losgelaten, blijft een knop soms na loslaten repeteren. Dit lijkt met de kwaliteit van het touchscreen te maken te hebben. Om die reden stoppen de knoppen Harder en Zachter altijd na een verandering van 20%, de knoppen Vijf procent verder en terug na 50%, en de knoppen Tien seconden verder en terug na 100 seconden. Om verder te gaan met repeteren, moet je ze nogmaals aanraken.

Voorwaarden waaraan de html en css moeten voldoen

Om het script goed te laten werken, moet de html aan drie simpele eisen voldoen.

De eerste eis: het <video>-element moet binnen een element met class=”videobox” staan. Op de pagina’s met voorbeelden is dat een simpele `<div class="videobox">`. Binnen deze staat het <video>-element. <video>’s die niet binnen een element met deze class staan, worden door het script volledig genegeerd. De <video> moet een direct kind van het element met class=”videobox” zijn.

Zo’n <video> buiten `div.videobox` werkt nog steeds gewoon, maar met de ingebouwde standaardspeler, niet met de door het script ingevoegde speler.

```
<div class="videobox">
    <video>
    </video>
</div>
```

is goed, maar

```
<div class="videobox">
    <div id="ik-hoor-hier-niet">
        <video>
        </video>
    </div>
</div>
```

levert een foutmelding op, omdat de <video> geen direct kind is van `div.videobox`: `div#ik-hoor-hier-niet` staat ertussen.
Voor en na <video> kunnen wel allerlei andere elementen staan, als <video> maar een direct kind is van `div.videobox`. Het volgende is dus ook correct:

```
<div class="videobox">
    <h1></h1>
    <p></p>
    <video></video>
    <p></p>
</div>
```

want ook hier is <video> een direct kind van `div.videobox`.
In theorie kan ook een ander element dan een worden gebruikt, maar een lijkt me hier het meest geschikt voor.
Als je 'videobox' wilt wijzigen in een andere classnaam, kun je bij ClassId vinden, hoe dat moet. Lees vooral het stukje onder `wrapperDivClassId` ook, want deze ClassId is een wat apart geval.

Elk element met class="videobox" móét een <video> bevatten. Als dat niet zo is, loopt de telling van het script volledig in het honderd. Als er een element met class="videobox" zonder <video> op de pagina staat, geeft het script een foutmelding.

En dan de derde en laatste eis: er moet een <body> aanwezig zijn. In html5 is het toegestaan <body> weg te laten. Wat daar het nut van is, is mij een compleet raadsel, maar het mag. Als je <body> weglaat, voegt de browser trouwens zelf een <body> toe.
Het script heeft <body> nodig om bepaalde elementen in te kunnen voegen. In theorie gaat dat ook goed, als je <body> weglaat, maar ik zou er niet op vertrouwen dat elke browser <body> altijd op precies de juiste plaats invoegt. En als dat niet gebeurt, kan dat leiden tot 'n soort Frankenstein: een lichaam dat op de verkeerde plaats begint. Met als gevolg een script dat op de verkeerde plaats dingen invoegt.

Voor de css gelden eigenlijk geen eisen, op eentje na: gebruik geen inline-style bij <body> (css in de <body>-tag zelf). Als je kiest voor de aangepaste of de ingebouwde videospeler, wordt – vanwege een bug in Safari op OSX – een inline-style toegevoegd aan <body>. Als daar al een inline-style staat, wordt die overschreven. Meer hierover bij document.body.setAttribute("style", "zoom: 1.0001");. Dit zou eigenlijk geen problemen moeten opleveren, omdat het gebruik van inline-styles altijd een bijzonder slecht idee is. Hiernaast is een kleine hoeveelheid css absoluut noodzakelijk voor een goede werking, maar die wordt automatisch door het script ingevoegd als inline-styles. Wat het script precies invoegt, staat bij Door het script ingevoegde css. Omdat deze css, op het hierboven genoemde na, allemaal wordt ingevoegd binnen de elementen die het script zelf maakt, levert dit geen problemen op met andere css.

Uitschakelen en verplaatsen van bedieningselementen

Als een bepaald bedieningselement niet wordt gebruikt, is dit uiterst simpel te verwijderen met `display: none;`. Elke groep gelijke bedieningselementen heeft dezelfde class. Zo hebben bijvoorbeeld alle Speel-pauzeerknoppen de class 'play'.

```
.play {display: none;}
```

verwijdert in één keer al die knoppen, uit alle spelers.
Daarnaast heeft elk individueel bedieningselement ook een eigen id. Je kunt dus ook in slechts één of meer spelers, maar niet in allemaal, elementen verwijderen:

```
#play-1 {display: none;}
```
verwijdert alleen de Speel-pauzeerknop van de eerste speler.
Een overzicht van namen van classes en id's vind je bij Overzicht van door het script ingevoegde bedieningselementen, classes en id's.
Als een bedieningselement wordt verborgen met behulp van `display: none;` wordt dit ook door schermlezers genegeerd.
Overigens worden de elementen niet écht verwijderd. Op de achtergrond blijft alles gewoon werken. Als je de gegenereerde code bekijkt, zie je dat ook elementen met `display: none;` gewoon nog aanwezig zijn, maar alleen op het scherm niet meer zijn te zien.

Het verbergen van de keuzemogelijkheid tussen aangepaste en ingebouwde standaardvideospeler aan de bovenkant van het venster van de browser gebeurt op precies dezelfde manier:
```
#kiezen-div {display: none;}
```
verwijdert de hele keuzemogelijkheid. Uiteraard kun je dan niet meer kiezen voor de ingebouwde standaardvideospeler.

De bedieningselementen van de videospelers worden door het script altijd direct na het <video>-element ingevoegd in de html. Dit kan niet worden gewijzigd. Maar met behulp van css kunnen de bedieningselementen overal worden neergezet: boven de video, onder de download-links, rondom de video, enz. Je zou ze zelfs een heel eind bij de video vandaan kunnen zetten, of de bediening voor de eerste video bij de tweede video.

De bedieningselementen zelf worden door het script in een bepaalde volgorde ingevoegd in de html. Die volgorde kan niet worden veranderd. Wel kunnen de elementen met behulp van css overal worden neergezet, ook in een andere volgorde dan waarin ze zijn ingevoegd. Beter is trouwens om die volgorde zoveel mogelijk aan te houden, want bijvoorbeeld schermlezers negeren de css en blijven de volgorde in de html volgen.
Bij een aantal spelers is de volgorde uit de html niet aangehouden. Links, knoppen, e.d. kunnen ook met behulp van de Tab-toets worden bezocht, als mensen om een of andere reden de muis niet kunnen of willen gebruiken. Om te zorgen dat bij die spelers met gewijzigde volgorde de Tab-toets de volgorde op het scherm volgt, en niet die in de html, is bij die spelers de tabindex aangepast. Meer over de tabindex kun je vinden bij Tabindex.

Werking van knoppen en sleepbalken

De bediening van knoppen en sleepbalk van de videospeler wordt volledig geregeld door het script. Hier is niets aan te veranderen, althans niet zonder het script fors aan te passen. Er is zoveel mogelijk geprobeerd aan te sluiten op de standaardmanier, waarop knoppen e.d. op de diverse systemen werken.
De bediening wordt volledig afgehandeld door het script. Het uiterlijk van de bedieningselementen heeft geen enkele invloed op de werking ervan, omdat het uiterlijk volledig door css wordt bepaald. Voor breedte e.d. van de sleepbalken worden automatisch aanpassingen gemaakt, zodat de knoppen van de sleepbalken altijd op de goede plaats staan, ongeacht breedte, randen, e.d.

Speel-pauzeer

Muis: klikken op de knop pauzeert of start het afspelen van de video.
Aanraken: aanraken van de knop pauzeert of start het afspelen van de video.
Toetsenbord: als de knop focus heeft, pauzeren of starten spatiebalk en Enter het afspelen van de video.

De Speel-pauzeerknop heeft nog een extra sneltoets: met Control+← of Control+Alt+← (afhankelijk van de browser) wordt naar de vorige Speel-pauzeerknop gegaan. Met Control+→ of Control + Alt+→ (afhankelijk van de browser) wordt naar de volgende Speel-pauzeerknop gegaan.
Als de knop in de pauzestand staat, wordt in de html door het script de class 'not-playing' toegevoegd. Als de knop in de afspelenstand staat, wordt deze class weer verwijderd. (Dit kan door css gebruikt worden om het uiterlijk van de knop aan de stand aan te passen.)
De title bij deze knop is afhankelijk van de stand van de knop. Dit wordt door het script geregeld.
De door een schermlezer voor te lezen tekst bij deze knop is afhankelijk van de stand van de knop. Dit wordt door het script geregeld.

Zachter

Muis: klikken op de knop vermindert de geluidssterkte met 1%. Als de knop ingedrukt blijft, gaat de knop na een korte pauze repeteren. Na een verlaging van 20% moet de knop opnieuw worden aangeklikt.
Aanraken: aanraken van de knop vermindert de geluidssterkte met 1%. Als de knop aangeraakt blijft worden, gaat de knop na een korte pauze repeteren. Na een verlaging van 20% moet de knop opnieuw worden aangeklikt.
Toetsenbord: als de knop focus heeft, verminderen spatiebalk en Enter de geluidssterkte met 1%. Als spatiebalk en Enter ingedrukt blijven, gaat de knop na een korte pauze repeteren.
Door het bedienen van deze knop wordt de stand van de sleepbalk voor het geluid beïnvloed. Dit wordt door het script geregeld.
Door het bedienen van deze knop wordt het getoonde percentage geluidssterkte beïnvloed. Dit wordt door het script geregeld.
Terwijl de geluidssterkte verandert, wordt de verandering weergegeven in sleepbalk en percentage. De eindstand is gelijk aan de nieuwe geluidssterkte.
Als de geluidssterkte met behulp van deze knop is gewijzigd, wordt de nieuwe geluidssterkte in schermlezers voorgelezen. Alleen de laatste sterkte wordt voorgelezen. Als de video speelt, wordt geen geluidssterkte voorgelezen, omdat dat overbodig is (je kunt het gewoon horen). Maar voor het geval de video geen geluid heeft, worden 0% en 100% ook voorgelezen, als de video speelt.

Harder

Werkt precies hetzelfde als de knop Zachter gelijk hierboven. Alleen neemt de geluidssterkte bij deze knop toe en niet af (ja, inderdaad, daarom heet de knop Harder. Slim!).

Geluid aan-uit

Muis: klikken op de knop zet het geluid aan of uit.
Aanraken: aanraken van de knop zet het geluid aan of uit.
Toetsenbord: als de knop focus heeft, zetten spatiebalk en Enter het geluid aan of uit.
Als het geluid uitstaat, wordt in de html door het script de class 'muted' toegevoegd. Als het geluid aan staat, wordt deze class weer verwijderd. (Dit kan door css gebruikt worden om het uiterlijk van de knop aan de stand aan te passen.)
Het aan- of uitzetten van het geluid heeft geen invloed op de ingestelde geluidssterkte. Als het geluid wordt uit- en weer aangezet, is de geluidssterkte hetzelfde als voor het uitzetten. Deze knop heeft dan ook geen invloed op de stand van de sleepbalk voor geluid of op het getoonde percentage geluidssterkte.

De title bij deze knop is afhankelijk van de stand van de knop. Dit wordt door het script geregeld.
De door een schermlezer voor te lezen tekst bij deze knop is afhankelijk van de stand van de knop. Dit wordt door het script geregeld.

Sleepbalk geluid

De sleepbalk voor het geluid kan rechtstreeks worden bediend, maar ook de knoppen Harder en Zachter passen de sleepbalk aan. Dit regelt het script. De knop Geluid aan-uitknop heeft geen invloed op de sleepbalk.

De sleepbalk bestaat uit drie geneste 's, waarvan eigenlijk alleen de middelste belangrijk is voor de werking. De buitenste is alleen voor de css van belang, de binnenste is de knop en die staat alleen maar voor de show knop te wezen: zonder knop werkt de sleepbalk precies hetzelfde.
De middelste is de eigenlijke sleepbalk. Aanraken, klikken, enz. werken alleen boven deze . Het is ook deze middelste die focus kan krijgen.

Muis: door op een bepaalde plaats op de sleepbalk te klikken, wordt de geluidssterkte ingesteld afhankelijk van de plaats, waar is geklikt. Als je op driekwart vanaf de linkerkant op de sleepbalk klikt, wordt de geluidssterkte ingesteld op driekwart: 75%.
Als de knop van de muis ingedrukt blijft, kan de knop worden versleept. Omdat je makkelijk onbedoeld buiten de sleepbalk kunt komen tijdens het slepen, wordt pas gestopt met slepen, als je de muis niet meer indrukt. Zolang de knop van de muis blijft ingedrukt, kun je teruggaan naar de sleepbalk en verdergaan met slepen.
Aanraken: werkt precies hetzelfde als de muis hierboven, maar dan met je vinger.
Toetsenbord: als de sleepbalk focus heeft, kan deze ook met het toetsenbord worden bediend. Dit gebeurt met behulp van de pijltjestoetsen, PgUp, PgDn, End en Home.
← of ↓: 1% zachter;
→ of ↑: 1% harder;
PgDn: 10% zachter.
PgUp: 10% harder;
End: geluidssterkte 100%;
Home: geluidssterkte 0% (uit);
Deze toetsen worden vaak, in combinatie met bijzondere toetsen als de Alt-toets, als sneltoets voor andere dingen gebruikt, zoals een pagina teruggaan in de browser. Als de balk focus heeft, zouden deze sneltoetsen niet meer werken. Daarom werken de genoemde toetsen alleen als de Alt-, Control-, Shift-, Windows-, Meta-, Command-, Super-, Option-, welke-fantasienaam-dan-ook-speciale-toets niet is ingedrukt. Met Alt+← (of Command+←) ga je dus ook 'n pagina terug, als de sleepbalk voor geluid toevallig focus heeft.

Tijdens het veranderen van de geluidssterkte wordt de weergave in procenten voortdurend aangepast. De eindstand is gelijk aan de nieuwe geluidssterkte.
De nieuwe geluidssterkte wordt door een schermlezer voorgelezen. Dit gebeurt pas, als de uiteindelijke geluidssterkte is bereikt, dus niet tijdens slepen of zo, want dan zou je 'n hele serie nutteloze tussenliggende percentages horen. Als de video speelt, wordt de nieuwe geluidssterkte niet voorgelezen, behalve als die 0% of 100% is.

Naar begin

Muis: klikken op de knop brengt de video terug naar het begin.
Aanraken: aanraken van de knop brengt de video terug naar het begin.

Toetsenbord: als de knop focus heeft, brengen spatiebalk en Enter de video terug naar het begin.
Door het bedienen van deze knop gaat de sleepbalk voor het afspelen terug naar het begin. Dit wordt door het script geregeld.
Door het bedienen van deze knop worden de verstreken en resterende speelduur aangepast. Dit wordt door het script geregeld.
Na het bedienen van deze knop leest een schermlezer ’Begin video’ voor.

Naar einde

Precies hetzelfde als Naar begin hierboven, maar dan naar het einde van de video.

Vijf procent terug

Muis: klikken op de knop zet de video vijf procent van de speelduur achteruit. Als de knop ingedrukt blijft, gaat de knop na een korte pauze repeteren. Nadat tien keer is gerepeteerd (50%), moet de knop opnieuw worden aangeklikt.
Aanraken: aanraken van de knop zet de video vijf procent van de speelduur achteruit. Als de knop aangeraakt blijft worden, gaat de knop na een korte pauze repeteren. Nadat tien keer is gerepeteerd (50%), moet de knop opnieuw worden aangeraakt.
Toetsenbord: als de knop focus heeft, zetten spatiebalk en Enter de video vijf procent van de speelduur achteruit. Als spatiebalk en Enter ingedrukt blijven, gaat de knop na een korte pauze repeteren.
Door het bedienen van deze knop wordt de stand van de sleepbalk voor het afspelen beïnvloed. Dit wordt door het script geregeld.
Door het bedienen van deze knop worden verstreken en resterende speelduur beïnvloed. Dit wordt door het script geregeld.
Terwijl de verstreken en resterende speelduur veranderen, worden deze weergegeven. De eindstand is gelijk aan de nieuwe verstreken en resterende speelduur.
Als de verstreken speelduur met behulp van deze knop is gewijzigd, wordt de nieuwe verstreken speelduur in schermlezers voorgelezen.

Tien seconden terug

Precies hetzelfde als Vijf procent terug hierboven, maar dan met tien seconden terug.

Tien seconden vooruit

Precies hetzelfde als Vijf procent terug hierboven, maar dan met tien seconden vooruit.

Vijf procent vooruit

Precies hetzelfde als Vijf procent terug hierboven, maar dan met vijf procent van de speelduur vooruit.

Fullscreen

Muis: klikken op de knop laat de video fullscreen afspelen.
Aanraken: aanraken van de knop laat de video fullscreen afspelen.
Toetsenbord: als de knop focus heeft, laten spatiebalk en Enter de video fullscreen afspelen.
(Als de video fullscreen afspeelt, is deze knop uiteraard verdwenen. De browser bepaalt, hoe je weer naar normale weergave teruggaat.)

Het fullscreen afspelen wordt door het script geregeld met behulp van `requestFullScreen()`. Oudere browsers kennen dit niet. In dat geval wordt deze methode min of meer geïmiteerd door het script. Dit gebeurt o.a. door css in te

voegen, waarover meer te vinden is bij css die tijdelijk wordt ingevoegd, zoals alleen tijdens fullscreen afspelen.
Daarnaast wordt de waarde van het attribuut `data-fullscreen` bij deze knop van 'no' veranderd in 'on', zolang fullscreen wordt afgespeeld. Hierdoor kun je met eigen css bijvoorbeeld een melding geven, hoe je weer terug naar normale weergave kunt gaan. Een voorbeeld daarvan vind je bij .fullscreen[data-fullscreen="on"]::after.
Teruggaan naar de normale weergave gaat in dit geval door het aanraken van of klikken op de video, of door het indrukken van Escape.
Maar deze dingen gebeuren dus alleen, als de browser `requestFullScreen()` niet kent. Als de browser `requestFullScreen()` kent, wordt het afgehandeld door het script en de browser.

Snelheid

De snelheidsregeling wordt bediend door vier radioknoppen, met bijbehorende <label>'s. De knoppen staan buiten de <label>, omdat de snelheidsregeling daardoor makkelijker met css is op te maken. (De <label>'s worden door het script wel met behulp van een `for` gekoppeld aan de bijbehorende knop.)
Bij openen van de pagina is de snelheid altijd normaal. De snelheid kan veranderd worden naar halve snelheid, anderhalf keer de normale snelheid en dubbele snelheid.
De verandering van snelheid vindt plaats, zodra een radioknop wordt aangevinkt, zonder dat verdere bevestiging nodig is. Dat is geen probleem, want de verandering van snelheid wordt ongetwijfeld opgemerkt. Als het een vergissing is, kan die simpel worden hersteld.
Muis: klikken op één van de radioknoppen (of de bijbehorende <label>) verandert de snelheid in de snelheid die bij de gekozen knop hoort.
Aanraken: aanraken van één van de radioknoppen (of de bijbehorende <label>) verandert de snelheid in de snelheid die bij de gekozen knop hoort.
Toetsenbord: als één de radioknoppen focus heeft, kan met behulp van de pijltjestoetsen door de knoppen worden gelopen.

Sleepbalk afspelen

De sleepbalk voor het afspelen kan rechtstreeks worden bediend, maar ook met de knoppen Vijf procent vooruit of terug en Tien seconden vooruit of terug. Alle knoppen passen de sleepbalk aan. Dit regelt het script. De Speel-pauzeerknop heeft geen invloed op de sleepbalk.
Als de sleepbalk wordt veranderd, worden de getoonde resterende en verstreken speelduur tijdens het veranderen aangepast. Aan het eind van het slepen wordt de nieuwe verstreken speelduur door schermlezers voorgelezen. Ook dit wordt door het script geregeld.

De sleepbalk bestaat uit drie geneste 's, waarvan eigenlijk alleen de middelste belangrijk is voor de werking. De buitenste is alleen voor de css van belang, de binnenste is de knop en die staat er alleen maar voor de show knop te wezen: zonder knop werkt de sleepbalk precies hetzelfde.
De middelste is de eigenlijke sleepbalk. Aanraken, klikken, enz. werken alleen boven deze . Het is ook deze middelste die focus kan krijgen.

Muis: door op een bepaalde plaats op de sleepbalk te klikken, wordt de verstreken speelduur ingesteld afhankelijk van plaats, waar is geklikt. Als je op driekwart vanaf de linkerkant op de sleepbalk klikt, wordt de verstreken speelduur ingesteld op driekwart van de totale speelduur.
Als de knop van de muis ingedrukt blijft, kan de knop worden versleept.

Omdat je tijdens het slepen makkelijk onbedoeld buiten de sleepbalk kunt komen, wordt pas gestopt met slepen, als je de muis niet meer indrukt. Zolang de knop van de muis blijft ingedrukt, kun je teruggaan naar de sleepbalk en verdergaan met slepen.
Aanraken: werkt precies hetzelfde als de muis hierboven, maar dan met je vinger.
Toetsenbord: als de sleepbalk focus heeft, kan deze ook met het toetsenbord worden bediend. Dit gebeurt met behulp van de pijltjestoetsen, PgUp, PgDn, End en Home.
← of ↓: 5% terug;
→ of ↑: 5% vooruit;
PgDn: 10 seconden terug.
PgUp: 10 seconden vooruit;
End: naar einde video;
Home: naar begin video;
Deze toetsen worden vaak, in combinatie met bijzondere toetsen als de Alt-toets, als sneltoets voor andere dingen gebruikt, zoals een pagina teruggaan in de browser. Als de balk focus heeft, zouden deze sneltoetsen niet meer werken. Daarom werken de genoemde toetsen alleen als de Alt-, Control-, Shift-, Windows-, Meta-, Command-, Super-, Option-, welke-fantasienaam-dan-ook-speciale-toets niet is ingedrukt. Met Alt+← (of Command+←) ga je dus ook ’n pagina terug, als de sleepbalk voor afspelen toevallig focus heeft.

Tijdens het veranderen van de sleepbalk, wordt de weergave van verstreken en resterende speelduur voortdurend aangepast. De eindstand is gelijk aan de nieuwe verstreken en resterende speelduur.
De nieuwe verstreken speelduur wordt door een schermlezer voorgelezen. Dit gebeurt pas, als de uiteindelijke verstreken speelduur is bereikt, dus niet tijdens slepen of zo, want dan zou je ’n hele serie nutteloze tussenliggende tijden horen.

De code aanpassen aan je eigen ontwerp

- Als je dit voorbeeld gaat aanpassen voor je eigen site, houdt het dan in eerste instantie zo eenvoudig mogelijk. Ga vooral geen details invullen.
- Gebruik vooral geen FrontPage, Publisher of Word (alle drie van Microsoft). Deze programma’s maken niet-standaard code die alleen goed te bekijken is in Internet Explorer. In alle andere browsers zie je grotendeels bagger, áls je al iets ziet.
  Publisher en Word zijn niet bedoeld om websites mee te maken. FrontPage is zwaar verouderd en wordt niet meer onderhouden door Microsoft. Als je beslist iets van Microsoft wilt gebruiken, schaf dan (voor ’n fikse prijs) een nieuwer programma aan, dat zich wel aan de standaarden houdt.
  Ook LibreOffice levert een uiterst beroerd soort html af. Tekstverwerkers met al hun toeters en bellen zijn gewoon niet geschikt om websites mee te bouwen.
  Je kunt natuurlijk ook een goed gratis programma gebruiken. Links naar dat soort programma’s vind je op de pagina met links onder Gereedschap → wysiwyg-editor.
  Maar het allerbeste is om gewoon zelf html, css, enz. te leren, omdat zelfs het allerbeste programma het nog steeds zwaar verliest van ’n op de juiste manier met de hand gemaakte pagina.
- Ik maak zelf het liefst een site in Firefox. Als je ’n site maakt in Firefox, Opera, Safari, Google Chrome of Internet Explorer 10 of later, is er ’n hele grote kans dat hij in alle browsers werkt. Ik geef de voorkeur aan Firefox, omdat er zoveel extensies (uitbreidingen) voor bestaan. De twee belangrijkste voor ’t maken van sites zijn Web Developer en Firebug, maar er bestaan nog tientallen andere. Verder is Firefox de enige grote browser die niet bij een bedrijf hoort dat vooral op je centen of data uit is.

Google Chrome wordt ook door veel mensen gebruikt, maar ik heb dus wat moeite met hoe Google je hele surfgedrag, schoenmaat en kleur van je onderbroek vastlegt, daarom gebruik ik zelf Google Chrome alleen om te testen.

- Het allereerste dat je moet invoeren, is het doctype, vóór welke andere code dan ook. Een lay-out met een missend of onvolledig doctype ziet er totaal anders uit dan een lay-out met een geldig doctype. Wát er anders is, verschilt ook nog 'ns tussen de diverse browsers. Als je klaar bent en dan nog 'ns 'n doctype gaat invoeren, weet je vrijwel zeker dat je van voren af aan kunt beginnen met de lay-out.
  Geldige doctypes vind je op www.w3.org/QA/2002/04/valid-dtd-list.
  Gebruik het volledige doctype, inclusief de url, anders werkt het niet goed.
- Gebruik een 'strict' doctype of (beter!) het doctype voor html5. Deze zijn bedoeld voor nieuwe sites. Het transitional doctype is bedoeld voor al bestaande sites, niet voor nieuwe. Het doctype voor html5 is uiterst simpel: `<!DOCTYPE html>`.
  Het transitional doctype staat talloze tags toe, die in html5 zijn verboden. Deze tags worden al zo'n tien jaar afgeraden. Het transitional doctype is echt alleen bedoeld om de puinhoop van vroeger, toen niet volgens standaarden werd gewerkt, enigszins te herstellen.
  Het strict doctype staat verouderde tags niet toe. Daardoor kan met 'n strict doctype, of het nu html of xhtml is, probleemloos worden overgestapt naar html5. Met een transitional doctype en het gebruik van afgekeurde tags kun je niet overstappen naar html5. Je moet dan eerst alle verouderde tags verwijderen, wat echt ontzettend veel werk kan zijn.
- Als tweede voer je de charset in. Het beste kun je utf-8 nemen. Als je later van charset verandert, loop je 'n grote kans dat je alle aparte tekens als letters met accenten weer opnieuw moet gaan invoeren. In html5 is het simpele `<meta charset="utf-8">` voldoende.
- Test vanaf het allereerste begin in zoveel mogelijk verschillende browsers in 'n aantal resoluties (schermgroottes). Onder het kopje Getest in kun je in deze uitleg vinden, waar ikzelf op test. Ook van Internet Explorer kun je meerdere versies naast elkaar draaien. Je kunt daarvoor zoeken op internet en op de pagina met links staan onder de kopjes Gereedschap → Meerdere versies van Internet Explorer draaien en Gereedschap → Weergave testen 'n aantal links die daar ook bij kunnen helpen. De compatibiliteitsweergave in Internet Explorer is niet geschikt om te testen, omdat deze enigszins verschilt van de weergave in échte browsers.
- Voor alle voorbeelden geldt: breng veranderingen stapsgewijs aan. Als je bijvoorbeeld foto's wilt laten weergeven, begin dan met het alleen veranderen van de namen van de foto's, zodat je eigen foto's worden weergegeven. Maakt niet uit als de maten niet kloppen en de teksten fout zijn. Als dat werkt, ga dan bijvoorbeeld de maten aanpassen. Dan de teksten. En controleer steeds, of alles nog goed werkt.
- Als het om een lay-out of iets dergelijks gaat: zorg eerst dat header, kolommen, footer, menu, en dergelijke staan en bewegen, zoals je wilt, en ga dan pas details binnen die blokken invullen. In eerste instantie gebruik je dus bijvoorbeeld 'n leeg blok op de plaats, waar uiteindelijk het menu komt te staan.
  Als je begint met allerlei details, is er 'n heel grote kans dat die de werking van de blokken gaan verstoren. Bouw eerst het huis, en ga dan pas de kamers inrichten. Zorg eerst dat de blokken werken, zoals je wilt. Dan zul je het daarna gelijk merken, als 'n toegevoegd detail als tekst of 'n afbeelding iets gaat storen. Daarvoor moet je natuurlijk wel regelmatig controleren in verschillende browsers, of alles nog wel goed werkt.
  Je kunt de blokken tijdens het aanpassen opvullen met bijvoorbeeld
1
2
3 enz., tot ze de juiste hoogte hebben. Het is handig om aan het einde even iets toe te voegen als 'laatste', zodat je zeker weet dat er niet ongemerkt drie regels onderaan naar 't virtuele walhalla zijn verhuisd.

Om de breedte te vullen, kun je het best 'n kort woord als 'huis' duizend keer of zo herhalen. Ook hier is het handig, om aan 't einde (en hier ook aan 't begin) 'n herkenningsteken te maken, zodat je zeker weet dat je de hele tekst ziet.

- Zolang je in grotere dingen zoals 'n lay-out aan 't wijzigen bent, zou ik je aanraden de verschillende delen een achtergrondkleur te geven. Je ziet dan goed, waar 'n deel precies staat. Een achtergrondkleur heeft – anders dan bijvoorbeeld een border – verder geen invloed op de lay-out, dus die is hier heel geschikt voor.
- Als je instellingen verandert in de style, verander er dan maar één, hooguit twee tegelijk. Als je er zeventien tegelijk verandert, is de kans groot dat je niet meer weet, wat je hebt gedaan. En dat je 't dus niet meer terug kunt draaien.
- Marges, padding en border worden bij de hoogte en breedte van de inhoud opgeteld. Hier worden vaak fouten mee gemaakt. Als je bijvoorbeeld in een lay-out 'n border toevoegt aan een van de 'hoofdvakken' (header, footer, kolommen), dan wordt deze er bij opgeteld. Bij 'n border van 2 px rondom de linkerkolom wordt deze dus plotseling 4 px breder (2 aan beide kanten), en 4 px hoger. Zoiets kan je hele lay-out verstoren, omdat iets net te breed of te hoog wordt. Je moet dan elders iets 4 px kleiner maken. Dat zal vaak zo zijn: als je één maat verandert, zul je vaak ook 'n andere moeten aanpassen.
  css geeft de mogelijkheid om marge, padding en border bínnen de breedte en hoogte van de inhoud te zetten met behulp van `box-sizing`, als je dat handiger vindt.
- In plaats van px kun je ook andere maten gebruiken, met name em. Voordeel van em is dat een lettergrootte, regelhoogte, e.d. in em ook in Internet Explorer kan worden veranderd. Nadeel is dat het de lay-out sneller kan verstoren dan bijvoorbeeld px. Dit moet je gewoon van geval tot geval bekijken. Voor weergave in mobiele apparaten zijn relatieve eenheden als em vrijwel altijd beter dan vaste eenheden als px.
  Zoomen kan trouwens altijd, ongeacht welke eenheid je gebruikt.
- Valideren, valideren, valideren en dan voor 't slapen gaan nog 'ns valideren.
  Valiwie???
  Valideren is het controleren van je html en css op 'n hele serie fouten. Computers zijn daar vaak veel beter in dan mensen. Als je 300 keer <h2> hebt gebruikt en 299 keer </h2> vindt 'n computer die ene missende </h2> zonder enig probleem. Jij ook wel, maar daarna ben je misschien wel aan vakantie toe.
  Je kunt je css en html zowel valideren, als 't online staat, als wanneer 't nog in je computer staat.
  html kun je valideren op: validator.w3.org/nu/.
  css kun je valideren op: jigsaw.w3.org/css-validator/.
  Valideren kan helpen om gekmakende fouten te vinden. Valide code garandeert ook dat de weergave in verschillende browsers (vrijwel) hetzelfde is. En valide code is over twintig jaar ook nog te bekijken.
  Valideren moet trouwens ook niet worden overdreven. Het is een hulpmiddel om echte fouten te vinden, meer niet. Het gaat erom dat je site goed werkt, niet dat je het braafste kind van de klas bent. Als de code niet valideert, maar daar is een goede reden voor, is daar niets op tegen.
  Op deze site is alle css en html gevalideerd. Als de code niet helemaal valide is (wat regelmatig voorkomt), staat daar onder Bekende problemen (en oplossingen) de reden van.

Toegankelijkheid en zoekmachines

Het eerste deel van deze tekst is voor alle voorbeelden hetzelfde. Eventueel specifiek voor dit voorbeeld geldende dingen staan verderop onder het kopje Specifiek voor dit voorbeeld.
Toegankelijkheid (in het Engels 'accessibility') is belangrijk voor bijvoorbeeld blinden die een schermlezer gebruiken, of voor motorisch gehandicapte mensen die moeite hebben met het bedienen van een muis. Een spider van een zoekmachine (dat is het programmaatje dat de site indexeert voor de zoekmachine) is te vergelijken met een blinde. Als je je site goed

toegankelijk maakt voor gehandicapten, is dat gelijk goed voor een hogere plaats in een zoekmachine. Dus als je 't niet uit sociale motieven wilt doen, kun je 't uit egoïstische motieven doen.
(Op die plaats in de zoekmachine heb je maar beperkt invloed. De toegankelijkheid van je site is maar één van de factoren, maar zeker niet onbelangrijk.)
Als je bij het maken van je site al rekening houdt met toegankelijkheid, is dat nauwelijks extra werk. 't Is ongeveer te vergelijken met inbraakbescherming: doe dat bij 'n nieuw huis en 't is nauwelijks extra werk, doe 't bij 'n bestaand huis en 't is al snel 'n enorme klus.
Enkele tips die helpen bij toegankelijkheid:

- Gebruik altijd een alt-beschrijving bij een afbeelding. De alt-tekst wordt gebruikt, als afbeeldingen niet kunnen worden getoond of gezien (dat geldt dus ook voor zoekmachines). Als je iets wilt laten zien, als je over de afbeelding hovert, gebruik daar dan het title-attribuut voor, niet de alt-beschrijving.
  Als een afbeelding alleen maar voor de sier wordt gebruikt, zet je daarbij `alt=""`, om aan te geven dat de afbeelding niet belangrijk is voor het begrijpen van de tekst of zo.
- Als uit de tekst bij een link niet duidelijk blijkt, waar de link naartoe leidt, gebruik dan een title bij de link. Een tekst als 'pagina met externe links' is waarschijnlijk duidelijk genoeg, een tekst als alleen 'links' mogelijk niet. Een duidelijke zwart-witregel is niet te geven, omdat dit ook van tekst e.d. in de omgeving van de link afhangt.
- Accesskeys (sneltoetsen) kun je beter niet gebruiken, deze geven te veel problemen omdat ze vaak dubbelop zijn met sneltoetsen voor de browser of andere al gebruikte sneltoetsen. Bovendien is voor de gebruiker meestal niet duidelijk, welke toetsen het zijn.
  Op zichzelf zijn accesskeys een heel goed idee. Maar helaas zijn ze ook in html5 volstrekt onvoldoende gedefinieerd. Er is nog steeds geen standaard voor de meest gebruikelijke accesskeys, zoals Zoek of Home.
  Er is nog steeds niet vastgelegd, hoe accesskeys zichtbaar gemaakt kunnen worden. Voor de makers van browsers zou dit 'n relatief kleine moeite zijn, voor de makers van 'n site is het bergen extra werk.
  Voor mij redenen om accesskeys (vrijwel) niet te gebruiken. Ik kan me wel voorstellen dat ze op sites die gericht zijn op 'n specifieke groep gebruikers nog enig nut kunnen hebben, maar voor algemene sites zou ik zeggen: niet gebruiken.
- Met behulp van de Tab-toets (of op 'n soortgelijke manier) kun je in de meeste browsers door links, invoervelden, e.d. lopen. Elke tab brengt je één link, invoerveld, e.d. verder, Shift+Tab één plaats terug. Met behulp van tabindex kun je de volgorde aangeven, waarin de Tab-toets werkt. Zonder tabindex wordt de volgorde van de html aangehouden bij gebruik van de Tab-toets, maar soms is een andere volgorde logischer.
  In principe is het beter als tabindex niet nodig is, maar gewoon de volgorde van de html wordt aangehouden. Bij verkeerd gebruik kan tabindex heel verwarrend zijn. Het is niet bedoeld om van de pagina een hindernisbaan voor kangoeroes te maken, waarop van beneden via links over rechts naar boven wordt gesprongen.
- In het verleden werd vaak aangeraden de volgorde van de code aan te passen. Een menu bijvoorbeeld kon in de html onderaan worden gezet, terwijl het op het scherm met behulp van css bovenaan werd gezet. Inmiddels zijn schermlezers e.d. zo sterk verbeterd dat dit niet meer wordt aangeraden. De volgorde in de html kan tegenwoordig beter hetzelfde zijn als die op het scherm, omdat het anders juist verwarrend kan werken.
  Een andere mogelijkheid is een zogenaamde skip-link: een link die je buiten het scherm parkeert met behulp van css, zodat hij normaal genomen niet te zien is. Zo'n

link is wel gewoon zichtbaar in speciale programma's zoals tekstbrowsers en schermlezers, want die kijken gewoon naar wat er in de broncode staat.
Zo'n link staat boven menu, header, e.d. en linkt naar de eigenlijke inhoud van de pagina, zodat mensen met één toetsaanslag naar de eigenlijke inhoud van de pagina kunnen gaan.
Een skip-link is ook nuttig voor gebruikers van de Tab-toets. Zodra de normaal genomen onzichtbare link door het indrukken van de Tab-toets focus krijgt, kun je hem op het scherm plaatsen, waardoor hij zichtbaar wordt. Bij een volgende tab wordt hij dan weer buiten het scherm geplaatst en is dus niet meer zichtbaar, zodat de lay-out niet wordt verstoord.

- Van oorsprong is html een taal om wetenschappelijke documenten weer te geven, pas later is hij gebruikt voor lay-out. Maar daar is hij dus eigenlijk nooit voor bedoeld geweest. Het gebruiken van html voor lay-out leidt tot enorme problemen voor gehandicapten en tot een lage plaats in zoekmachines.
  De html hoort alleen inhoud te bevatten, lay-out doe je met behulp van css. Die css moet in een externe stylesheet staan of, als hij alleen voor één bepaalde pagina van toepassing is, in de head van die pagina. Zoekmachines zijn ook niet dol op een oerwoud van inline-stijlen (dat zijn stijlen in de tag zelf: `<div style="...">`.)
- Breng een logische structuur aan in je document. Gebruik een <h1> voor de belangrijkste kop, een <h2> voor een subkop, enz. Schermlezers e.d. kunnen van kop naar kop springen. En een zoekmachine gaat ervan uit dat <h1> belangrijke tekst bevat.
  Dit geldt voor al dit soort structuurbepalende tags.
  Als een <h1> te grote letters geeft, maak daar dan met behulp van je css 'n kleinere letter van, maar blijf die <h1> gewoon gebruiken. Op dezelfde manier kun je al dit soort dingen oplossen.
- <table> is fantastisch, maar alleen als die wordt gebruikt om een echte tabel weer te geven, niet als hij voor opmaak wordt misbruikt. In het verleden is dat op grote schaal gebeurd bij gebrek aan andere mogelijkheden. Een tabel is, als je niet heel erg goed oplet, volstrekt ontoegankelijk voor gehandicapten en zoekmachines. Het lezen van een tabel is ongeveer te vergelijken met het lezen van een krant van links naar rechts: niet per kolom, maar per regel. Dat gaat dus alleen maar goed bij een echte tabel zoals een spreadsheet. In alle andere gevallen garandeert 'n tabel 'n lagere plaats in een zoekmachine.
- Frames zijn een volstrekt verouderde techniek, die heel veel nadelen met zich meebrengt. Deze zijn hier netjes op een rijtje gezet: www.webrichtlijnen.nl/aan-de-slag/alles-over-frames. <iframe>'s hebben voor een deel dezelfde nadelen.
  Als je 'n stuk code vaak wilt gebruiken, zoals 'n menu dat op elke pagina hetzelfde is, voeg dat dan in met PHP of SSI (Server Side Includes). Dan ziet iedereen (ook 'n zoekmachines dus!) alles als één pagina in plaats van als los zand aan elkaar hangende teksten.
- Geef de taal van het document aan, en bij woorden en dergelijke die afwijken van die taal de afwijkende taal met behulp van `lang=".."`. Ik doe dat op mijn eigen site maar af en toe, omdat de tekst (en vooral de code) een mengsel is van Engels, Nederlands en eigengemaakte namen. Dit soort teksten is gewoon niet goed in te delen in een taal. Maar bij enigszins 'normale' teksten hoor je een taalwisseling aan te geven.
- Gebruik de tag <abbr> bij afkortingen. Doe dat de eerste keer op een pagina samen met de title-eigenschap: `<abbr title="ten opzichte van">t.o.v.</abbr>`. Daarna kun je op dezelfde pagina volstaan met `<abbr>t.o.v.</abbr>`. Doe je dit niet, dan is er 'n grote kans dat 'n

schermlezer ’t.o.v.’ uit gaat spreken als ’tof’, en ’n zoekmachine kan er ook geen chocola van maken.

- De spider van ’n zoekmachine, schermlezers, en dergelijke kunnen geen plaatjes ’lezen’. Het is soms verbazingwekkend om te zien hoe veel, of eigenlijk: hoe weinig tekst er overblijft op een pagina, als de plaatjes worden weggehaald. Het zelfde geldt voor die fantastisch mooie flash-pagina’s, als daarbij geen voorzieningen voor dit soort programma’s zijn aangebracht.
  Op Linux kun je met Lynx kijken, hoe je pagina eruitziet zonder plaatjes e.d., als echt alleen de tekst overblijft. Een installatie-programma voor Lynx op Windows is te vinden op invisible-island.net/lynx.
  Ook kun je in Windows het gratis programma WebbIE installeren. WebbIE laat de pagina zien, zoals een tekstbrowser e.d. hem zien. WebbIE is te downloaden vanaf www.webbie.org.uk.
  De Firefox-extensie Web Developer heeft de mogelijkheid om ’n pagina te bekijken zonder css en/of afbeeldingen.
  Tenslotte kun je je pagina nog online laten controleren op ’n behoorlijk aantal sites. Ik noem er hier enkele.
  colorfilter.wickline.org/
  Laat zien hoe een kleurenblinde de site ziet. Engelstalig.
  wave.webaim.org/
  Deze laat grafisch zien hoe de toegankelijkheid is. Engelstalig.
  Op de pagina met links kun je onder Toegankelijkheid links naar meer tests e.d. vinden.

Specifiek voor dit voorbeeld

- Hieronder staat, hoe het in een ideale wereld allemaal zou moeten werken. Nogal wat schermlezers werken niet helemaal goed, waardoor niet alles altijd volledig zal werken, zoals hieronder staat beschreven. Zo wordt bijvoorbeeld in VoiceOver toch een andere geluidssterkte voorgelezen, als de video speelt (wat alleen zou mogen gebeuren bij 0% en 100%).
- Zonder JavaScript wordt gewoon de in de browser ingebouwde standaardvideospeler gebruikt. (Deze werkt vaak ook niet zonder JavaScript, maar dat heeft niets met dit voorbeeld te maken.)
  Er is een mogelijkheid om in kleinere browservensters (in dit voorbeeld smaller of lager dan 480 px, maar dat kan worden aangepast) kleinere video’s te gebruiken. Dat werkt alleen, als JavaScript aan staat. Die kleinere vensters kom je alleen in mobieltjes tegen. De kans dat JavaScript uit staat op een mobieltje is nihil, omdat je dan niets meer hebt dan een groot uitgevallen horloge. En zelfs als JavaScript uit zou staan, dan werkt alles nog, alleen krijg je dan de grotere video.
- Zonder css werkt de hele handel niet. Dat wil zeggen: de videospeler is wel aanwezig, maar het uiterlijk van de speler niet. Bij Achterliggend idee staat iets onder het begin een afbeelding van een speler zonder css. Alles werkt en zo, maar de bediening is ’n pietsie onduidelijk.
- Zonder afbeeldingen is niet echt grondig getest, omdat het ietwat onwaarschijnlijk lijkt dat iemand die geen afbeeldingen wil zien, wel video’s gaat bekijken... In principe zou alles moeten werken. Maar in de videospelers waar voor de symbolen van de bedieningselementen achtergrond-afbeeldingen worden gebruikt, zijn alleen lege knoppen te zien. Die wel werken, dat wel, maar het is wat lastig.
- In dit voorbeeld wordt fors gebruik gemaakt van WAI-ARIA-codes. Daarom is daar een eigen hoofdstukje aan gewijd: Semantische elementen en WAI-ARIA. Hetzelfde geldt voor tabindex, waarover meer staat bij Tabindex.
- Een schermlezer (maar ook een zoekmachine) weet niets van de inhoud van de video, als die niet in de tekst wordt omschreven. Daarom moet in het <video>-

element met behulp van `aria-label`, `aria-labelledby` of `aria-describedby` worden verwezen naar een omschrijving van de video. Als die omschrijving mist, wordt automatisch door het script een simpele verwijzing naar de bijbehorende video ingevoegd. Meer daarover bij aria-describedby.

- De volgorde waarin het script de bedieningselementen invoegt, kan niet worden gewijzigd. Je kunt de bedieningselementen wel met behulp van css op allerlei verschillende plaatsen neerzetten, en daarmee dus de volgorde op het scherm veranderen.
  Maar een schermlezer zal altijd de volgorde van de html aanhouden, niet de volgorde zoals die op het scherm staat. Als je de volgorde op het scherm heel erg laat afwijken van die in de html, kan dat heel verwarrend zijn.
  De volgorde waarin het script de bedieningselementen invoegt, is het duidelijkst te zien op de tekening bij Overzicht van door het script ingevoegde bedienings-elementen, classes en id's.
- Als de geluidssterkte wordt veranderd, wordt in schermlezers de nieuwe geluidssterkte in procenten voorgelezen. De tekst hiervan kan worden aangepast, zoals beschreven bij Text. De nieuwe geluidssterkte wordt niet voorgelezen, als de video speelt, omdat dit storend kan zijn en je de nieuwe geluidssterkte toch wel hoort. Maar voor het geval de video geen geluid heeft, worden 0% en 100% altijd voorgelezen.
  Om het voorlezen van de geluidssterkte volledig uit te schakelen, kun je volstaan met `display: none;` bij de betreffende :

  ```
  span.aria-volume {display: none;}
  ```
- Als de video voor- of achteruit wordt gezet met behulp van knoppen of sleepbalk, wordt in schermlezers de nieuwe verstreken speelduur voorgelezen in uur, minuut en seconden. De tekst hiervan kan worden aangepast, zoals beschreven bij Text.
  Om het voorlezen van de tijd uit te schakelen, kun je volstaan met `display: none;` bij de betreffende :

  ```
  span.aria-elapsed {display: none;}
  ```
- Als begin of einde van de video is bereikt, wordt de tekst 'Begin video' of 'Einde video' voorgelezen. De tekst hiervan kan worden aangepast, zoals beschreven bij Text.
- Op de vijf pagina's met voorbeelden is boven de links naar vorige en volgende pagina een <h2> neergezet, die aangeeft om wat voor soort links het hier gaat. Deze <h2> wordt door een schermlezer voorgelezen, maar is buiten het scherm geparkeerd, zodat je hem niet ziet en de lay-out niet wordt verstoord.
- Er zijn vier mogelijke situaties, waarin een melding kan worden getoond, zoals beschreven bij Door het script gegenereerde waarschuwingen (alerts, pop-ups). Drie hiervan zijn gewone JavaScripts-alerts, die alleen kunnen verschijnen bij een verkeerd gebruik van het script. Die alerts zijn, vriendelijk uitgedrukt, niet echt mooi. Maar omdat die alleen verschijnen bij het maken van een pagina, heb ik dat zo gelaten. Sluiten van een alert zal geen problemen opleveren wat betreft toegankelijkheid.
  De vierde melding kan ook door bezoekers worden gezien. Hoe het openen en sluiten daarvan zo toegankelijk mogelijk is gemaakt, is te vinden bij Openen en sluiten van de melding.
- Ondertiteling en captions hebben niets met dit script te maken, dat moet in de html worden geregeld. Op de pagina met links staat meer daarover onder HTML → <embed>, <object>, <video>, <audio>, <iframe> → Ondertiteling en bijschriften (<track>).

- De meeste schermlezers lezen tekst uit pseudo-elementen die met behulp van `::after` en `::before` gewoon voor. Hierdoor kan een schermlezer op de eerste en vijfde pagina met videospelers, afhankelijk van de instellingen, een knop twee keer voorlezen.
  Op de tweede pagina wordt een webfont met symbolen gebruikt. Deze symbolen zitten in een zogenaamd 'private area': een gebied wat vrij is om te gebruiken voor wat je maar wilt. In principe worden deze symbolen niet voorgelezen, maar er zijn verhalen dat dit toch zou gebeuren, wat weer door anderen wordt ontkend. Bij het testen werden bij mij de symbolen niet voorgelezen.
  Op de vierde pagina zijn achtergrond-afbeeldingen gebruikt, deze worden door schermlezers genegeerd.
  Op de vijfde pagina zijn data-uri's gebruikt. Dat zijn in feite ook afbeeldingen, dus die worden door schermlezers ook genegeerd.
- Met sneltoetsen ← en → wordt naar vorige of volgende video gegaan (afhankelijk van de browser in combinatie met Control of Control+Alt).
  Sneltoets 8 opent of sluit het venstertje met uitleg (werkt niet in Internet Explorer).
  Op de vierde pagina kan de besturing van de tweede video worden getoond en verborgen met sneltoets 7. Beide laatste sneltoetsen werken, afhankelijk van de browser, in combinatie met Alt, Alt+Shift of Alt+Control.
- Als VoiceOver, de schermlezer op iOS en OS X, aan staat, kan de snelheid niet worden veranderd, omdat aan- en uitvinken van de radioknoppen voor snelheid niet werkt.
  In VoiceOver werken de sleepbalken voor geluidssterkte en afspelen niet, ook niet als je ze op 'n bepaalde plaats aanraakt.
- In WebbIE, een browser voor blinden en slechtzienden, kun je geen video's afspelen. Omdat WebbIE moeite heeft met JavaScript, zijn geen videospelers aanwezig. Downloaden van de video lukt wel. Als wordt gekozen voor weergave als webpage, werkt alles goed. Maar dat is niet zo vreemd, want dan wordt in feite Internet Explorer gebruikt.
- De schermlezer NVDA leest alle knoppen e.d. correct voor. Afhankelijk van de instellingen worden sommige knoppen twee keer voorgelezen.
- TalkBack, de schermlezer op Android werkt goed, alleen kun je de sleepbalken niet slepen. Bij aanraken van de sleepbalk verspringt de geluidssterkte of de verstreken speelduur wel naar de bijbehorende waarde.

Getest in

Laatst gecontroleerd op 17 april 2015.
Onder dit kopje staat alleen maar, hoe en waarin is getest. Eventuele problemen, ook die met betrekking tot zoomen en lettergroottes, staan hieronder bij Bekende problemen (en oplossingen). Het is belangrijk dat deel te lezen, want uit een test kan ook prima blijken dat iets totaal niet werkt!
Eventuele opmerkingen over de toegankelijkheid specifiek voor dit voorbeeld staan onderaan Toegankelijkheid en zoekmachines onder het kopje Specifiek voor dit voorbeeld.
Dit voorbeeld is getest op de volgende systemen:

- Windows 7:
  Firefox, Opera, Google Chrome, Internet Explorer 9 en 10 in de resoluties 800x600, 1024x768 en 1280x1024.
- Windows 8:
  Bureaublad-versie: Firefox, Opera, Google Chrome en Internet Explorer 11 in de resoluties 800x600, 1024x768 en 1366x768.
  Startscherm-versie: Internet Explorer 11 (heette ooit Metro-versie).
  Windows-8-modus: Google Chrome (heette ooit Metro-versie).

- OS X:
  Firefox, Safari, Opera en Google Chrome in een resolutie van 1680x1050 en kleiner. Kleiner is wat vaag: het browservenster is traploos verkleind van 1680x1050 tot ongeveer 1000x600. Waarbij alle tussenliggende groottes dus ook zijn getest.
- Linux:
  Firefox, Opera en Google Chrome in de resoluties 800x600, 1024x768 en 1280x1024.
- Windows Phone 8.1 in een resolutie van 800x480 (Nokia Lumia 520):
  Internet Explorer (portret en landschap).
- iPad met iOS 8.3 in een resolutie van 1024x768 (MC979NF):
  Safari (portret en landschap).
  Opera Mini (portret en landschap). Meer over deze browser hieronder bij Android 4.4.2
  Chrome for iOS (portret en landschap).
- Android 4.0.3 in een resolutie van 1024x768 (CRESTA CTP888):
  Android browser (portret en landschap).
- Android 4.1 in een resolutie van 800x480 (Samsung Galaxy Core i8620):
  Opera, Chrome, Android browser en Firefox (alle portret en landschap).
  Opera Mini (éénkolomsstand aan/uit, portret en landschap). Meer over deze browser hieronder bij Android 4.4.2.
- Android 4.4.2 in een resolutie van 1280x800 (Samsung Tab 3):
  Android browser, Opera, Firefox en Chrome (alle portret en landschap).
  Opera Mini (éénkolomsstand aan/uit, portret en landschap). Deze browser is een apart geval. De opgevraagde site wordt gedownload via de servers van Opera, waarbij het in zo'n klein browservenster normaal is dat (een groot deel van) de lay-out wordt verwijderd. Daarom wordt bij deze browser voornamelijk gekeken, of er geen content (tekst e.d.) verloren gaat.

Er is steeds getest met de laatste versie van de browsers op de aan het begin van dit hoofdstukje genoemde controledatum, omdat ik geen zin heb om rekening te houden met mensen die met zwaar verouderde browsers surfen. Dat is trouwens vragen om ellende, want updates van browsers hebben heel vaak met beveiligingsproblemen te maken.
Ook op Windows XP kunnen mensen surfen met Firefox, Opera of Google Chrome, dus ook daar zijn mensen niet afhankelijk van Internet Explorer 8. Ik maak één uitzondering: Android browser op Android 4.0.3. Omdat Android vaak niet geüpdatet kan worden, test ik ook nog in Android browser 4.0.3.
In resoluties groter dan 800x600 is ook in- en uitzoomen en – voor zover de browser dat kan – een kleinere en grotere letter getest. Er is ingezoomd en vergroot tot zover de browser kan, maar niet verder dan tot 200%.
Er is getest met behulp van muis en toetsenbord, behalve op de iPad, Android en Windows Phone, waar een touchscreen is gebruikt. Op Windows 8 is getest met een touchscreen, met een combinatie van toetsenbord en touchpad, en met een combinatie van toetsenbord en muis.

Op de desktop is ook getest, als JavaScript uitstaat. Eventuele problemen staan onderaan Toegankelijkheid en zoekmachines onder het kopje Specifiek voor dit voorbeeld. (Op iOS, Android en Windows Phone is niet getest zonder JavaScript, omdat je JavaScript in een toenemend aantal mobiele browsers niet uit kunt zetten. Bovendien is een mobiel apparaat zonder JavaScript niet veel meer dan een duur en groot uitgevallen horloge.)

Naast deze 'gewone' browsers is ook getest in Lynx, WebbIE, NVDA, TalkBack op Android en VoiceOver op iOS en OS X. Lynx is een browser die alleen tekst laat zien en geen css gebruikt. WebbIE is een browser die gericht is op mensen met een handicap. NVDA is een

schermlezer, zoals die door blinden wordt gebruikt. (NVDA is getest in combinatie met Firefox op Windows 7.) TalkBack is een in Android ingebouwde schermlezer (TalkBack is getest in combinatie met Chrome). VoiceOver is een in iOS en OS X ingebouwde schermlezer (VoiceOver is getest in combinatie met Safari).
Als het voorbeeld in deze programma's toegankelijk is, zou het in principe toegankelijk moeten zijn in alle aangepaste browsers en dergelijke. En dus ook voor zoekmachines, want een zoekmachine is redelijk vergelijkbaar met een blinde. Eventuele opmerkingen over de toegankelijkheid specifiek voor dit voorbeeld staan onderaan Toegankelijkheid en zoekmachines onder het kopje Specifiek voor dit voorbeeld.

Alleen op de hierboven genoemde systemen en browsers is getest. Er is dus niet getest op bijvoorbeeld 'n Blackberry. De kans is (heel erg) groot dat dit voorbeeld niet (volledig) werkt op niet-geteste systemen en apparaten. Om het wel (volledig) werkend te krijgen, zul je vaak (kleine) wijzigingen en/of (kleine) aanvullingen moeten aanbrengen, bijvoorbeeld met JavaScript.
Er is ook geen enkele garantie dat iets werkt in een andere tablet of smartphone dan hierboven genoemd, omdat fabrikanten in principe de software kunnen veranderen. Dit is anders dan op de desktop, waar browsers altijd (vrijwel) hetzelfde werken, zelfs op verschillende besturingssystemen. Iets wat in Android browser werkt, zal in de regel overal werken in die browser, maar een garantie is er niet. De enige garantie is het daadwerkelijk testen op een fysiek apparaat. En aangezien er duizenden mobiele apparaten zijn, is daar eigenlijk geen beginnen aan.

De html is gevalideerd met de validator van w3c, de css ook. Als om een of andere reden niet volledig gevalideerd kon worden, wordt dat bij Bekende problemen (en oplossingen) vermeld.

Nieuwe browsers test ik pas, als ze uit het bèta-stadium zijn, omdat er anders 'n redelijke kans is dat ik 'n bug zit te omzeilen, die voor de uiteindelijke versie nog gerepareerd wordt. Dit voorbeeld is alleen getest in de hierboven met name genoemde browsers. Vragen over niet-geteste browsers kan ik niet beantwoorden, en het melden van fouten in niet-geteste browsers heeft ook geen enkel nut. (Melden van fouten, problemen, enz. in wel geteste browsers: graag!)

Bekende problemen (en oplossingen)

Waarop en hoe is getest, kun je gelijk hierboven vinden bij Getest in.
Als je hieronder geen oplossing vindt voor een probleem dat met dit voorbeeld te maken heeft, kun je op het forum proberen een oplossing te vinden voor je probleem. Om forumspam te voorkomen, moet je je helaas wel registreren, voordat je op het forum een probleem kunt aankaarten.

Onderstaande lijst is nogal lang, maar veel problemen zijn tamelijk onbelangrijk. Veel zitten ook ingebakken in een bepaalde browser of besturingssysteem en hebben dus weinig met dit voorbeeld te maken. Een groot deel van de hieronder genoemde dingen heeft te maken met css, zoals problemen met het relatief nieuwe `@keyframes`, en dus niet met de kern van dit voorbeeld.
Als een probleem zich in álle browsers voordoet, ligt het voor de hand te denken aan een probleem met de code. Doet het zich slechts in één specifieke browser voor, dan ligt het mogelijk eerder aan die specifieke browser.

Alle browsers

Er verschijnt een JavaScript-waarschuwingsvenster

Het gaat mogelijk om een door het script gegeneerde waarschuwing. Deze worden besproken bij Door het script gegenereerde waarschuwingen (alerts, pop-ups).

De video speelt niet af

Een browser moet het zogenaamde MIME-type van een bestand weten, om het als een video te herkennen. De ene browser is daar wat kritischer in dan de andere.
Het MIME-type moet op de server worden geconfigureerd. Als dit niet goed is gedaan, kan het zijn dat de video aan de browser wordt gemeld als een gewoon tekstbestand. Firefox bijvoorbeeld speelt in dat geval WebM niet af, en Internet Explorer 11 doet niets met MP4.
Als je alles hebt geïnstalleerd en zo en het werkt niet, kun je controleren of de MIME-types wel goed zijn ingesteld. Hieronder beschrijf ik, hoe je dat in Firefox en Internet Explorer doet, omdat ik toevallig die twee heb gebruikt. Maar ook Opera, Safari en Google Chrome hebben dit soort hulpmiddelen.

WebM kun je in Firefox controleren met behulp van de extensie Firebug.
Start Firebug en klik in de menubalk bovenaan op 'Net'. Klik in de nu verschijnende tweede balk op 'Alles'. Herlaadt de pagina met de video. In de linkerkolom zie je nu de namen van de bestanden verschijnen, die zijn gedownload.
Start één van de video's. Nu weet je zeker dat deze is opgevraagd. (Of dat opvragen gebeurt vanaf internet of vanuit de cache, een soort tijdelijke opslag op de computer, maakt niet uit.)
Zoek de naam van de afgespeelde video op in de linkerkolom en klik op het plusje voor de naam. Als er voor 'Antwoordheaders' een plusje staat, klik daar dan op. Er opent een lijstje met enge woorden. Als je 'n beetje omlaag scrolt, zie je daar 'Content-Type' staan. Hier moet 'video/webm' achter staan. Als er iets anders staat, zoals 'text/plain', klopt het MIME-type niet.
(Inmiddels kan dit ook in de in Firefox ingebouwde ontwikkelgereedschappen: Extra → Webontwikkelaar → Netwerk. Onderaan de pagina opent een venstertje. Herlaad de pagina. Start een video, zodat je zeker weet dat deze echt is gedownload. Zoek in de kolom Bestand naar de video. In de kolom Type moet 'webm' staan. Als er 'plain' staat, is het MIME-type niet goed ingesteld. Overigens lijkt Firefox webm inmiddels altijd te herkennen, maar daar kun je niet blind op vertrouwen.)

MP4 kun je in Internet Explorer controleren. Microsoft vindt het kennelijk geestig om bij elke nieuwe versie van Internet Explorer een nieuwe puzzel te bedenken, want in elke versie werkt het ontwikkelgereedschap weer anders. In Internet Explorer 11 werkt het als volgt: open de pagina met de video. Druk op F12 om het ontwikkelgereedschap te openen. Klik in de linkerbalk op het icoontje voor Netwerk (welk icoontje dat is, moet je even paranormaal vaststellen). Klik in de balk bovenin op het groene pijltje. Herlaadt de pagina. In de linkerkolom verschijnen de namen van de bestanden die zijn gedownload.
Start een video, zodat je zeker weet dat die video echt is gedownload. Zoek de naam van de afgespeelde video op in de linkerkolom. In de vierde kolom,

onder ’Type’, moet staan ’video/mp4’. Als er iets anders staat, zoals ’text/plain’, klopt het MIME-type niet.

Op Apache, de meest gebruikte server voor websites, kan dit worden ingesteld in een zogenaamd .htaccess-bestand. Ik heb er geen flauw idee van, hoe het op een Windows server moet. Mijn hartelijke deelneming en veel sterkte gewenst dus als je met ’n Windows-server zit opgescheept. Je hoster zou je moeten kunnen helpen.
Niet alle hosters laten je zelf een .htaccess-bestand maken. Als dat bij jouw hoster niet kan, moet de hoster deze verandering even zelf aanbrengen. Elke fatsoenlijke hoster doet dit (en gaat zich daarna ’n halfuurtje zitten schamen onder het bureau, omdat ze dit kennelijk vergeten waren).
Als je zelf een .htaccess kunt maken of bewerken, des te beter, want je kunt nog veel meer met ’n .htaccess.
Aan een bestaande .htaccess kunnen gewoon onderstaande regels worden toegevoegd. Als je nog geen .htaccess hebt, maak je ’n nieuwe aan. Een .htaccess-bestand is een gewoon tekstbestand. Je moet het dus maken in een zelfde soort simpele editor, als waarin je html of css schrijft. Programma’s als Word of LibreOffice stoppen er allemaal extra meuk in voor de opmaak, en daar wordt de server hevig ongelukkig van.
Het bestand krijgt de naam ’.htaccess’. Verder niets. De punt aan het begin zorgt ervoor dat het bestand op Linux onzichtbaar is. Als de .htaccess na het uploaden onzichtbaar is op de server, moet je waarschijnlijk even in je FTP-programma (het programma waarmee je de site uploadt) aangeven dat onzichtbare bestanden ook getoond moeten worden.
Het .htaccess-bestand wordt in de root van de site gezet. Dat is op dezelfde plaats, als waar je index.html of index.php staat. In het .htaccess-bestand zet je de volgende regels:

```
AddType video/webm .webm
AddType video/mp4 .mp4
```

Dat is alles. Als het goed is, zou je nu de juiste MIME-types moeten zien in Firefox en Internet Explorer 11, zodat je in ieder geval zeker weet dat een eventueel probleem niet hieraan kan liggen.

Een andere mogelijkheid is dat de browser de gebruikte codec niet kent. WebM, MP4, enz. zijn een zogenaamde container, net zoals een zip-bestand dat is. Binnen een zip kunnen allerlei verschillende formaten zitten, zoals gewone tekst, html, afbeeldingen, enz.
WebM, MP4, enz. zijn enigszins te vergelijken met een zip. Binnen WebM, MP4, enz. kunnen verschillende codecs zijn gebruikt, en niet elke browser kent elke codec. Bij de in dit voorbeeld gebruikte video’s zijn geen codecs opgegeven, omdat elke browser ze probleemloos kan afspelen.
Iets meer info over codecs kun je vinden op WHATWG Living Standard onder punt 4.8.12 The source element, en op wikipedia.org/wiki/Video_codec.
Omdat ik verder zelf ook geen verstand heb van al die verschillende codecs en zo, kan ik er verder niets zinnigs over melden. Als je mocht denken dat het probleem hiermee te maken zou kunnen hebben, kun je op de pagina met links onder Forums hopelijk een plaats vinden, waar je meer hulp kunt krijgen.

Er verschijnen helemaal geen aangepaste videospelers, of sommige knoppen e.d. doen het niet goed

Voldoen html en css aan de Voorwaarden waaraan de html en css moeten voldoen? Als bijvoorbeeld bepaalde classes missen, zal het script niet of onvolledig werken.

Het is onmogelijk met de Tab-toets van knop naar knop, link e.d. te springen

Mogelijk staat dit niet ingeschakeld in de instellingen van de browser of van het besturingssysteem. In Opera bijvoorbeeld staat deze handigheid om onbegrijpelijke redenen standaard uit. (Inschakelen in Opera: klik bovenin op Opera, → Instellingen → Websites en vink daar aan 'Navigeren op een webpagina met Tab gaat langs alle links, evenals langs formuliervelden'.)
In OS X staat dit ook standaard uitgeschakeld. Ook al schakel je het in de browser in, dan nog zal de Tab-toets niet goed werken in sommige browsers als de instellingen in OS X zelf niet worden gewijzigd. (Mogelijk is dit in de nieuwste versie van OS X niet meer nodig. Maar omdat Apple die instellingen steeds op andere plaatsen verstopt, zul je even zelf moeten zoeken. In ieder geval staat het nog steeds uit in Safari. Inschakelen in Safari: Safari → Voorkeuren → Geavanceerd en vink daar aan 'Tab-toets markeert elk onderdeel op webpagina'.)

Als in de voorkeuren is opgegeven dat de Tab-toets alle links e.d. moet bezoeken en dat gebeurt niet, kijk dan in de gegenereerde code. Als je gewoon de code van de pagina bekijkt, zie je de door het script toegevoegde elementen niet. Je kunt nu zien of in de bedieningselementen van de videospeler wel een tabindex aanwezig is. Als die er niet is, heb je vermoedelijk bovenin het script geen tabindex ingevuld. Hoe je dat kunt doen, staat bij TabIndex.

De Tab-toets volgt niet de juiste volgorde bij springen van knop naar knop, link, e.d.

Dan klopt de nummering van de tabindexen niet. Je kunt de gegenereerde code bekijken en daarin de nummering van de tabindexen controleren.
De tabindexen van de bedieningselementen van de videospelers en van de radioknoppen voor het kiezen van de soort videospeler worden aangebracht door het script. De volgnummers van die tabindexen moeten (uiteraard) goed aansluiten bij de volgnummers van de tabindexen die je zelf hebt aangebracht in de html.
De door het script aangebrachte tabindexen kun je wijzigen. Hoe je dat kunt doen, staat bij TabIndex. (Het is trouwens beter om de volgorde van de knoppen e.d. aan te houden, zoals die worden ingevoegd, want dan heb je helemaal geen tabindexen nodig.)

Na klikken op een sleepbalk stopt het slepen niet meer

Windows 7 en 8, en mogelijk ook andere systemen, blijken een klikvergrendeling te hebben, te vinden bij de instellingen voor de muis. Door deze in te schakelen blijft de linkermuisknop ingedrukt, ook als je die loslaat. Uiterst handig. Als je het weet. Ik kende die instelling niet, wat me in Windows 8 ongeveer 'n week gekost, omdat ik met 'n voor mij volstrekt onbegrijpelijke spookmuis met 'n geheel eigen wil te maken leek te hebben. Als dit gebeurt, controleer die instellingen dan even, wil ik maar zeggen...

De sleepbalken voor geluid geven nooit 99% of 1% aan

Complimenten voor je opmerkingsgave. En je hebt het goed gezien, dit is expres zo gedaan. Als je de knoppen voor Harder en Zachter gebruikt, wordt gewoon 99% en 1% gebruikt. Maar bij de sleepbalken voor het geluid wordt het percentage berekend afhankelijk van waar de knop van de sleepbalk staat, waarna het percentage wordt afgerond. Door dit afronden kan het percentage uitkomen op -1% of 101%. Om dat te voorkomen, wordt alles groter dan 98% afgerond op 100%, en alles kleiner dan 2% op 0%.

Uit een door mij uitgevoerd representatief onderzoek bleek dat slecht 0,007% van de ondervraagden de 1% en 99% echt misten. En die ondervraagden waren ook nog door overvloedig alcoholmisbruik in een weemoedige stemming: ze riepen ook om hun mama. Dus ik denk dat dit niet echt een enorm gemis is.

De knop van de sleepbalk voor afspelen verschuift sprongsgewijs

Dat is helemaal geen probleem, dat is expres zo gedaan. De positie van de knop wordt berekend aan de hand van de verstreken en de totale speelduur. Die tijd wordt afgerond op hele seconden. Bij de totale, de resterende en de verstreken speelduur wordt de tijd ook in hele seconden weergegeven, en niet in tienden of honderdsten van seconden.

De plaats van de knop op de sleepbalk voor afspelen correspondeert met die tijden. De video's in het midden en onderaan duren elk maar drie seconden (in browservensters breder en hoger dan 480 px). Dus heeft elke knop in de sleepbalk voor afspelen maar vier standen: 0, 1, 2 en 3 seconden.

Bij de tweede video, die 34 seconden duurt, zijn er 35 standen en is de knop al veel minder springerig. Hoe meer seconden de video duurt, hoe meer standen de knop heeft. Bij de eerste videospeler, waar de video ruim drie minuten duurt, zie je het verspringen (vrijwel) niet meer.

Bij het slepen van de knop zie je wel de beeldjes van de video verspringen, ook tussen de hele seconden in. Dat komt omdat je tussen de hele seconden in zittende beeldjes gewoon ziet. Maar de knop verspringt pas naar de volgende stand als wordt afgerond naar de volgende hele seconde.

De video springt 9 of 11 seconden voor- of achteruit, in plaats van 10

Bij gebruik van de knoppen voor Tien seconden voor- of achteruit, verspringt de video af en toe geen tien, maar negen of elf seconden.

Dat komt, doordat de weergegeven wordt afgerond op hele seconden. Heel af en toe wordt hierdoor één seconde te veel of te weinig weergegeven. Tenzij je toevallig tijdmeter bent bij wielrennen op de Olympische Spelen, lijkt dit me niet echt een probleem.

De speelduur van de video wordt pas gevonden, als de video wordt gestart

Dit kan een aantal oorzaken hebben.

Voor Chrome for iOS en Safari op iOS staat dit beschreven bij Safari op iOS. Voor andere browsers kan het tal van oorzaken hebben. Op de voorbeeldpagina's worden zes video's opgehaald. Als de server of de computer wat traag is, kan het voorkomen dat niet alle metadata (dingen als speelduur) van de video worden opgehaald. Ook heeft de ene browser er meer moeite mee dan de andere. Bij een browser als Internet Explorer 9 barst ik – tot diep verdriet van mijn buren – al uit in vreugdegezang, als die twee van de zes weet binnen te halen.

Eén seconde na het laden van de pagina wordt nogmaals de speelduur e.d. opgevraagd. De meeste browsers lukt het dan deze alsnog op te halen, als dat nog niet was gelukt.
Zolang de speelduur nog onbekend is, wordt als speelduur 00:00 ingevuld.

Bij zoomen werken de sleepbalken soms niet goed

Dit probleem speelt niet overal in even grote mate. Het speelt alleen bij zoomen, niet bij een andere lettergrootte of als de video fullscreen wordt bekeken. Het doet zich alleen voor, als er wordt gezoomd, terwijl de pagina al open is. Als het beeld al is vergroot (of verkleind) terwijl de pagina wordt geopend, is er geen enkel probleem. De knoppen voor de bediening van beeld en geluid werken altijd goed bij zoomen.
Omdat in browservensters smaller of lager dan 480 px de ingebouwde standaardvideospeler wordt gebruikt, heb ik verder niet gekeken hoe het daar werkt. In kleine venstertjes zoals op smartphones is het gewoon echt veel beter de standaardvideospeler te gebruiken.

Op de desktop werken de sleepbalken ook goed, als er wordt gezoomd, ook in het touchscreen van Windows 8. Er kunnen zich wel kleine problemen voordoen, zoals flikkeren als er wordt gesleept. De pagina herladen lost dit op.

Omdat op tablets (getest in Android 4.0.3 en 4.4.2 en iOS 8.2) de sleepbalken van dit voorbeeld bij zoomen niet werken, heb ik ter vergelijking ook de html5-sleepbalk, de jQuery sleepbalk en de jQuery Mobile sleepbalk (van beide de simpelste uitvoering), en de sleepbalk van de ingebouwde standaardspeler geprobeerd. Als de sleepbalken van jQuery of html5 echt veel beter zouden werken, zou dat een reden kunnen zijn die te gebruiken. Dat je het uiterlijk daarvan vrijwel niet kunt wijzigen met behulp van css, zou je dan op de koop toe kunnen nemen.
jQuery en html5 blijken echter alleen op iOS beter te werken, op Android werken alle sleepbalken ruwweg genomen ongeveer even goed. (De sleepbalk van de standaardspeler valt sowieso af, omdat dat zou betekenen dat je de hele standaardspeler moet gebruiken, en dus het uiterlijk van de speler, de plaats van de knoppen, enz. niet kunt aanpassen.)

In Android browser 4.0.3 werkt gewoon geen enkele sleepbalk, ook als er niet wordt gezoomd Niet de sleepbalken uit dit voorbeeld, niet die van jQuery en niet die van html5. De sleepbalken van dit voorbeeld werken als drukknop: waar je de slider aanraakt, daar wordt de geluidssterkte of verstreken speelduur neergezet. Van de sleepbalk van de ingebouwde standaardspeler springen de tranen je zonder zoomen al in de ogen, als die al te voorschijn wil komen.
Als je gaat zoomen, werken de sleepbalken nog steeds niet. Aanraken werkt nu ook niet meer, in geen enkele sleepbalk.
In Android browser 4.4.2 werken de sleepbalken goed, als er niet wordt gezoomd. Bij zoomen werkt de sleepbalk voor beeld niet, voor geluid soms. Dit geldt zowel voor slepen als voor aanraken op een bepaalde plaats binnen de balk. Voor de sleepbalken van jQuery en html5 geldt hetzelfde, die van jQuery Mobile werkt soms heel gebrekkig. De sleepbalk van de ingebouwde standaardspeler werkt ook bij zoomen goed.

In Firefox werken op Android 4.4.2 de sleepbalken bij zoomen niet meer. Dat geldt ook voor de sleepbalken van jQuery, jQuery Mobile en html5. De sleepbalk van de standaardspeler werkt ook bij zoomen goed.
Chrome en Opera op Android 4.4.2: bij zoomen werken de sleepbalken van de videospeler niet. Ook die van jQuery en html5 werken niet. De sleepbalk van jQuery Mobile werkt wel, maar zo ontzettend traag dat hij feitelijk ook niet werkt. De sleepbalk van de standaardvideospeler werkt ook bij zoomen goed.

Safari op iOS en Chrome for iOS: bij zoomen werken de sleepbalken niet meer. Dat geldt ook voor de sleepbalk van jQuery. De sleepbalken van html5, jQuery Mobile en van de standaardvideospeler werken ook bij zoomen goed (maar zijn dus niet of nauwelijks op te maken met css.)

Pop-up met uitleg opent niet bij hoveren na klikken

Hoewel dit eigenlijk niets met de videospeler te maken heeft, staat het op de pagina, dus hoort het ook hier thuis.
Als de pop-up is geopend door middel van klikken op of aanraken van het vraagteken, en daarna weer is gesloten, opent de pop-up niet meer, als je over het vraagteken hovert. Eerst moet je buiten het vraagteken klikken, of een ander element moet op een of andere manier focus hebben gekregen.
Dit is een eerste versie van een pop-up die werkt zonder JavaScript, ook op touchscreens. In de css is expliciet opgenomen dat de pop-up niet mag openen, als bepaalde onderdelen van de pop-up focus hebben. Dat is de verklaring.
(Op de site zelf is dit inmiddels nog iets verder uitgewerkt, waardoor je inmiddels ook met het toetsenbord de pop-up geopend kunt laten en kunt sluiten. Ook bovenstaand probleem is opgelost. Maar dat zou dit voorbeeld nog uitgebreider maken, dus dat komt ooit wel ’ns in ’n apart iets. Mocht je belangstelling hebben: o.a. de inhoudsopgave van de uitleg bij dit voorbeeld werkt op deze manier.)

Het contextuele menu werkt niet binnen de bedieningselementen van de speler

Dat klopt, dat is uitgeschakeld. En dat is geen probleem. Het zou juist een probleem zijn, als het wel werkt. Het zou dan op een touchscreen in sommige browsers worden geopend, als je ’n knop iets langer aanraakt. Terwijl het binnen de bedieningselementen geen enkel nut heeft.
Overigens is het alleen binnen de bedieningselementen uitgeschakeld, elders op de pagina werkt het gewoon, ook op de video zelf.

Op touchscreens blijven sommige knoppen na loslaten repeteren

Het gaat hier om de knoppen Harder, Zachter, Tien seconden vooruit, Tien seconden achteruit, Vijf procent vooruit en Vijf procent achteruit.
De oorzaak van dit probleem is me niet helemaal duidelijk. Ik vermoed dat het met de kwaliteit van het scherm te maken heeft, want het doet zich ook niet altijd voor.
Dit kan nogal verwarrend zijn. Als de geluidssterkte bijvoorbeeld op 3% staat en de knop blijft repeteren, eindigt dat pas bij 100%. Waarmee je dus ongewild de buren uit bent kunt laten stuiteren.
Om dit te voorkomen stopt het repeteren bij gebruik van een touchscreen of de muis bij verandering van de geluidssterkte na een verhoging of verlaging van 20%. Pas na opnieuw aanraken of aanklikken van de toets, gaat deze

weer repeteren. Als je op deze manier van 0% naar 100% wilt, moet je de knop dus vijf keer aanraken of aanklikken. Bij gebruik van het toetsenbord wordt gewoon onbeperkt gerepeteerd, want daar speelt dit probleem niet.
Bij vooruit of achteruit gaan wordt het repeteren gestopt na tien keer. Je kunt dus maximaal 100 seconden of 50% voor- of achteruit gaan. Ook hier wordt bij gebruik van het toetsenbord gewoon onbeperkt gerepeteerd.

Een inline-style bij <body> verdwijnt op mysterieuze wijze

Daar is niets mysterieus aan, die wordt er gewoon door het script uit geschopt.
Het gebruik van een inline-style is eigenlijk altijd een bijzonder slechte gewoonte. Als je iets aan de css wilt wijzigen en je gebruikt inline-styles, moet je dat op tig plaatsen wijzigen. De kans dat je iets vergeet, is levensgroot.
Vanwege een bug in Safari op OSX voegt het script een kleine inline-style toe aan <body>, zodra wordt gekozen voor aangepaste of ingebouwde videospeler. Deze style overschrijft een eventueel al bij <body> aanwezige inline-style. Gewoon niet gebruiken dus, maar netjes in een extern stylesheet of in een style in de <head> zetten.
Meer hierover staat bij document.body.setAttribute(”style”, ”zoom: 1.0001”);.

Android browser (Android 4.0.3 en 4.4.2)

Het onder dit kopje staande geldt alleen voor de standaardbrowser van Android, niet voor Opera, Chrome of Firefox.

- Op Android 4.0.3 kan de geluidssterkte alleen worden geregeld met behulp van de knop op het apparaat, niet met de knoppen of sleepbalk in de videospeler. Dit is een bug in Android browser.
  Op Android 4.4.2 kan de geluidssterkte wel met de knoppen en de sleepbalk van de videospeler worden geregeld.
  (Anders dan bij iOS is testen op de mogelijkheid de geluidssterkte te veranderen niet mogelijk, omdat Android browser 4.0.3 meldt dat de geluidssterkte is gewijzigd. Dat is echter een vuige leugen: in werkelijkheid is de geluidssterkte helemaal niet gewijzigd.)
- Op Android 4.0.3 werkt slepen in de sleepbalken voor geluid en beeld niet. Als je de sleepbalk aanraakt, wordt wel de aangeraakte positie ingesteld. Een gewone aanraking werkt dus wel. Op Android 4.4.2 werken de sleepbalken wel goed.
- Op Android 4.0.3 repeteren de knoppen voor Harder en Zachter, Vijf procent verder en terug, en Tien seconden verder en terug niet, als je ze blijft aanraken. Op Android 4.2.2 repeteren ze wel, als je ze blijft aanraken.
- De snelheid kan niet worden gewijzigd. De radioknoppen geven wel een andere snelheid aan, en die andere snelheid wordt zelfs gemeld als je die opvraagt in JavaScript, maar in werkelijkheid verandert de snelheid niet. Dit is gewoon onmogelijk in deze browser.
  Op Android 4.0.3 veranderen de resterende en afgespeelde speelduur niet meer, als je de snelheid wijzigt. Ondanks dat die snelheid dus niet echt verandert. Op Android 4.2.2. is dit verholpen.
  (’n Tijdje geleden kon je ook Firefox op Android de snelheid niet softwarematig wijzigen, inmiddels wel, dus mogelijk gaat dit in de toekomst ook in deze browser werken.)
- Als de video op Android 4.0.3 fullscreen wordt afgespeeld, wordt hij in de linkerbovenhoek van het scherm gezet, als weer wordt teruggegaan naar normale

weergave. Dit geldt voor alle video's, dus als je dit zes keer doet, staan er zes video's gezellig bovenop elkaar. De knoppen werken allemaal nog gewoon, dat wel.
In Android 4.4.2 is dit verholpen.

- In Android browser 4.0.3 krijgt de sleepbalk van de zesde videospeler op de tweede pagina geen gradiënt als achtergrond, maar is egaal geel. Dit kun je eventueel oplossen door de oudere vorm van `-webkit-linear-gradient` te gebruiken voor deze browser. Persoonlijk vond ik dat niet de moeite waard.
- Bij de eerste videospeler op de tweede pagina is het lijntje aan de linkerkant van de sleepbalk voor afspelen niet rond, maar recht. Je ziet dit pas als de video gaat spelen, of als je sleept. Hoewel in versie 4.4.2 ongelooflijk veel bugs zijn gerepareerd, zit dit kleine bugje (buggetje? buggy? buggertje?) er nog in.
- Alleen Android 4.4.2: bij openen van de pagina is geen van de radioknoppen aangevinkt. Je ziet even heel kort een vinkje, daarna verdwijnt dat weer. Als een van de radioknoppen wordt aangeraakt, verschijnt het vinkje weer op de normale wijze. Het lijkt erop dat Android browser 4.4.2 het checked-attribuut na 'n seconde weer 'vergeet'. In Android browser 4.0.3 werkt dit, zoals het hoort te werken.
  De oorzaak heb ik niet kunnen vinden. Omdat deze browser de enige is die dit doet, en eerdere versies dit probleem niet hadden, neem ik aan dat het om een bug in deze browser gaat.
- Bij de zesde videospeler op de vijfde pagina verschijnt geen draaiende lijn in het midden van de video. De knoppen Afspelen/pauzeren en Geluid aan/uit hebben geen draaiende rand. De border rondom de video verkleurt niet. Omdat het hier eigenlijk niet om gaat (en er genoeg onzin overblijft bij deze videospeler), heb ik dit verder niet uitgezocht.
- Alleen Android 4.0.3: bij de zesde videospeler op de vijfde pagina slingert de snelheidsregeling heen en weer. Op ogenschijnlijk willekeurige momenten verdwijnt de hele snelheidsregeling. Omdat ook deze snelheidsregeling een project is van de Afdeling Dolzinnigheid Voor Gevorderden, heb ik ook dit niet verder uitgezocht.
- Als de speelduur (nog) onbekend is en bepaalde knoppen of de sleepbalk voor afspelen worden gebruikt, verschijnt een melding dat dit nog niet kan. Onderaan deze melding staat een sluitknop. Deze knop staat in Android browser niet horizontaal gecentreerd, omdat `margin: 0 auto;` niet werkt, als de knop geen vaste breedte heeft. (Als de knop een vaste breedte krijgt, ziet dat er niet uit in andere browsers.)

Chrome (Android 4.4.2)

De snelheid kan niet worden gewijzigd. De radioknoppen geven wel een andere snelheid aan, en die andere snelheid wordt zelfs gemeld, als je die opvraagt in JavaScript, maar in werkelijkheid verandert de snelheid niet. Dit is gewoon onmogelijk in deze browser. ('n Tijdje geleden werkte dit ook niet in Firefox op Android, inmiddels wel, dus mogelijk gaat dit in de toekomst ook in deze browser werken.)

Chrome for iOS

Deze browser maakt gebruik van de weergave-machine van Safari. Daarom staat alles voor deze browser hieronder bij Safari op iOS.

Firefox (Android)

- Als je de eerste keer het vraagteken aanraakt om de pop-up met uitleg te openen, moet je dat twee keer aanraken, voor de pop-up opent. Hoewel dit eigenlijk niets met de videospeler te maken heeft, staat het op de pagina, dus hoort het ook hier bij de Bekende problemen.

Dit is een eerste versie van een pop-up die werkt zonder JavaScript, ook op touchscreens. Op de site zelf is dit inmiddels nog iets verder uitgewerkt, waardoor dit probleem is opgelost. Maar dat zou dit voorbeeld nog uitgebreider maken, dus dat komt ooit wel 'ns in 'n apart iets. Mocht je belangstelling hebben: o.a. de inhoudsopgave van de uitleg bij dit voorbeeld werkt op deze manier.

- De knoppen voor Harder en Zachter, Vijf procent verder en terug, en Tien seconden verder en terug werken goed bij gewoon aanraken. Maar als je ze laat repeteren, blijven ze repeteren, tot het scherm ergens buiten de knop wordt aangeraakt.
  Dit verschijnsel begon in de laatste versie van Firefox op Android. Dit is waarschijnlijk een tijdelijk verschijnsel.
  De meer technische uitleg: op dit ogenblik gebruik ik hand.js van Microsoft, zodat Pointer events ook werken in andere browsers dan Internet Explorer. Mozilla is bezig Pointer events te implementeren. Als in about:config 'dom.w3c_pointer_events.enabled' op true wordt gezet, stopt het repeteren. Standaard staat dit nog op false, maar ik neem aan dat dit binnenkort standaard op true staat en dat dit probleem dan is opgelost. Kennelijk is er op dit moment even 'n botsing tussen waar Mozilla mee bezig is (het implementeren van Pointer events) en hand.js van Microsoft (de tijdelijke oplossing die ik gebruik).
  Omdat dit probleem vaker speelde (op goedkope touchscreens), is een rem op het repeteren gezet. Bij verandering van geluidssterkte stopt dit na een verandering van 20%, bij voor of achteruitgaan na een verandering van 100 seconden of 50%. Meer hierover bij Op touchscreens blijven sommige knoppen na loslaten repeteren.

Firefox (OS X)

Bij klikken op een knop binnen de videospeler, krijgt deze geen focus. Bij klikken op een sleepbalk krijgt deze wel focus, het geldt alleen voor de knoppen.
Beste Applefan, ik zal je de moeite besparen: foei, hoe durf je dat een probleem te noemen! Dat is geen probleem, dat is wat Apple een extra noemt. Firefox gedraagt zich hier net als Safari. Zie verder bij Safari (OS X).

Firefox (Windows 8 met touchscreen)

- De knoppen voor Harder en Zachter, Vijf procent verder en terug, en Tien seconden verder en terug repeteren niet, als je ze blijft aanraken. Als je deze knoppen bedient met muis, toetsenbord (spatiebalk of Enter) of touchpad, repeteren ze wel.
- De pop-up met hulp opent pas, nadat het vraagteken twee keer is aangeraakt. (Dit is een eerste versie van een pop-up die werkt zonder JavaScript, ook op touchscreens. Op de site zelf is dit inmiddels nog iets verder uitgewerkt, waardoor dit probleem is opgelost. Maar dat zou dit voorbeeld nog uitgebreider maken, dus dat komt ooit wel 'ns in 'n apart iets. Mocht je belangstelling hebben: o.a. de inhoudsopgave van de uitleg bij dit voorbeeld werkt op deze manier.)
- Bij de tweede video op de tweede pagina en bij de tweede video op de derde pagina verkleurt de sleepbalk niet altijd bij focus. Zie voor een uitgebreider verhaal gelijk hieronder het eerste onderwerp bij Firefox (OS X en Windows).

Firefox (OS X en Windows)

- Bij de tweede videospeler op de tweede pagina verkleuren sleepbalk en sleepknop niet, als precies op de sleepknop wordt geklikt, of als deze wordt aangeraakt.
  Hetzelfde speelt bij de tweede speler op de derde pagina, maar dan voor de sleepbalk voor geluidssterkte. Het gemeenschappelijke van beide sleepknoppen is de tekst die op de knop staat (respectievelijk percentage dat is afgespeeld en percentage geluidssterkte).

Als op de balk van de sleepbalk wordt geklikt, of als de balk wordt aangeraakt, verkleuren balk en knop wel. Ze verkleuren alleen niet, als je precies op de sleepknop klikt of precies de knop aanraakt.
Eigenlijk moet ik het beter formuleren: de sleepbalk krijgt geen focus, en verkleurt dus ook niet. De toetsen om de sleepbalk te bedienen (pijltjes e.d.) werken ook niet, omdat de sleepbalk geen focus krijgt.
Bij gebruik van de Tab-toets verkleuren balk en knop wel en werken de toetsen ook gewoon. Feitelijk is dit geen groot probleem. Het aangeven dat ’n bedieningselement focus heeft, is bedoeld voor gebruikers van het toetsenbord, en daar werkt het wel goed. Weinig mensen zullen met de muis op de sleepbalk klikken, om daarna met het toetsenbord de sleepbalk te gaan bedienen.
(Firefox is bezig Pointer events te implementeren. Op dit moment gebruik ik daar hand.js voor, een script om Pointer events in andere browsers dan Internet Explorer te laten werken. Als je het experimentele Pointer events inschakelt in Firefox, werkt klikken op de knop wel. Dus dit is binnenkort opgelost. Overigens zit er een bug in Firefox die niets met dit voorbeeld heeft te maken: als je Pointer events inschakelt, wat dus waarschijnlijk binnenkort standaard gaat gebeuren, kun je op touchscreens niet meer met de hand scrollen. Hopelijk wordt die bug snel geplet.)

Google Chrome (alle versies)

- Als bij <video> `preload="none"` werd ingevuld, duurde het tot voor kort tot wel tien seconden voordat de pagina werd weergegeven. Inmiddels is deze bug gerepareerd.
- In de zesde videospeler op de vijfde pagina beweegt de knop Vijf procent verder niet van links naar rechts over het venster. Mogelijk dat het wel aan de praat is te krijgen, als je bijvoorbeeld meer animatie-eigenschappen gebruikt of zo (dat hoort niet, maar dat soort trucs werkt wel vaker), maar dat heb ik verder niet uitgezocht. Ik kan me namelijk niet voorstellen dat iemand echt serieus z’n bezoekers weg gaat jagen door dit daadwerkelijk ergens te gaan gebruiken. (Op iOS werkt dit wel, omdat Chrome daar de weergave-machine van Safari gebruikt.)

Google Chrome (Windows 7 en 8)

Inmiddels lijkt deze bug gerepareerd, maar tot voor kort leverde het afspelen van een mp4 een probleem op. Als je met behulp van de sleepbalk of de toetsen naar het eind van de video was gegaan, dus zonder hem gewoon af te laten spelen in het normale tempo, kon de video niet opnieuw worden afgespeeld. Pas als de pagina opnieuw was geladen, kon de video opnieuw worden afgespeeld. Vandaar dat bij de <src> in de <video> webm voor mp4 staat, want bij webm speelde dit probleem niet.
Maar inmiddels is deze bug dus kennelijk gerepareerd.

Internet Explorer (alle versies)

- De eerste videospeler op de tweede pagina heeft een aangepaste sleepbalk voor afspelen. Hoe verder de video is afgespeeld, hoe langer de blauwe achtergrond op de balk wordt. Bij klikken op de balk moet deze achtergrond oranje worden, om aan te geven dat de balk focus heeft. Bij klikken op het witte deel van de balk gebeurt dat ook, maar bij klikken op het blauwe/oranje deel gebeurt dat niet: de achtergrond die aangeeft hoeveel er is afgespeeld, blijft dan blauw.

Bij klikken op het blauwe of oranje deel van de sleepbalk blijft in Internet Explorer blauw blauw en wordt oranje blauw, terwijl het in beide gevallen oranje zou moeten zijn.

Als de sleepbalk door middel van de Tab-toets focus krijgt, werkt het wel goed.

Omdat dit niet zo ernstig is, heb ik hier verder niet naar gekeken. (Meer over de Tab-toets en focus is te vinden bij Tabindex en Focus.)

- Bij alle video's op pagina vier en bij de video's onderaan op pagina vijf wordt gebruik gemaakt van `outline-offset`. Internet Explorer kent deze css-eigenschap niet. Bedieningselementen van deze video's krijgen, als ze focus hebben, een rand rondom. Deze staat aan de bovenkant iets over de video heen, omdat hij niet op de juiste plaats gezet kan worden. Dat gebeurt namelijk met behulp van `outline-offset`.
- In de zesde videospeler op de vijfde pagina veranderen de kleuren van de border niet. Voor Internet Explorer 9 is dat niet vreemd, want die kent het voor deze verandering gebruikte `@keyframes` niet, maar ook in Internet Explorer 10 en 11 werkt dit niet. Mogelijk dat het wel aan de praat is te krijgen, als je bijvoorbeeld meer animatie-eigenschappen gebruikt of zo (dat hoort niet, maar dat soort trucs werkt wel vaker), maar dat heb ik verder niet uitgezocht. Ik kan me namelijk niet voorstellen dat iemand echt serieus z'n bezoekers weg gaat jagen door dit daadwerkelijk ergens te gaan gebruiken.
- In de zesde videospeler op de vijfde pagina beweegt de knop Vijf procent verder niet van links naar rechts over het venster. Verder zelfde verhaal als gelijk hierboven.
- Bij de sleepbalken voor afspelen en geluidssterkte verkleuren sleepbalk en sleepknop niet, als precies op de sleepknop wordt geklikt, of als deze wordt aangeraakt.
  Als op de balk van de sleepbalk wordt geklikt of als de balk wordt aangeraakt, verkleuren balk en knop wel. Ze verkleuren alleen niet, als je precies op de sleepknop klikt, of precies de knop aanraakt.
  Eigenlijk moet ik het beter formuleren: de sleepbalk krijgt geen focus, en verkleurt dus ook niet. De toetsen om de sleepbalk te bedienen (pijltjes e.d.) werken ook niet, omdat de sleepbalk geen focus krijgt.
  Bij gebruik van de Tab-toets verkleuren balk en knop wel en werken de toetsen ook gewoon. Feitelijk is dit geen groot probleem. Het aangeven dat 'n bedieningselement focus heeft, is bedoeld voor gebruikers van het toetsenbord, en daar werkt het wel goed. Weinig mensen zullen met de muis op de sleepbalk klikken, om daarna met het toetsenbord de sleepbalk te gaan bedienen.

Internet Explorer 9

- Voordat de video wordt geopend, moet een afbeelding worden getoond. Deze is opgegeven met het poster-attribuut bij <video>. Bij de eerste videospeler ziet dit er als volgt uit:

```
poster="../103-images/s_brock_cellular_
      respiration.jpg"
```

  Deze afbeelding moet worden getoond, voordat de video wordt afgespeeld. Internet Explorer 9 toont deze afbeelding alleen, als bij <video> `preload="none"` staat. Strikt gezien is dat een correcte uitleg van de specificatie, maar wel de soort uitleg die alleen door een wel heel creatief figuur aan de specificatie kan worden gegeven. (De afbeelding wordt volgens de specificatie alleen getoond, als de video niet beschikbaar is. En inderdaad, als je `preload="metadata"` gebruikt, dan ís de video beschikbaar. Je ziet hem wel niet, maar hij ís beschikbaar.)
  In Internet Explorer 10 en later werkt dit niet meer volgens de juridisch-technische uitleg van de specificatie, maar gewoon zoals het hoort te werken: de afbeelding wordt gewoon getoond.
- Als bij <video> `preload="none"` of `preload="metadata"` staat, kan de video soms wel, soms niet opnieuw worden afgespeeld. Soms kan de video eerst niet opnieuw worden afgespeeld, maar iets later plotseling weer wel. Soms moet eerst de

pagina worden herladen. Dit probleem speelt ook op veel pagina's elders op internet die met <video> werken, dus het gaat kennelijk om een bug in Internet Explorer 9.
Als de de attributen `loop` én `autoplay` worden gebruikt, lukt opnieuw afspelen wel. Alleen spelen alle video's dan dus non-stop af, steeds opnieuw.
In het script gelijk voor `play()` een `load()` geven werkt ook, maar dan gaan sommige andere browsers bij opnieuw afspelen de hele video opnieuw downloaden, ook als deze in de cache is opgeslagen.
Ik heb geen oplossing voor deze bug kunnen vinden. Althans: geen die bruikbaar is. Testen in het script werkt ook niet, omdat wordt gemeld dat de video kan worden afgespeeld, ook als dat in werkelijkheid dus niet meer kan.
Overigens loopt ook het gebruik van deze verouderde browser snel terug, dus lang zal dit probleem niet meer spelen.

- Op een aantal plaatsen zijn gradiënts (verlopende kleuren) gebruikt. Hiervoor is de css-eigenschap `linear-gradient` gebruikt. Internet Explorer 9 kent die niet en negeert deze dus. Je zou dit kunnen opvangen door voor deze browser een zogenaamd filter te gebruiken, een soort eigen gradiënt van Microsoft voor oudere versies van Internet Explorer. Maar dat vind ik de moeite niet meer waard: gewoon geen verlopende kleuren in dit fossiel.
- Bij de zesde videospeler op de vijfde pagina bewegen de bedieningselementen niet, omdat Internet Explorer 9 `@keyframes` niet kent. Mogelijk zou dat met nogal wat moeite op te lossen zijn, maar ik houd nou eenmaal niet van fossielen die maar niet willen doodgaan. Een welopgevoed fossiel hoort gewoon te gaan hemelen na overschrijding van de aaibaarheidsdatum.
- Op de pagina's met video's staat bovenin een korte beschrijving van de video's. In bredere browservensters is deze beschrijving opgedeeld in kolommen, zodat de regels niet te lang worden. Internet Explorer 9 kent `column-...` niet, dus daar blijft de tekst gewoon in één kolom staan. Met wat extra css voor deze browser is dat niet al te ingewikkeld op te lossen, maar dat heb ik dus niet gedaan.

Internet Explorer 10

- In een aantal videospelers kan de snelheid worden geregeld met behulp van een aantal radioknoppen. Als de video draait en je klikt op een van die radioknoppen, verandert de snelheid.
  Je kunt ook met behulp van de pijltjestoetsen door de radioknoppen lopen. Als de video draait en je verandert de eerste keer van knop, verandert de snelheid ook.
  Als je echter, terwijl de video speelt, nu nog eens van snelheid verandert door op een andere radioknop te klikken, of door met de pijltjestoetsen nogmaals van knop te veranderen, gebeurt er niets.
  Je moet eerst op de Speel-pauzeerknop klikken, pas dan verandert de snelheid. Op de Speel-pauzeerknop zelf moet je twee keer klikken, voor de knop werkt. (Je kunt ook met behulp van Shift+Tab teruggaan naar de Speel-pauzeerknop en dan nogmaals Enter of de spatiebalk indrukken, dat werkt hetzelfde als erop klikken.)
  In Internet Explorer 9 en 11 werkt dit, zoals het verondersteld wordt te werken: de snelheid verandert gelijk als je 'n andere radioknop selecteert. Kennelijk is dit een bug die er na Internet Explorer 9 in is geslopen en die in Internet Explorer 11 al weer is geplet. (Bugs zijn insecten, dus die plet je. Excuses aan entomologen, maar zo heet dat nu eenmaal.)
- Bij de zesde videospeler op de vijfde pagina verschijnt geen draaiende lijn in het midden van de video. Omdat het hier eigenlijk niet om gaat (en er genoeg onzin overblijft bij deze videospeler), heb ik dit verder niet uitgezocht.

Lynx

In de tekstbrowser Lynx blijkt, tot mijn verbazing, alles te werken. Omdat Lynx JavaScript niet ondersteunt, zie je uiteraard geen knoppen e.d. Maar alle teksten zijn aanwezig en als je ’n video aanklikt, kan die worden afgespeeld in een of andere ingebouwde speler, of in een speler van het besturingssysteem.

Opera (Android 4.4.2)

- Bij het afspelen vult de video altijd het volledige venster van de browser. Dit gebeurt niet alleen in dit voorbeeld, maar is standaardgedrag bij álle video’s. Hierdoor verdwijnt wel de aangepaste videospeler, want deze wordt vervangen door de in Opera ingebouwde standaardvideospeler.
  Door op de Terug-knop van het apparaat te drukken, wordt de video weer niet-venstervullend vertoond en verschijnt de aangepaste videospeler weer.
- De snelheid kan niet worden gewijzigd. De radioknoppen geven wel een andere snelheid aan, en die andere snelheid wordt zelfs gemeld als je die opvraagt in JavaScript, maar in werkelijkheid verandert de snelheid niet. Dit is gewoon onmogelijk in deze browser. (’n Tijdje geleden werkte dit ook niet in Firefox op Android, inmiddels wel, dus mogelijk gaat dit in de toekomst ook in deze browser werken.)

Opera (Linux)

Op dit ogenblik gebruikt Opera op Linux nog een eigen weergave machine: Presto. Alle andere versies van Opera zijn al overgestapt op Blink, en Opera op Linux volgt binnenkort. Daarom ga ik geen tijd meer stoppen in het verhelpen van problemen in deze versie van Opera. Maar voor de volledigheid volgt hier wel ’n lijstje met problemen in deze versie van Opera.

- Bij de knoppen in de videospelers werkt de spatiebalk niet in alle knoppen. Normaal genomen kunnen deze knoppen ook worden bediend met de spatiebalk, maar in Opera op Linux kan dat in sommige knoppen alleen met de Enter-toets. (Dit was overigens standaard in alle versies van Opera, tot deze browser overstapte op Blink.)
- Bij de eerste videospeler op de tweede pagina is het lijntje aan de linkerkant van de sleepbalk voor afspelen niet rond, maar recht. Je ziet dit pas als de video gaat spelen, of als je sleept.
- De zesde videospeler op de vijfde pagina is een weergaveramp. Dat is die speler hoe dan ook, maar in Opera op Linux wordt hij ook nog ’ns zwaar verminkt weergegeven. Ik ga niet de hele lijst geven van wat er mis is, want het is gewoon één groot drama. Het humeur van Diederik Samson is er niets bij. De video speelt wel gewoon af. En de knoppen lijken goed te werken, voor zover dat was te testen.

Opera (OS X en Windows)

Als bij <video> `preload="none"` werd ingevuld, duurde het tot voor kort tot wel tien seconden voordat de pagina werd weergegeven. Inmiddels is deze bug gerepareerd.

Opera Mini (alle versies)

Opera Mini vertoont geen video’s. Sterker nog: er wordt zelfs geen videospeler getoond. Wel kun je in principe de video’s gewoon downloaden en in ’n ander programma afspelen. (Met behulp van uitbreidingen schijn je wel video’s te kunnen bekijken in deze browser, maar daar heb ik verder niet naar gekeken, omdat dat niet relevant is voor dit voorbeeld.)

Safari (iOS) en Chrome for iOS

Chrome for iOS gebruikt de weergave-machine van Safari, daarom geldt de tekst hieronder ook voor Chrome for iOS.

- De geluidssterkte kan niet worden aangepast met behulp van de knoppen in de videospeler, maar alleen met de knop op het apparaat zelf. Dit is een eigenschap van iOS. In het script zit een testje ingebouwd, waardoor iOS een extra class krijgt. Hierdoor is de geluidsregeling op iOS te verbergen, van een ander uiterlijk te voorzien, op te leuken met de tekst 'Apple is stom', of wat dan ook.
- Bij <video> staat `preload="metadata"`. In principe zouden hierdoor dingen als speelduur al moeten worden ingelezen. Maar dit attribuut is een aanbeveling en geen verplichting. Op iOS wordt dit genegeerd, waardoor de speelduur pas wordt opgehaald als de video wordt gestart.
  Zolang de speelduur nog onbekend is, wordt als speelduur 00:00 ingevuld.
  Bij de laatste update voor iOS, die aankwam toen deze tekst vrijwel af was, lijkt het erop dat ook deze browsers nu de speelduur e.d. ophalen, voordat de video wordt gestart. Op de pagina's van Apple met wat Safari ondersteunt, staat echter nog dat `preload` wordt genegeerd. Helemaal zeker weet ik dit dus niet.
- De sleepbalk voor afspelen kan alleen worden gesleept over het deel van de video dat al is gedownload. Tot die tijd werkt alleen aanraken van de sleepbalk op een bepaalde positie. De sleepbalk van de ingebouwde standaardvideospeler heeft precies hetzelfde gedrag, behalve dat aanraken op een bepaalde positie daar helemaal geen effect heeft. Kennelijk is dit iets dat is ingebakken in iOS.
- Bij de zesde videospeler op de vijfde pagina verschijnt geen draaiende lijn in het midden van de video. Omdat het hier eigenlijk niet om gaat (en er genoeg onzin overblijft bij deze videospeler), heb ik dit verder niet uitgezocht.

Safari (OS X)

- Bij klikken op een knop binnen de videospeler, krijgt deze geen focus. Bij klikken op een sleepbalk krijgt deze wel focus, het geldt alleen voor de knoppen.
  Foei, dat is geen probleem, dat is een extra van Apple! Om redenen die alleen een échte fan van Apple je vermoedelijk kan vertellen, krijgt een knop gewoon nooit focus, als je erop klikt in Safari op OS X. Met de Tab-toets zijn de knoppen wel één voor één af te lopen, en ook Ctrl+Alt+→ en Ctrl+Alt+← om van video naar video te gaan werken gewoon.
  Dit zou trouwens met JavaScript zijn op te lossen, maar ik neem aan dat gebruikers van OS X dit gedrag om voor mij ondoorgrondelijke redenen waarderen. Anders zou Apple het niet zo inbouwen, neem ik aan.
  Google Chrome volgde dit gedrag trouwens tot voor kort, maar behandelt nu focus net als in al hun andere versies.
- Bij de zesde videospeler op de vijfde pagina verschijnt geen draaiende lijn in het midden van de video. Omdat het hier eigenlijk niet om gaat (en er genoeg onzin overblijft bij deze videospeler), heb ik dit verder niet uitgezocht.

Validatie

- Op een aantal plaatsen wordt `@webkit-keyframes` gebruikt om een bug in oudere versies van Android browser te omzeilen. Dit valideert niet. Omdat de reden van deze foutmelding precies bekend is, is dit geen probleem.
  (Omdat dit alleen voor oudere versies van Android browser nodig is, gebruik ik in dit geval `@keyframes` alleen met het voorvoegsel `-webkit-`. In dit geval is het handig dat browsers die niet op webkit zijn gebaseerd dit negeren. Maar normaal genomen is het echt hartstikke fout om iets te gebruiken, dat gericht is op één weergave-machine.)

- Het script voegt bij elke met bedieningselementen `touch-action` en `user-select` toe (en diverse varianten met voorvoegsels: `-ms-touch-action`, `-moz-user-select`, `-ms-user-select`, `-webkit-user-select`).
  Deze eigenschappen worden als inline-style in de html gezet, waardoor je bij normaal valideren geen foutmelding krijgt. Maar als je ze apart in een stylesheet zet en valideert, worden ze niet gevalideerd.
  `user-select` wordt nog niet gevalideerd, maar is wel onderweg. `touch-action` maakt nog geen deel uit van welke (ontwerp-)standaard dan ook, maar werkt al in alle browsers. Beide zijn onmisbaar voor een fatsoenlijke werking van de spelers.
  Het gebruik van `user-select` is niet helemaal zonder risico, omdat dit in nog geen enkele specificatie staat. In de toekomst zou de werking ervan daardoor wat kunnen veranderen. Maar omdat het hier heel zwart-wit wordt gebruikt (gewoon helemaal niets selecteren), durf ik het toch wel te gebruiken.

Zoomen en andere lettergrootte

Een kleinere lettergrootte is nergens een probleem, hoewel het er niet mooier op wordt.
Een grotere lettergrootte kan wel een probleem zijn. Op de eerste en vijfde pagina wordt voor de symbolen op de knoppen een gewoon font gebruikt. Tot ongeveer 150% is het nog bruikbaar, daarboven gaat het echt mis. Op de tweede, derde en vierde pagina, waar een webfont, achtergrond-afbeelding en data-uri's worden gebruikt, blijft alles bruikbaar. Bruikbaar, het ziet er bepaald niet mooi uit. Feitelijk is het vergroten van alleen tekst niet goed mogelijk.
Inzoomen (vergroten) is nergens een probleem, met uitzondering van de sleepbalken. Daar staat een apart verhaal over bij Bij zoomen werken de sleepbalken soms niet goed.
Uitzoomen (verkleinen) werkt ook, maar sommige knoppen verspringen van plaats, omdat ze niet meer naast elkaar passen en op de volgende regel komen te staan.

Wijzigingen

Alleen grotere wijzigingen worden hier vermeld, geen dingen als een link die is geüpdatet.
17 april 2015:
Nieuw opgenomen.

Inhoud van de download en licenties

afbeelding-103-dl.html: de pagina met het overzicht van de spelers
afbeelding-103.txt: een kopie van de tekst onder dit kopje (inhoud download en licenties)
103-css-dll:
- afbeelding-103-dl.css: stylesheet voor pagina met overzicht van de spelers
- afbeelding-103-1-dl.css: stylesheet voor eerste pagina met videospelers
- afbeelding-103-2-dl.css: stylesheet voor tweede pagina met videospelers
- afbeelding-103-3-dl.css: stylesheet voor derde pagina met videospelers
- afbeelding-103-4-dl.css: stylesheet voor vierde pagina met videospelers
- afbeelding-103-5-dl.css: stylesheet voor vijfde pagina met videospelers

103-files-dl:
- afbeelding-103-1-dl.html: eerste pagina met videospelers
- afbeelding-103-2-dl.html: tweede pagina met videospelers
- afbeelding-103-3-dl.html: derde pagina met videospelers
- afbeelding-103-4-dl.html: vierde pagina met videospelers

afbeelding-103-5-dl.html: vijfde pagina met videospelers

103-fonts:

icomoon.ttf: font gebruikt op tweede pagina met videospelers

icomoon.woff: font gebruikt op tweede pagina met videospelers

103-images:

buttons-background-page-3-invert.png: gebruikt voor knoppen derde pagina zesde videospeler

buttons-background-page-3-klein.png: gebruikt voor knoppen derde pagina tweede, vierde en vijfde videospeler

buttons-background-page-3.png: gebruikt voor knoppen derde pagina eerste en derde videospeler

elliot_b_senior_project_with_after_effects.jpg: screenshot tweede videospeler

how_to_drive_fast_with_david_rodriguez.jpg: screenshot vierde videospeler

iconenfont-fallback.png: fallback iconenfont knoppen videospelers op tweede pagina

making_video_a_game_with_bill_decker.jpg: screenshot zesde videospeler

peer_tutoring_program_by_angelina_g.jpg: screenshot vijfde videospeler

photography_of_music_by_victoria_holt.jpg: screenshot derde videospeler

s_brock_cellular_respiration.jpg: screenshot eerste videospeler

sprites-help.png: gebruikt voor afbeeldingen pop-up met uitleg

103-js:

afbeelding-103.js: javascript voor de eerste en tweede pagina met videospelers

afbeelding-103-3.js: javascript voor de derde pagina met videospelers

afbeelding-103-4.js: javascript voor de vierde pagina met videospelers

afbeelding-103-5.js: javascript voor de vijfde pagina met videospelers

afbeelding-103-met-commentaar.js: zelfde als afbeelding-103.js, maar met commentaar

hand.minified-1.3.8.js: javascript afkomstig van hand.js.

103-video:

Deze map bevat alle video's. Van elke video zijn vier stuks aanwezig: een grote in webm-formaat, een kleine in webm-formaat, een grote in mp4-formaat en een kleine in mp4-formaat. De kleine zijn herkenbaar aan het woord 'klein' in de naam. De video's zijn afkomstig van het Internet Archive. Op de pagina's met videospelers staat boven elke video de precieze pagina op die website.

De video's vallen, op Cellular Respiration – Block na, onder de Public Domain Creative Commons license.

Cellular Respiration – B Block valt onder de Attribution-Noncommercial-No Derivative Works 3.0 United States Creative Commons license. Het kleine formaat van deze video is ingekort tot de eerste tien seconden. De producer van deze video is Sharon Brock.

Verschil tussen de diverse html-, JavaScript- en css-bestanden

CSS

'afbeelding-103-dl.css' is een apart geval: dit hoort bij de overzichtspagina. De andere vijf css-bestanden (die met een volgnummer in de haam) horen bij de pagina's met videospelers. Deze bestanden zijn echt volledig van elkaar verschillend. Alleen het eerste deel is grotendeels hetzelfde. Dat is het deel waarin de pop-up met uitleg, de links naar vorige en volgende pagina, e.d. staan.

Deze css-bestanden regelen het uiterlijk van de videospelers volledig. Omdat dat uiterlijk nogal verschilt, is het logisch dat de bijbehorende css-bestanden ook verschillen. Het volgnummer in de naam van het bestand geeft aan, bij welke pagina ze horen. 'afbeelding-103-1-dl.css' hoort bij de eerste pagina, 'afbeelding-103-2-dl.css' bij de tweede, enz.

In principe had alle css ook in één bestand gezet kunnen worden. Dat had als voordeel gehad dat de css voor het eerste deel van de pagina niet vijf keer wordt herhaald. Maar ik vond het nadeel groter wegen: een heel erg lang, volstrekt onoverzichtelijk bestand.
Dat het uiterlijk van de videospelers echt volledig door de css wordt geregeld, kun je makkelijk zelf vaststellen. In de <head> van de eerste pagina met video's staat de volgende regel:

```
<link rel="stylesheet" href="../../../css-
    dl/afbeelding-103-1-dl.css">
```

Als je '103-1' verandert in '103-2', verandert het uiterlijk van de videospelers, en zien de videospelers er plotsklaps hetzelfde uit als op de tweede pagina. Als je de '1' verandert in '3', '4' of '5', zien de videospelers er hetzelfde uit als op de derde, vierde of vijfde pagina.

JAVASCRIPT

'afbeelding-103-met-commentaar.js' is precies hetzelfde als 'javascript-103.js'. Alleen staat in dit bestand uitgebreide uitleg in het script. Dit maakt het bestand met commentaar ruim drie keer zo groot als hetzelfde script zonder commentaar. Het script met commentaar is dan ook niet bedoeld om echt te gebruiken.
De andere vier JavaScript-bestanden zijn vrijwel hetzelfde. Omdat bij sommige videospelers de volgorde van de bedieningselementen op het scherm anders is dan die in de code, is bij de derde, vierde en vijfde pagina de tabindex aangepast. Dat aanpassen gebeurt in het script. Dit is het enige verschil tussen de diverse scripts. 'afbeelding-103.js' hoort bij de eerste en tweede, 'afbeelding-103-3.js' bij de derde, 'afbeelding-103-4.js' bij de vierde en 'afbeelding-103-5.js' bij de vijfde pagina met videospelers.
Als je in de html de volgnummers van het script omwisselt, zul je op het scherm geen enkel verschil zien. Alleen de tabindex is veranderd, maar dat merk je alleen bij gebruik van de Tab-toets. Uitleg over de tabindex staat bij Tabindex (uitleg), hoe je de tabindex in het script kunt wijzigen staat bij TabIndex (wijzgen).

HTML

De pagina met het overzicht heet 'afbeelding-103-dl.html'. Deze is verder weinig interessant.
De vijf pagina's met videospelers zijn vrijwel hetzelfde. Het verhaaltje bovenin, waarin de videospelers worden beschreven, verschilt uiteraard. Net als de links naar vorige en vorige pagina. Ook de links naar de bestanden met JavaScript en css verschillen. Omdat bij de derde, vierde en vijfde pagina in het script de tabindex is aangepast, verschilt op deze pagina's ook de tabindex in de html. Anders zou de volgorde niet meer kloppen. Uitleg over de tabindex staat bij Tabindex.
De belangrijkste html, die voor de videospelers, is op alle vijf de pagina's exact hetzelfde.
Bij de derde, vierde en vijfde pagina zijn nog wat dingen anders. Deze drie pagina's hebben alle drie onderaan een klein stukje JavaScript staan. In het bijbehorende externe JavaScript is de tabindex veranderd, omdat de volgorde van de bedieningselementen in de videospelers in de code anders is dan die op het scherm. Die verandering in het script geldt voor álle videospelers op 'n pagina. Maar omdat ook binnen dezelfde pagina verschillen binnen de videospelers zitten, wordt op de derde en vijfde pagina de tabindex nogmaals aangepast voor sommige aparte spelers. Dit gebeurt onderaan de pagina in een klein scriptje.
Op de derde en vierde pagina worden bij een aantal videospelers WAI-ARIA-attributen verwijderd, ook vanwege de volgorde van de bedieningselementen.

Maar de belangrijkste html, die van de videospelers, is dus precies hetzelfde.
Meer over de scriptjes onderaan de html is te vinden bij Het JavaScript onderaan de html-bestanden.
Ten slotte is in de vierde pagina een kleine aanpassing gemaakt voor de tweede videospeler, waarvan de besturing kan worden verborgen en getoond.

HTML

De html die te maken heeft met de basis van dit voorbeeld is rood gekleurd. Alle niet-essentiële code html staat in een afwijkende `zwarte` lettersoort. (In de inhoudsopgave staat alles in een gewone zwarte letter vanwege de leesbaarheid.)

```
<!DOCTYPE html>
<html lang="nl">
```

Een document moet met een doctype beginnen, om weergaveverschillen tussen browsers te voorkomen. Zonder doctype is de kans op verschillende (en soms volkomen verkeerde) weergave tussen verschillende browsers heel erg groot.
Geldige doctypes vind je op www.w3.org/QA/2002/04/valid-dtd-list.
Gebruik het volledige doctype, inclusief de eventuele url, anders werkt het niet goed.
Het hier gebruikte doctype is dat van html5.

```
<meta charset="utf-8">
```

Zorgt dat de browser letters met accenten e.d. goed kan weergeven. In html5 hoeft deze regel niet langer te zijn, dan wat hier staat.
utf-8 is de beste charset (tekenset), omdat deze alle talen van de wereld (en nog heel veel andere extra tekens) bestrijkt, maar toch niet meer ruimte inneemt voor de code, dan nodig is. Als je utf-8 gebruikt, hoef je veel minder entiteiten (`ä` e.d.) te gebruiken, maar kun je bijvoorbeeld gewoon ä gebruiken.
Deze regel moet zo hoog mogelijk komen te staan, als eerste regel binnen de head, omdat hij anders door sommige browsers niet wordt gelezen. (Feitelijk moet hij binnen de eerste 512 bytes staan, maar als je hem gewoon als eerste regel zet, weet je zeker dat je altijd goed zit.)

```
<meta name="viewport" content="width=device-width,
    initial-scale=1">
```

Mobiele apparaten variëren enorm in breedte. En dat is een probleem. Sites waren, in ieder geval tot voor kort, gemaakt voor desktopbrowsers. En die hebben, in vergelijking met bijvoorbeeld een smartphone, heel brede browservensters. Hoe moet je op ’n smartphone een pagina weergeven die is gemaakt voor de breedte van een desktop? Je kunt natuurlijk wachten tot álle sites zijn omgebouwd voor smartphones, tablets, enz., maar dan moet je waarschijnlijk heel erg lang wachten.
Mobiele browsers gokken erop dat een pagina een bepaalde breedte heeft. Safari voor mobiel bijvoorbeeld gaat ervan uit dat een pagina 980 px breed is. De pagina wordt vervolgens zoveel versmald dat hij binnen het venster van het apparaat past. Op een iPhone wordt de pagina dus veel smaller dan op een iPad. Vervolgens kan de gebruiker inzoomen op het deel van de pagina dat hij of zij wil zien.
Dit betekent ook dat bij het openen van de pagina de tekst meestal heel erg klein wordt weergegeven. (Meestal, want niet alle browsers en apparaten doen het op dezelfde manier.)
Niet erg fraai, maar bedenk maar ’ns ’n betere oplossing voor bestaande sites.
Nieuwe sites of pagina’s kunnen echter wel rekening houden met de veel kleinere browservensters van mobiele apparaten. Deze lay-out bijvoorbeeld past zich aan de breedte van het venster aan, ook bij heel smalle vensters. Maar die stomme mobiele browser weet

dat niet, dus die gaat ervan uit dat ook de heel smalle lay-out 980 px breed is, en verkleint die dan. Dat is ongeveer even behulpzaam als de gedienstige kelner die behulpzaam de stoel naar achteren trekt, net als jij wilt gaan zitten.
Om de door de browser aangeboden hulp vriendelijk maar beslist te weigeren, wordt deze tag gebruikt. Hiermee geef je aan dat de pagina is geoptimaliseerd voor mobiele apparaten. Een iPad in portretmode bijvoorbeeld is 768 px breed. De kreet `width=device-width` zegt tegen de mobiele browser dat de breedte van de weer te geven pagina gelijk is aan de breedte van het apparaat. Voor een iPad in portretmode dus 768 px.
En dat klopt, want de weer te geven pagina past zich automatisch aan de breedte van het apparaat aan, omdat de pagina altijd 100% breed is. Als het apparaat 300 px breed is, is de pagina ook 300 px breed, maar nog altijd 100%. Dit wordt simpelweg bereikt door geen breedte aan <body> e.d. op te geven. In dat geval vult de pagina de volle breedte van het browservenster.
Simpeler gezegd: je zegt tegen het mobiele apparaat dat de pagina geen vaste breedte heeft, dat alles al is geregeld en dat het dus niet nodig is om de weergave aan te passen.
Voor bredere vensters wordt vervolgens aparte css opgegeven, die alleen voor die bredere vensters werkt. Zou je dat niet doen, dan zou een regel tekst even breed worden als een breedbeeldmonitor, wat vrijwel onleesbaar is.

Steeds meer apparaten krijgen een hogere resolutiedichtheid: er staan meer pixels in een inch dan bij een monitor voor de desktop het geval is (hoewel ook op de desktop hogere resolutiedichtheden oprukken). Als een lijn 4 px breed is, en de pixels staan vier keer zo dicht op elkaar als op de desktop, zou de lijn maar 1 px breed zijn op een mobiel apparaat. Het is wel een supermooie en strakke lijn, omdat je met vier dicht bij elkaar staande pixels fantastisch scherp kunt weergeven. Alleen is de lijn helaas nauwelijks te zien, omdat hij in al z'n pracht maar 1 px breed is.
Ook niet echt geweldig. Daarom geven ook deze apparaten de breedte weer, alsof de resolutie een standaarddichtheid heeft. Een hogeresolutiescherm, zoals een retina-scherm, jokt een beetje. Ook een venster van 2048 of 4096 px breed beweert 1024 px breed te zijn. Hierdoor ziet een lijn van 4 px breed er ook op zo'n scherm als 4 px breed uit, en niet als 1 of 2 px breed.
Die hogeresolutiedichtheid heeft wel zin, maar dan op een andere manier. Letters bijvoorbeeld worden al jaren getekend aan de hand van wiskundige formules, het zijn allang geen statische afbeeldingen meer. Die formules kunnen prima met die hogeresolutiedicht-heid uit de voeten: een ronding in een letter is in een hogere resolutie veel fijner dan in een lagere resolutie. Ook voor hoge kwaliteit afbeeldingen is een hogeresolutiedichtheid zinvol. Het enige waar het niet zinvol is, is bij het aangeven van een maat. Een lijn van 4 px breed moet 4 px breed zijn, en niet 1 px breed in superkwaliteit.
(Bij hogere resoluties wordt onderscheid gemaakt tussen een 'hardware-pixel' en een 'css-pixel'. De hardware-pixel is de echte fysieke pixel. De css-pixel is de pixel zoals die al tijden in gebruik is bij css. Meer hierover kun je vinden op de pagina met links onder Mobiele apparatuur → Theorie, links, forums, e.d.)

Er staat nog een tweede deel in de tag: `initial-scale=1`. Sommige mobiele apparaten zoomen een pagina gelijk in of uit. Ook weer in een poging behulpzaam te zijn. Ook dat is hier niet nodig, want de pagina past zich aan het apparaat aan. Er is ook een instructie om zoomen helemaal onmogelijk te maken, maar die gebruik ik niet. De bezoeker kan zelf nog gewoon zoomen, wat belangrijk is voor mensen die wat slechter zien.

```
<link rel="stylesheet" href="../103-css-dl/afbeelding-103-1-
    dl.css">
```

(Deze html hoort bij de eerste pagina. Bij de tweede pagina eindigt de naam van het bijbehorende stylesheet op ”103-2-dl.css”, enz.)

Dit is een koppeling naar een extern stylesheet (stijlbestand), waarin de css staat. In html5 is de toevoeging `type="text/css"` niet meer nodig, omdat dit standaard al zo staat ingesteld. Je moet uiteraard de naam van en het pad naar de stylesheet aanpassen aan de naam en plaats van je eigen stylesheet staan.

Voordeel van een externe stylesheet is o.a. dat deze geldig is voor alle pagina’s, waaraan deze is gelinkt. ’n Verandering in de lay-out hoef je dan maar in één enkele stylesheet aan te brengen, in plaats van in elke pagina apart. Op een grotere site kan dit ontzettend veel werk schelen. Bovendien hoeft de browser zo’n extern stylesheet maar één keer te downloaden, ongeacht hoeveel pagina’s er gebruik van maken. Zou je de css in elke pagina opnieuw aanbrengen, dan worden de te downloaden bestanden veel groter.

In dit voorbeeld heeft een extern eigenlijk stylesheet geen nut, omdat er maar één pagina is die dit stylesheet gebruikt. In dit geval kun je de css beter in de <head> van de html-pagina zelf zetten. Voor de omvang maakt het hier niets uit, want de css wordt hoe dan ook altijd precies één keer gedownload, en nooit vaker. Voor het onderhoud maakt het ook geen verschil, want ook hier hoef je de css maar op één plaats te wijzigen. Maar het scheelt wel een extra aanroep naar de server, omdat geen apart stylesheet hoeft te worden gedownload.

Dat opnemen in de <head> gaat heel simpel: je kopieert gewoon het hele stylesheet en zet die bovenin de <head>, tussen <style> en </style>:

```
<style>
    body {color: black;}
        (…) rest van de css (...)
    div {color: red;}
</style>
```

Maar zodra een stylesheet op meerdere pagina’s wordt gebruikt, wat meestal het geval zal zijn, is een extern stylesheet beter.

(De reden dat er toch externe stylesheets zijn, terwijl ik hierboven omstandig beweer dat dat in dit voorbeeld eigenlijk geen nut heeft: overzichtelijkheid. Nu kun je html en css los van elkaar bekijken.)

Het <video>-element

```
<video data-smscr="480" aria-describedby="titel-1" controls
    preload="metadata" poster="../103-images/s_brock_cellular_
        respiration.jpg">
    <source src="../103-video/s_brock_cellular_respiration.webm"
        type="video/webm">
    <source data-smscr="480" src="../103-video/s_brock_cellular_
        respiration_klein.webm" type="video/webm">
    <source src="../103-video/s_brock_cellular_respiration.mp4"
        type="video/mp4">
    <source data-smscr="480" src="../103-video/s_brock_cellular_
        respiration_klein.mp4" type="video/mp4">
    Je browser ondersteunt het afspelen van video's niet. Je kunt
        hieronder de video nog wel downloaden.
</video>
```

De hierboven staande code is de code, zoals die is te vinden op de pagina met html (afbeelding-103-1-dl.html). Dit is ook de code die je krijgt te zien, als je gewoon in de browser de broncode bekijkt. Als je de echter de gegenereerde code – de code die de browser echt gebruikt – bekijkt, zie je een iets andere code. In browservensters smaller of lager dan 480 px zie je 'n weer andere code.
De veranderingen worden aangebracht door het script. Hier wordt alleen bovenstaande code bekeken, de door het script aangepaste veranderingen staan hieronder bij Door het script aangebrachte wijzigingen in het <video>-element.
Bovenstaande code hoort bij de eerste videospeler. Andere videospelers hebben andere afbeeldingen en video's, en `aria-describedby` wijkt ook iets af.

Hier gelijk onder staat een korte omschrijving van de onderdelen uit bovenstaande code, iets lager staat de uitgebreide beschrijving van alle onderdelen.
`<video>` en `</video>`: begin- en eindtag. In de begintag staan enkele attributen die aangeven, hoe het element zich moet gedragen. Er zijn veel meer attributen, die verderop worden besproken.
`<source>`: binnen elke <source> staat een video. Zodra de browser een video vindt die kan worden afgespeeld, wordt niet verder gekeken naar volgende <source>'s. Bij dit vinden helpt het opgegeven MIME-type (`type`) van de video.

```
<video data-smscr="480" aria-describedby="titel-1" controls
    preload="metadata" poster="../103-images/s_brock_cellular_
    respiration.jpg">
```

`<video>` is gewoon de openingstag. Helemaal achteraan, na een hele reeks elementen die binnen <video> zitten, wordt het element afgesloten met </video>.

`data-smscr="480"`

Dit attribuut is een eigengemaakt, je zult het in geen enkele specificatie tegenkomen. In html5 kun je eigen attributen maken, die precies hetzelfde worden behandeld als bijvoorbeeld `width="300"` bij een afbeelding. Je kunt ze dus bijvoorbeeld ook in een selector gebruiken. De enige voorwaarde is dat de naam begint met `data-`.
Het wordt hier gebruikt om in browservensters smaller of lager dan 480 px de in de browser ingebouwde standaardvideospeler te gebruiken, en niet de door het script aangestuurde. Hoe het precies werkt, staat hieronder bij Door het script aangebrachte wijzigingen in het <video>-element.

`aria-describedby="titel-1"`

Dit is een zogenaamde WAI-ARIA-code. Deze is van belang voor schermlezers e.d. De video wordt op deze manier gekoppeld aan een element met id="titel-1". Dat is hier de <h2> boven de eerste video, waarin de titel van de video staat. Zonder deze koppeling zou een schermlezer (maar ook een zoekmachine) niet weten, wat dit voor video is. Meer hierover staat bij WAI-ARIA-codes binnen de videospeler.

`controls`

Geeft aan dat knoppen moeten worden weergegeven, waarmee de videospeler is te bedienen. De maker van de browser heeft voor deze knoppen gezorgd, dus ze verschillen enigszins per browser. Als je `controls` weglaat, worden de bedieningsknoppen niet getoond. Maar ook dan kan in de meeste browsers nog steeds met rechtsklikken een contextueel menu worden geopend, waarmee de video

toch kan worden afgespeeld. In sommige browsers echter kan de video nu helemaal niet worden afgespeeld, omdat die mogelijkheid in het contextueel menu ontbreekt. Op mobiele browsers kun je niet rechtsklikken, dus daar kan de video zonder bedieningsknoppen gewoon helemaal niet worden afgespeeld.
Het weglaten van `controls` heeft alleen maar zin, als je de video via JavaScript wilt laten afspelen, of via JavaScript eigen knoppen wilt laten weergeven. Dat is precies wat hier gebeurt, en een van de eerste dingen die het script doet is dan ook het verwijderen van `controls`, zoals verder wordt beschreven bij Door het script aangebrachte wijzigingen in het <video>-element.

`preload="metadata"`

Hiermee regel je het downloaden van de video: zodra de pagina wordt geopend, of pas als de video daadwerkelijk wordt afgespeeld. Dit is niet meer dan een hint voor de browser, de browser kan afwijken van wat hier wordt opgegeven. Ook al geef je bijvoorbeeld op dat de video gelijk bij openen van de pagina moet worden gedownload, dan zullen de meeste mobiele browsers dit toch pas gaan doen, als de gebruiker kiest voor afspelen.
Mogelijke waarden:

- `none`: pas downloaden als wordt afgespeeld.
- `metadata`: alleen metagegevens zoals breedte, hoogte en afspeelduur ophalen.
- `auto`: video gelijk downloaden, al voordat wordt gekozen voor afspelen. Als je niets invult bij `preload`, is dit de standaardwaarde.

Hier wordt als waarde 'metadata' gebruikt, zodat de speelduur van de video gelijk bij openen van de pagina wordt opgevraagd. Daardoor kan deze in de videospeler worden ingevuld. Bovendien werkt een aantal functies van de speler pas, als de lengte van de video bekend is.
(Als je zo'n functie – zoals Tien seconden vooruit gaan – wilt gebruiken, voordat de speelduur bekend is, levert dat een waarschuwingsvenstertje op. Meer daarover bij Door het script gegenereerde waarschuwingen (alerts, pop-ups).
In Internet Explorer 9 zit een eigenaardigheid, waardoor `poster` (zie gelijk hieronder) niet werkt, als je als waarde iets anders dan 'none' gebruikt. Safari op iOS negeert (of negeerde tot voor heel kort) `preload` volledig. Meer over deze twee dingen bij Bekende problemen (en oplossingen).

`poster="../103-images/s_brock_cellular_respiration.jpg"`

Achter `poster` kun je de url van een afbeelding opgeven, die wordt getoond voordat de video wordt afgespeeld.
Als je `poster` weglaat, wordt op de desktop het eerste frame uit de video getoond, maar de meeste mobiele browsers tonen dan alleen een zwart schermpje met knoppen.
In Internet Explorer 9 zit een eigenaardigheid, waardoor `poster` alleen werkt als bij `preload` 'none' is ingevuld. Zie verder bij Bekende problemen (en oplossingen).

`autoplay`

Wordt niet gebruikt in dit voorbeeld. Zorgt ervoor dat de video automatisch begint af te spelen. Heel fijn als je toevallig in een gehorig huis woont naast een beroepsbokser met 'n kort lontje. Vooral als je vaak 's nachts surft en op de betreffende video luidkeels het Wilhelmus wordt gezongen door het voltallige Nederlandse elftal. Na

de revolutie komt hier drie maanden zonnepanelen aanbrengen op een verlaten olieplatform op te staan.
Serieus: dit is een prima manier om bezoekers weg te jagen. Laat mensen zelf kiezen of ze wel of niet je video willen afspelen. Dit geldt nog sterker, als de video ook geluid heeft. In dat geval zou je op z'n minst ook `muted` moeten opgeven (zie hieronder).

`loop`

Wordt niet gebruikt in dit voorbeeld. Als de video is uitgespeeld, wordt weer van voren af aan begonnen.

`height`

Wordt niet gebruikt in dit voorbeeld. De hoogte van de video in pixel. Zodra ook een hoogte in de css wordt opgegeven, overrulet deze de hier opgegeven hoogte.
In dit voorbeeld wordt de hoogte van de video opgegeven in de css.

`width`

De breedte van de video in pixel. Zodra ook een breedte in de css wordt opgegeven, overrulet deze de hier opgegeven breedte.
In dit voorbeeld wordt de breedte van de video opgegeven in de css.

`muted`

Wordt niet gebruikt in de html van dit voorbeeld. Zet het geluid uit. De browser is niet verplicht hiernaar te luisteren.
(Normaal genomen zal wel worden geluisterd, als het geluid is uitgezet. Als het geluid aanstaat, zal niet altijd door de browser worden gehoorzaamd. Mocht je partner als onderdeel van een knetterende ruzie een scheldpartij aan je hebben gestuurd, dan wordt daardoor voorkomen dat deze onbedoeld tijdens de directievergadering wordt afgespeeld.)
In dit voorbeeld wordt `muted` in- en uitgeschakeld door middel van een knop, die wordt afgehandeld door het script.

`mediagroup`

Wordt niet gebruikt in dit voorbeeld. Wordt gebruikt om twee of meer video's gelijktijdig af te laten spelen. Dit maakt het bijvoorbeeld mogelijk om een video te voorzien van een doventolk. De video's worden met dezelfde knoppen bediend.
Achter `mediagroup` wordt de naam van de groep ingevuld: `mediagroup="video-1"`. Elke video komt in een apart <video>-element te staan.

`src`

Hierin komt de url van de af te spelen video. Dit wordt alleen gebruikt, als er maar één video is opgegeven. Zodra er meerdere video's zijn, waaruit de browser moet kiezen, komt elke video in een aparte <source> te staan binnen <video>. In de code hierboven staat dan ook geen `src` in het de <video>-tag.
(Als aan bepaalde voorwaarden wordt voldaan, kan wel `src` worden toegevoegd door het script. Meer daarover hieronder bij Door het script aangebrachte wijzigingen in het <video>-element.)

```
<source src="../103-video/s_brock_cellular_respiration.webm"
     type="video/webm">
<source data-smscr="480" src="../103-video/s_brock_cellular_
     respiration_klein.webm" type="video/webm">
<source src="../103-video/s_brock_cellular_respiration.mp4"
     type="video/mp4">
<source data-smscr="480" src="../103-video/s_brock_cellular_
     respiration_klein.mp4" type="video/mp4">
```

Binnen het <video>-element kun je meerdere video's opgeven. De browser bekijkt alle video's. Zodra er eentje wordt gevonden die kan worden afgespeeld, wordt die gebruikt. De rest van de <source>'s wordt niet meer bekeken.

`src`

Hierachter staan pad en naam van de video.

`type`

Dit is een zogenaamd MIME-type. Het deel voor de schuine streep geeft een soort grove indeling, hier 'video'. Het tweede deel geeft het formaat weer. Met de twee formaten `webm` en `mp4` kun je alle browsers bedienen. Helaas konden de browsermakers het niet eens worden over één formaat, anders had je – wat het formaat betreft – met één <source> kunnen volstaan.
In principe kun je zoveel <source>'s gebruiken, als je wilt. Als je dus om een of andere reden speciaal voor Internet Explorer een exotisch formaat wilt gebruiken, zet je dat gewoon vooraan. Browsers die dat exotische formaat niet kennen, negeren het gewoon en gebruiken een latere <source>.

`data-smscr="480"`

Dit attribuut is een eigengemaakt, je zult het in geen enkele specificatie tegenkomen. In html5 kun je eigen attributen maken, die precies hetzelfde worden behandeld als bijvoorbeeld `width="300"` bij een afbeelding. Je kunt ze dus bijvoorbeeld ook in een selector gebruiken. De enige voorwaarde is dat de naam begint met `data-`.
Dit attribuut is in twee <source>'s aanwezig. Dit zijn video's voor kleinere browservensters. Als er een <source> met dit attribuut aanwezig is, zal de video uit deze <source> worden gebruikt in browservensters smaller of lager dan – in dit geval – 480 px. Hoe dit precies werkt, staat hieronder bij Door het script aangebrachte wijzigingen in het <video>-element.

```
Je browser ondersteunt het afspelen van video's niet. Je kunt
     hieronder de video nog wel downloaden.
```

Tenslotte staat er nog een tekst in het <video>-element. Deze tekst wordt getoond in browsers die het <video>-element niet kennen. In dit geval verwijst de tekst naar de download-links die onder elke video zijn aangebracht.

Door het script aangebrachte wijzigingen in het <video>-element

De hieronder beschreven veranderingen zie je niet, als je gewoon de bron van de pagina bekijkt. Dan zie je gewoon de html zoals die in de pagina staat. Om de door het script gewijzigde html te zien, moet je de gegenereerde code bekijken.

Als dat niet met behulp van css is verborgen, kun je bovenaan de pagina kiezen voor de in de browser ingebouwde standaard- of voor de door het script aangestuurde videospeler. Als

die keuze is verborgen, wordt altijd de door het script aangestuurde speler gebruikt.
Feitelijk is het niet helemaal juist, wat ik hierboven zeg. Alleen het script wordt uitgevoerd. Of de aangepaste videospeler daadwerkelijk wordt gebruikt, is nog helemaal niet zeker.
Het script kijkt als eerste of er in een van de <video>-elementen een attribuut met de naam 'data-smscr' voorkomt. Zodra dat wordt gevonden, wordt gekeken of de waarde daarvan een positief getal is. Als dat zo is, wordt niet verder gezocht in andere <video>-elementen.
In het voorbeeld staat in elke <video> `data-smscr="480"`, waarmee hieraan dus is voldaan.

Als geen `data-smscr` is opgegeven, of als de waarde daarvan geen positief getal is, wordt de rest van het script vervolgens gewoon uitgevoerd: de aangepaste videospeler wordt gebruikt.
In de html wordt door het script het attribuut `controls` verwijderd, waardoor de bedieningselementen van de standaardvideospeler worden verborgen. Als je dat niet zou doen, zouden er twee groepen bedieningselementen zijn: eentje van de standaardvideospeler en eentje van de door het script aangestuurde speler.
Als later wordt gekozen om toch de standaardvideospeler te gebruiken, wordt `controls` weer teruggezet, maar in iets aangepaste vorm: `controls="controls"`. Dit is eigenlijk, simpel gezegd, de xhtml-vorm van `controls`. Maar alleen `controls` terugzetten werkt niet in alle browsers, vandaar. (xhtml is een soort heel strenge variant van html, die niet verder meer wordt ontwikkeld.)

Het media-attribuut
Dit hele ingewikkelde verhaal hiernaast was tot voor kort niet nodig. Je kon in <video> het `media`-attribuut gebruiken, dat precies hetzelfde werkte als in een `@media`-regel. Dit was ook al in alle nieuwere browsers geïmplementeerd.
Om volstrekt onduidelijke redenen is dit, vlak voor de html5-specificatie standaard werd, eruit gehaald. Terwijl gelijktijdig ongeveer hetzelfde attribuut werd toegevoegd aan <picture>.
Met het `media`-attribuut kon je op uiterst simpele wijze de af te spelen video koppelen aan dingen als schermgrootte. Dat kan nu dus nog alleen op deze uiterst omslachtige wijze met gebruik van JavaScript.

Als `data-smscr` mist, of geen positief getal als waarde heeft, vindt geen aanpassing van het <video>-element plaats.

In het voorbeeld staat in elk <video>-element `data-smscr="480"`. Als je een ander getal invult, blijft de werking hetzelfde. Alleen verandert de grens, waarbij die werking optreedt. Als de breedte én de hoogte van het browservenster meer zijn dan 480 px, wordt het script uitgevoerd en wordt de door het script aangestuurde videospeler gebruikt.

Als het venster lager of smaller is dan 480 px, wordt het script niet uitgevoerd, maar wordt de in de browser ingebouwde standaardvideospeler gebruikt. Dat is ook veel beter, want op zo'n klein scherm is gewoon geen ruimte voor een uitgebreide serie bedieningselementen. En de standaardvideospeler is, als het goed is, precies aangepast aan de betreffende smartphone e.d.
Een klein stukje van het script wordt echter wel altijd uitgevoerd, ook op kleine schermen: er wordt nogmaals bij elke <video> gekeken of daar een `data-smsrc`-attribuut met een positief getal aanwezig is. En deze keer wordt wel élke <video> bekeken. Als er geen `data-smscr` aanwezig is in het <video>-element, wordt er verder niets met de betreffende <video> gedaan, maar wordt naar de volgende <video> gegaan. Het simpele weglaten van

een `data-smscr` bij <video> zorgt ervoor, dat wat hieronder staat bij deze <video> niet gebeurt.
Als bij een <video>-element wel een `data-smscr` met een positief getal staat, worden van dat <video>-element alle <source>'s ingelezen. Dat zijn er in het voorbeeld bij elke <video> vier.
Vervolgens wordt bij elke <source> gekeken, of daar een `data-smscr` met een positief getal in staat. In het voorbeeld staat in elke tweede <source> `data-smscrc="480"`. Bij deze <source>'s wordt vervolgens gekeken, of de video die binnen die <source> in de `src` staat, afgespeeld kan worden door de videospeler. Als dat zo is, wordt niet verder gezocht. Als de video niet kan worden afgespeeld, wordt verdergegaan met de volgende <source>. Als de video afgespeeld kan worden, wordt aan het <video>-element een `src` toegevoegd met daarin de naam van de gevonden video. Als je in een browservenster smaller of lager dan 480 px in Firefox de gegenereerde code bekijkt, staat daarin:

```
<video poster="../103-images/s_brock_cellular_
     respiration.jpg" preload="metadata" controls=""
     aria-describedby="titel-1" data-smscr="480"
     src="http://www/css-voorbeelden.nl/afbeelding/
     video/103-video/s_brock_cellular_respiration_
     klein.webm">
```

Te zien is dat `controls=""` is toegevoegd aan <video>, waardoor de ingebouwde standaardvideospeler wordt gebruikt.
Daarnaast is toegevoegd een `src`, waarin de af te spelen video staat. Omdat een `src` aanwezig is, worden de ook nog steeds aanwezige <source>'s verder niet bekeken door de browser, maar wordt de in het `src`-attribuut staande video afgespeeld.

In het voorbeeld staan in de <source>'s met `data-smscr="480"` video's die kleiner zijn dan die in de <source>'s zonder `data-smscr="480"`. Dit geeft de mogelijkheid om op kleinere schermpjes kleinere video's af te spelen. Een smartphone heeft niets aan een video voor een breedbeeldscherm, dus die kan op deze manier een veel kleinere video downloaden.

Als er in een <video> of in geen van de <source>'s binnen die <video> een `data-smscr` met een positief getal staat, wordt dit hele stuk overgeslagen en wordt gewoon de eerste video die afspeelbaar is gebruikt. Dat is in het voorbeeld dus de grote video.

De download-links <a lang="en" href="..." download="..." title="...">

Onder elke video staan links om de video te kunnen downloaden. Deze links zijn aangebracht voor het geval dat de browser de video niet af kan spelen, zoals Opera Mini. Je kunt dan in ieder geval de video downloaden en op 'n andere manier afspelen.
De volledige code voor video's in groot formaat onder de eerste video:

```
<p class="groot">Video downloaden? Kies <a lang="en"
     href="../103-video/s_brock_cellular_respiration.mp4"
     download="S. Brock - Cellular Respiration - B
     Block.mp4" title="Download video in mp4-
     formaat">mp4</a> <span>(12,4 <abbr
     title="megabyte">MB</abbr>)</span> of <a lang="en"
     href="../103-
     video/s_brock_cellular_respiration.webm"
```

```
download="S. Brock - Cellular Respiration - B
Block.webm" title="Download video in webm-
formaat">webm</a> <span>(21,6
<abbr>MB</abbr>)</span>.</p>
```

Hier staan twee links: eentje voor de video in mp4-formaat, en eentje voor de video in webm-formaat. De bezoeker kan hierdoor kiezen welk formaat wordt gedownload. Als je maar één formaat opgeeft, is er een kans dat de gedownloade video niet kan worden afgespeeld. Door een class aan de <p> waar de links in staan te geven, kun je met css deze link verbergen. Dat gebeurt in dit voorbeeld in browservensters smaller of lager dan 480 px. Daar wordt deze code verborgen, maar wordt het zusje hiervan getoond: vrijwel hetzelfde, maar met video's in kleiner formaat.

Achter de normale href van een <a> staat extra code:

```
download="S. Brock - Cellular Respiration - B Block.mp4"
```

Normaal genomen zou als naam voor de download 's_brock_cellular_respiration.mp4' worden gebruikt. Maar in browsers die `download` herkennen, wordt de naam nu verandert in het veel mensvriendelijker 'S. Brock – Cellular Respiration – B Block.mp4'.
De extensie 'mp4' hoeft niet te worden gebruikt achter `download`, maar het is beter om dat wel te doen. Niet alle browsers blijken deze extensie zelf toe te voegen, waardoor deze niet altijd achter de naam van de video komt te staan. Een van de grootste ondingen wat betreft veiligheid op Windows is het standaard verbergen van extensies, waardoor 'n gebruiker niet snel kan zien wat voor soort bestand iets is. Ik heb er geen enkele behoefte aan dit veiligheidsrisico naar andere systemen te exporteren, dus daarom gebruik ik een extensie achter `download`.

De grootte van de video staat in een . Door de grootte apart in een te zetten, ontstaat een aanknopingspunt voor css, waardoor je de grootte bijvoorbeeld in een kleinere letter kunt weergeven.
Omdat MB een afkorting is, staat deze in een <abbr>. De afkorting MB komt voor de eerste keer voor onder de eerste video, daarom staat bij de eerste <abbr> de afkorting voluit geschreven:

```
<abbr title="megabyte">MB</abbr>
```

Afhankelijk van de (instellingen van de) browser begint deze bij activeren van deze link gelijk met downloaden, of wordt eerst om bevestiging gevraagd. Mogelijk wordt niet iedereen gelijk dol van vreugde, als een browser zonder eerst om bevestiging te vragen een video van 300 gigabyte begint te downloaden. Daarom is achter de link de grootte van de download aangegeven.

Als je wilt uitproberen, hoe deze download-link in de diverse browsers op de diverse systemen werkt, moet je dat op een server uitproberen. Firefox bijvoorbeeld speelt de video gewoon af, als je het lokaal probeert. Maar als je het probeert op 'n server, wordt de video in datzelfde Firefox wel gedownload.

Links naar de scripts

```
<script src="../103-js/afbeelding-103.js"></script>
<script src="../103-js/hand.minified-1.3.8.js"></script>
```

Onderaan de html staan twee links naar scripts. De bovenste is de link naar het script dat de videospelers aanstuurt. De onderste is een link naar een script van Microsoft, dat muis, aanraking, e.d. stuurt.

In html5 is de toevoeging `type="text/javascript"` niet meer nodig, omdat dit de standaardinstelling is.
De links naar de scripts staan helemaal onderaan de pagina, om te voorkomen dat de browser staat te wachten tijdens het verwerken van de scripts. Nu kan eerst de rest van de pagina worden geladen, voordat de scripts worden verwerkt.

CSS

Anders dan in andere voorbeelden, wordt veel css hier alleen maar heel summier beschreven. Alleen css die enigszins apart of ingewikkeld is, wordt uitgebreider beschreven. Het gaat om vijf aparte stylesheets, één voor elke pagina met videospelers, die alle vijf op zich al tamelijk fors zijn. De papieren telefoonboeken zijn net min of meer afgeschaft, en het lijkt me geen goed idee om hier te beginnen met verhalen met de omvang van een telefoonboek. Bovendien is 'n telefoonboek ook wat saai om te lezen.
Ook anders dan in andere voorbeelden: de essentiële css is niet rood gekleurd, omdat in feite álle css essentieel is (of helemaal niets, afhankelijk van hoe je het bekijkt).

Technisch gezien is er geen enkel bezwaar om de css in de stylesheet allemaal achter elkaar op één regel te zetten:

```
div#header-buiten {position: absolute; right: 16px;
width: 100%; height: 120px; background: yellow;}
div#header-binnen {margin-left: 16px; height: 120px;
text-align: center;}
```

Maar als je dat doet, garandeer ik je hele grote problemen, omdat het volstrekt onoverzichtelijk is. Beter is het om de css netjes in te laten springen:

```
div#header-buiten {
    position: absolute;
    right: 16px;
    width: 100%;
    height: 120px;
    background: yellow;
}

div#header-binnen {
    margin-left: 16px;
    height: 120px;
    text-align: center;
}
```

Hiernaast is het heel belangrijk voldoende commentaar (uitleg) in de stylesheet te schrijven. Nu weet je waarschijnlijk (hopelijk...), waarom je iets doet. Maar over vijf jaar kan dat volstrekt onduidelijk zijn. Op deze site vind je nauwelijks commentaar in de stylesheets, maar dat heeft een simpele reden: deze uitleg is in feite een groot commentaar.
Op internet zelf is het goed, als de stylesheet juist zo klein mogelijk is. Dus voor het uploaden kun je normaal genomen het beste het commentaar weer verwijderen. Veel mensen halen zelfs alles wat overbodig is weg, voordat ze de stylesheet uploaden. Inspringingen bijvoorbeeld zijn voor mensen handig, een computer heeft ze niet nodig.
Je hebt dan eigenlijk twee stylesheets: eentje waarin je dingen uitprobeert, verandert, enz., met commentaar, inspringingen, e.d. Dat is de mensvriendelijke versie. Daarnaast is er dan een stylesheet die je op de echte site gebruikt: een gecomprimeerde versie.

Dat comprimeren kun je met de hand doen, maar er bestaan ook hulpmiddelen voor. Als je op internet zoekt naar 'css' en 'compress' of 'comprimeren', vind je tal van sites, waar je dat automatisch kunt doen.
(Stylesheets op deze site zijn niet gecomprimeerd. Omdat het vaak juist om de css gaat, wil ik dat mensen zonder al te veel moeite de css kunnen bekijken.)

Pagina 1

De css die hier wordt beschreven, hoort bij de stylesheet 'afbeelding-103-1-dl.css'. Deze stylesheet hoort bij de eerste pagina met videospelers.

Op de eerste pagina zien alle videospelers er hetzelfde uit.

css voor alle breedtes

```
@-webkit-keyframes bugfix {from {padding-left: 0;} to {padding-
    left: 0;}}
```

Op een aantal plaatsen wordt een animatie uitgevoerd met behulp van `animation`. In oudere versies van Android browser zit een bug, waardoor deze animaties niet (goed) werken, als in de selector ~ of + zit. Deze nep-animatie repareert die bug. Een nep-animatie, want de padding-left wordt van 0 naar 0 veranderd, dus er gebeurt feitelijk niets. Toch neutraliseert deze flauwekul-animatie de bug in Android.
Normaal genomen is het een bijzonder slecht idee css te gebruiken voor slechts één weergave-machine: webkit. Maar in dit geval is dit terecht, want de bug zit alleen in Android browser, en die gebruikt webkit. Het heeft dus geen zin om Internet Explorer of Firefox ook met deze ongein te belasten.
Waarom dit alleen in webkit-browsers werkt, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

`body`

Het element waarbinnen de hele pagina staat. Veel instellingen die hier worden opgegeven, worden geërfd door de nakomelingen van <body>. Ze gelden voor de hele pagina, tenzij ze later worden gewijzigd. Dit geldt bijvoorbeeld voor de lettersoort, de lettergrootte en de voorgrondkleur.

```
background: #ff9; color: black; font-family: Arial, Helvetica,
    sans-serif; margin: 0; padding: 0;
```

Achtergrond geel, voorgrond zwart, lettersoort, marge en padding verwijderen.

Weinig interessant allemaal. Wat wel interessant is: in Safari op OS X zit een bug. Om die te repareren, wordt bij het kiezen voor standaard- of aangepaste videospeler door het script een inline-style toegevoegd aan <body>:

`style="zoom: 1.0001"` of `style="zoom: 1"`

Dit heeft verder geen enkele merkbare invloed, behalve dat het die bug repareert. Mocht je toch meer willen weten, dan is dat te vinden bij document.body.setAttribute(”style”, ”zoom: 1.0001”);.

Deze door het script ingevoegde css is niet te zien in de gewone broncode, maar alleen in de gegenereerde code.

Deze door het script ingevoegde inline-style overschrijft eventueel al bij <body> bestaande inline-styles. Het is altijd al een bijzonder slechte gewoonten om inline-styles te gebruiken, maar in dit geval moet dat dus zeker niet bij <body> gebeuren, omdat die inline-style gewoon wordt overschreven, zodra je kiest voor standaard- of aangepaste videospeler.

```
abbr[title] {cursor: help; border-bottom: dotted 1px;}
```

<abbr>’s die een `title`-attribuut hebben aan de onderkant een stippellijntje geven en de cursor in een vraagteken veranderen.

`main`

Het element waar vrijwel de hele pagina in staat.

```
max-width: 1500px; margin: 0 auto;
```

In hele brede browservensters voorkomen dat de pagina te breed wordt. Omdat bij margin geen waarden voor onder en links zijn ingevuld, krijgen die automatisch dezelfde waarde als boven en rechts. Hier staat dus eigenlijk `margin: 0 auto 0 auto;` in de volgorde boven – rechts – onder – links. Boven en onder geen marge, links en rechts `auto`, wat hier betekent: evenveel. De pagina staat hierdoor altijd horizontaal gecentreerd, ongeacht de breedte van het venster van de browser.

`h1`

Alle <h1>’s.

```
font-size: 1.2em; text-align: center; margin: 3px 0;
```

Lettergrootte iets verkleinen, centreren, marge aanpassen.

`#links`

Het element met id=”links”. De links naar vorige en volgende pagina staan in een <nav id=”links”>, zodat een schermlezer gelijk ziet dat het hier om links gaat die met navigeren te maken hebben.

```
text-align: center; margin: 3px 0;
```

Inhoud centreren, marge aanpassen.

```
#links h2 {position: absolute; left: -10000px;}
```

De <h2>’s binnen het element met id=”links”. Dat is hier maar één <h2>.

De links naar vorige en volgende pagina met videospelers staan in een <nav id=”links”>, zodat schermlezers weten dat het hier om navigatie tussen pagina’s gaat. Maar ze weten niet om wélke navigatie het gaat. Daarom is binnen de <nav> een <h2> gezet met als tekst ’Navigatie door voorbeeld’. Deze heeft echter alleen maar nut voor schermlezers. Daarom wordt hij ver links buiten het scherm geparkeerd. Nu is de <h2> onzichtbaar, maar wordt wel gewoon voorgelezen door een schermlezer. (En een zoekmachine weet nu ook, waar de links voor zijn.)

Zou je de <h2> met `display: none;` of `visibility: hidden;` wegmoffelen, dan wordt hij volledig genegeerd door schermlezers, dus dat werkt niet. Maar op deze manier is hij ook onzichtbaar.

`#links + p`

De <p> die gelijk volgt op het element met id=”links”. Dat is de <p>, waarin een korte omschrijving van de videospelers op de betreffende pagina staat.

`display: none;`

Verberg de <p>. Omdat in browservensters smaller of lager dan 480 px de ingebouwde standaardspeler wordt gebruikt, heeft die melding daar geen zin. Verderop wordt hij weer zichtbaar gemaakt voor grotere vensters.

`#wrapper-help {display: none;}`

Het element met id=”wrapper-help”. Dat is de <aside>, waarbinnen de pop-up met uitleg staat. Hele <aside> verbergen. Voor grotere browservensters wordt die straks weer tevoorschijn getoverd.

`.videobox {margin-top: 20px;}`

De elementen met class=”videobox”. Elke video met bijbehorende titels, links, e.d. staat binnen een met class=”videobox”.
Binnen elk element met class=”videobox” móét een <video>-element staan. Als dat niet zo is, loopt het script vast. In dat geval krijg je een waarschuwing in de vorm van een JavaScript-alert en wordt het betreffende element met behulp van een inline-style rood omlijnd:

```
style="border-color: red; border-width: 3px; border-
    style: solid;"
```

Omdat deze inline-style door het script wordt ingevoegd, is deze alleen in de gegenereerde code te zien. Normaal genomen zul je dit echter niet tegenkomen. Meer over deze waarschuwing bij Er is een element met class=”videobox”, waarin geen <video> staat.

```
h2, h3, .origineel, .groot, .klein {background: white; color:
    black; max-width: 294px; text-align: center; margin: 0 auto;
    border: black solid; border-width: 0 1px 1px; border-radius: 0
    0 5px 5px; padding: 3px;}
```

Alle <h2>’s, <h3>’s, de elementen met class=”origineel”, class=”groot” en class=”klein”. Wat standaardinstellingen voor kopjes, link naar de originele video, en links om de grote of kleine video te downloaden.
<video> krijgt hieronder een breedte van 300 px (dat is de breedte van de kleine video’s), deze elementen worden maximaal 294 px breed (met padding e.d. is dat even breed als <video>). Voor grotere browservensters wordt dat later, samen met de meeste eigenschappen die hier worden opgegeven, weer aangepast.
Voor sommige van deze elementen staat hieronder nog aanvullende css.

```
h2, h3 {font-size: 1em; font-weight: normal; border-width: 1px
    1px 0; border-radius: 5px 5px 0 0; padding: 3px 3px 0;}
```

Alle <h2>’s en <h3>’s. Hier gelijk boven is ook al wat css aan deze elementen gegeven. Nog wat extra instellingen voor de kopjes. Deze worden later voor grotere browservensters weer grotendeels veranderd.

`.origineel {font-size: 0.85em; border-width: 0 1px; border-radius: 0; padding-top: 0;}`

De elementen met class=”origineel”. Dit zijn de <p>’s waarbinnen de link naar de originele video’s staat. Iets hierboven bij `h2, h3, .origineel, .groot, .klein` is ook al wat css aan deze elementen gegeven.

Wat instellingen om het iets te verfraaien en zo. Voor grotere browservensters wordt deze css later grotendeels weer aangepast.

`video`

Alle <video>’s. De elementen waar de eigenlijk video in staat.

`background: white; color: black; display: block; width: 300px; max-width: 100%; margin: 0 auto; border: black solid 1px;}`

Wat simpele instellingen, voornamelijk om te zorgen dat de video’s op kleine schermen niet buiten het browservenster komen te staan.

`.groot span, .klein span`

De ’s binnen elementen met een class=”groot” of class=”klein”. In deze ’s zit de grootte van de download. Omdat alleen deze ’s in `.groot` en `.klein` zitten, is verder geen class of zo nodig voor de ’s.

`font-size: 0.6em; letter-spacing: -0.05em; position: relative; top: -2px;`

Tekst iets op elkaar proppen, zodat het niet te groot is in kleine browservensters. Voor grotere vensters wordt dit later grotendeels weer aangepast.

`.groot {display: none;}`

De elementen met class=”groot”. Dit zijn de <p>’s, waarbinnen de download-links naar de grotere video’s staan. Download-links voor de kleinere video’s staan in een met class=”klein”.

De download-links naar de grotere video’s worden hier verborgen. Verderop wordt het voor grotere browservensters precies omgedraaid, daar worden de download-links naar de kleinere video’s verborgen.

css voor vensters breder en hoger dan 480 px

`@media screen and (min-width: 480px)and (min-height: 480px)`

De css die hier tot nu toe staat, geldt voor alle browservensters. De css die hieronder staat, geldt alleen voor vensters breder én hoger dan 480 px. Voor een deel is dit nieuwe css, voor een deel wordt hierboven staande css aangepast.

Browsers die dit niet begrijpen, gebruiken de hieronder staande css niet, maar alleen wat hierboven staat. Misschien is het resultaat daarvan niet mooi, maar het werkt in ieder geval.

`@media`: geeft aan dat het om css gaat die alleen van toepassing is, als aan bepaalde voorwaarden wordt voldaan. Al langer bestond de mogelijkheid om met behulp van zo’n regel css voor bijvoorbeeld printers op te geven. css3 heeft dat uitgebreid tot bepaalde fysieke eigenschappen, zoals de breedte en hoogte van het venster van de browser.

`screen`: deze regel geldt alleen voor schermweergave. Als je wilt printen, is het beter een stylesheet daarvoor te baseren op de algemene css die hierboven staat.

`and`: er komen nog meer voorwaarden, waaraan moet worden voldaan.

`(min-width: 480px)`: het browservenster moet minstens 480 px breed zijn. Is het venster smaller, dan wordt de css die binnen deze media-regel staat, genegeerd.

`(min-height: 480px)`: het browservenster moet minstens 480 px hoog zijn. Is het venster lager, dan wordt de css die binnen deze media-regel staat, genegeerd.
Het venster van de browser moet dus breder én hoger zijn dan 480 px. Meestal is het voldoende alleen op de breedte te testen. Maar dan doet zich in dit geval een probleem voor in Windows Phone.
In het script wordt op de breedte van het browservenster getest. (Je kunt met behulp van data-smscr opgeven of er getest moet worden, en op welke breedte.) Beneden een bepaalde breedte wordt dan de standaardvideospeler gebruikt, en niet de aangepaste. In Windows Phone blijkt dan echter altíjd de aangepaste videospeler te worden gebruikt, hoe smal het venster ook is. Door in het script ook op de hoogte van het venster te testen, wordt dit opgelost.

Om te zorgen dat de overschakeling van css voor vensters breder en hoger dan 480 px gelijk loopt met de overschakeling in het script, en omdat in het script noodgedwongen op breedte én hoogte wordt getest, doe ik dat hier ook. Dan loop je niet het risico dat net rondom de 480 px de aangepaste videospeler voor grotere vensters wordt gebruikt, maar de css voor smallere vensters. Of omgekeerd.
Gelijk na deze regel komt een `{` te staan, en aan het einde van de css die binnen deze regel valt een bijbehorende afsluitende `}`. Die zijn in de regel hierboven weggevallen, maar het geheel ziet er zo uit:

```
@media screen and (min-width: 480px) and (min-height:
    480px; {
    body {color: silver;}
        (...) rest van de css voor deze @media-regel (...)
    footer {color: gold;}
}
```

Voor de eerste css binnen deze media-regel staat dus een extra `{`, en aan het eind staat een extra `}`.

```
#links + p {background: white; color: black; display: block;
    width: 734px; max-width: 80%; font-size: 0.85em; margin: 0
    auto; border: black solid 1px; border-radius: 5px; padding:
    3px 4px;}
```

De <p> die gelijk volgt op het element met id=”links”.
In deze <p> staat een kort overzicht van bijzonderheden van de videospelers op de betreffende pagina. Voor kleinere browservensters is de <p> eerder onzichtbaar gemaakt, hier wordt hij weer getoond. Verder wat gewone opmaak als randjes. Weinig spannend allemaal.

Pop-up met uitleg

```
#wrapper-help {display: block; width: 50px; position: absolute;
    top: 5px; left: 5px;}
```

Het element met id=”wrapper-help”. Linksboven staat een vraagteken, waaronder een pop-up met uitleg is verborgen. Dat staat allemaal in een <aside> met id=”wrapper-help”, die hier op de juiste plaats wordt neergezet.

```
#checkbox-help {visibility: hidden;}
```

Het element met id=”checkbox-help”. Dit is een <input type=”checkbox”>. Als deze checkbox is aangevinkt, blijft de uitleg geopend. Tot het vinkje weer wordt weggehaald.

Nou is zo'n vinkje niet echt moeders mooiste, en voor de werking hoeft het ook niet zichtbaar te zijn. Daarom wordt het onzichtbaar gemaakt. Dat kan niet met `display: none;`, omdat de checkbox dan in sommige browsers niet meer werkt. Met `visibility: hidden;` is de checkbox gewoon aanwezig, alleen zie je het ding niet.

`#helptext`

Het element met id="helptext". Dit is de , waarin de eigenlijke uitleg staat.

`-webkit-animation: bugfix infinite 1s;`

In oudere versies van Android browser zit een bug, waardoor de uitleg nooit wordt geopend. Met behulp van deze regel wordt de uitleg toch geopend. Meer hierover bij @-webkit-keyframes bugfix.

`background: #eef; color: black; width: 700px; max-width: 90%; border: black solid 1px; border-radius: 8px;`

Wat css om het uiterlijk van de tekst met uitleg wat te verfraaien. De maximumbreedte zorgt ervoor dat er altijd 'n stukje van de onderliggende pagina zichtbaar blijft, ook in smallere browservensters.

`position: fixed;`

Hulptekst vast op het scherm zetten, zodat deze niet mee scrolt. Dit is handig in bredere browservensters, omdat je dan de uitleg geopend kunt hebben bij de videospeler waar je mee bezig bent.
Ja, handig, maar het levert wel problemen op.
De sluitknop (die op de plaats van het vraagteken verschijnt, als de uitleg wordt geopend), scrolt niet mee. De sluitknop mee laten scrollen zou geen goed idee zijn, want de uitleg kun je sluiten, die sluitknop niet, en die zou pontificaal boven die schitterende video kunnen gaan staan.
Of er juist onder kunnen verdwijnen, afhankelijk van de z-index en zo. Dit zou dan weer betekenen dat de uitleg is geopend, terwijl de knop om de uitleg te sluiten zich heeft verstopt onder de video. Bij gebruik van een toetsenbord is dat geen probleem, omdat de uitleg ook met sneltoets Control+8 (of iets soortgelijks) kan worden gesloten en geopend. Maar op een touchscreen zonder toetsenbord is het wel een probleem.
Daarom wordt verderop in browservensters smaller dan 850 px de positie van fixed verandert naar absoluut. Dan scrolt de uitleg gewoon mee met de rest van de pagina. In touchscreens met een venster breder dan 850 px blijft dit probleem echter bestaan. 'n Groot probleem zal het niet zijn, want de uitleg kan alleen worden geopend als het vraagteken zichtbaar is. Dus de op dezelfde plaats tevoorschijn komende sluitknop is ook per definitie zichtbaar. En scrollen zal niet zo snel gebeuren, omdat de uitleg een groot deel van het scherm bedekt.
Maar het is niet helemaal zoals het hoort. Op de site zelf is dit inmiddels iets verder ontwikkeld, waarbij deze problemen niet meer spelen. Omdat het hier niet om de uitleg gaat, ga ik dat hier niet meer aanpassen. Voor nieuwsgierigen: de inhoudsopgave op de pagina met de uitleg werkt op de nieuwe manier.
(Twee ander foutjes zijn op de site ook opgelost: in Firefox op Android moet je twee keer het vraagteken aanraken, voordat de uitleg opent. En als de uitleg is geopend door middel van klikken of aanraken, opent de uitleg bij hoveren over het vraagteken pas weer, als buiten het vraagteken is geklikt of aangeraakt.)

`left: -10000px;`

Ver links buiten het scherm parkeren. Pas als de uitleg getoond moet worden, wordt deze binnen het scherm gezet. Op deze manier staat de uitleg niet in de weg, maar schermlezers kunnen de uitleg gewoon voorlezen, wat niet zo zou zijn bij gebruik van `display: none;`.

`z-index: 100;`

Zorgt ervoor dat de uitleg niet onder andere elementen kan verdwijnen.

```
#helptext h2, h3 {background: #eef; color: black; max-width: none;
    font-weight: bold; margin: 0; border: none;}
#helptext h3 {text-align: left; border-top: black solid 1px;
    border-radius: 0; padding-left: 7px;}
#helptext p {text-indent: -8px; margin: 0 5px 3px 15px;}
```

De <h2>'s, <h3>'s en <p>'s binnen het element met id="helptext". De kopjes en tekst binnen de uitleg worden hier opgemaakt.

```
#icons {background: url(../103-images/sprites-help.png); display:
    block; width: 50px; height: 50px; margin-top: -22px;}
```

Het element met id="icons" is een <label> dat bij de checkbox voor de uitleg hoort. Als achtergrond-afbeelding wordt een zogenaamde 'sprite' gebruikt. Door drie verschillende afbeeldingen samen te voegen, worden twee aanroepen naar de server uitgespaard. Bovendien is er geen vertraging als een andere icon getoond moet worden, want alleen de achtergrondpositie hoeft te worden veranderd.

`#icons::after`

Met behulp van `::after` wordt bij het element met id="icons" een pseudo-element gemaakt. Met behulp hiervan kan, als de middelste icon wordt getoond, daarboven tekst worden gezet.

`content: "Open-\Alaten?\AKlik";`

De te tonen tekst. Binnen de tekst staat twee keer '\A'. Op deze manier wordt binnen dit soort teksten een nieuwe regel aangegeven. De '\A' wordt niet getoond in de uiteindelijke weergave. Nieuwe regels op deze manier aangeven werkt alleen, als ook `white-space: pre;` is opgegeven.

```
display: none; width: 50px; font-size: 0.7em; line-height:
    0.9em; text-align: center; white-space: pre; position:
    absolute; top: 17px;
```

De hele tekst wordt hier al opgemaakt. Hij wordt verborgen met `display: none;` en pas zichtbaar gemaakt. als dat nodig is.

```
#wrapper-help #focus-for-tab:focus ~ #checkbox-help:not(:checked)
    ~ #wrapper-icons #icons, #wrapper-help:hover:not(:focus)
    #checkbox-help:not(:checked) ~ #wrapper-icons #icons
    {background-position: 100px;}
```

Juist, ja. Wacht nog even met naar de whisky grijpen, want als je dit gaat ontleden, valt het (hopelijk) enigszins mee.

Voor de variatie begin ik met de eigenschap: `background-position: 100px;`. Verschuif de achtergrond-afbeelding 100 px naar links. Daardoor wordt de icon met de pin zichtbaar. Dat is dus de bedoeling van de ongein hierboven: toon een andere achtergrond.

De selector zelf valt in twee delen uiteen: de eerste selector voor de komma, de tweede erna. Eerst de eerste selector.

`#focus-for-tab`: een lege met id="focus-for-tab", die alleen via de Tab-toets kan worden bereikt. Omdat de helemaal leeg is, heeft deze geen enkele invloed op touchscreens. Ook klikken op of hoveren over de werkt niet,

omdat ook dat op een die verder helemaal leeg is niet werkt. Deze is puur bedoeld voor mensen die, om wat voor reden dan ook, de Tab-toets gebruiken om links, knoppen, e.d. langs te gaan. (Meer over de tabindex bij Tabindex.)
Omdat een normaal genomen niet door gebruik van de Tab-toets kan worden bezocht, heeft de in de html een `tabindex="0"` gekregen. Nu wordt, bij gebruik van de Tab-toets, de bezocht, net zoals een gewone link, knop, e.d.: de kan 'focus' krijgen. Daardoor kun je met behulp van css dingen laten gebeuren, als de tab focus heeft (of juist als deze geen focus heeft).
De bijbehorende html, met weglating van alles wat voor deze uitleg overbodig is:

```
<aside id="wrapper-help">
    <span id="focus-for-tab" tabindex="0"></span>
    <input id="checkbox-help" type="checkbox">
    <div id="wrapper-icons">
        <label id="icons" for="checkbox-
            help"></label>
    </div>
    <div id="helptext">
```

Verder met de eerste selector:

`#wrapper-help #focus-for-tab:focus`: als het element (de) met id="focus-for-tab", die binnen het element (de <aside>) met id="wrapper-help" ligt, focus heeft.

`~ #checkbox-help:not(:checked)`: dit stukje van de selector voorkomt alleen, dat er dingen door elkaar heen gaan lopen. Het teken ~ betekent hetzelfde als het bekendere +, alleen hoeven de elementen voor en na de ~ niet gelijk op elkaar volgend in de html te staan. Als `#checkbox-help` maar ergens na `#focus-for-tab` staat (en dezelfde ouder heeft), is het prima.

`:not(:checked)`: helaas is hiermee de ellende nog niet ten einde, er staat nog meer. `:checked` in een selector wil zeggen, dat de css alleen werkt als de checkbox, radioknop, of zoiets is aangevinkt. Maar omdat er hier `:not` voor staat, wordt dat precies omgedraaid: deze css werkt alleen, als de checkbox níét is aangevinkt.
Als de checkbox is aangevinkt, wordt de uitleg getoond en moet het sluitkruisje worden getoond, niet de icon met de pen. Deze selector zorgt juist voor het tonen van de pen.

`~ #wrapper-icons #icons`: het element (de <label>) met id="icons" dat binnen een element (de) met id="wrapper-icons" ligt. En `#wrapper-icons` moet dan in de html weer volgen op het element dat voor de ~ staat, hier `#checkbox-help`. Anders dan bij een + hoeven ze niet gelijk op elkaar te volgen in de html, als `#wrapper-icons` maar ergens na `#checkbox-help` staat en dezelfde ouder heeft.

De hele eerste selector in normale mensentaal: verander de positie van de achtergrond-afbeelding in het element met id="icons", dat binnen een element met id="wrapper-icons" ligt. `#wrapper-icons` moet volgen op `#checkbox-help`, dat niet aangevinkt mag zijn, en zelf weer moet volgen op een element met id="focus-for-tab", dat focus heeft en binnen een element met id="wrapper-help" zit.
Hmmm. Ik geef toe: als ik dat gewone mensentaal vind, wordt het misschien tijd dat ik weer 'ns wat vaker m'n hoofd uit de monitor trek en de kroeg in duik. Nog 'n poging. Als je met

de Tab-toets bij het vraagteken bent aangekomen, verander dan het vraagteken in het icoontje met de pen, maar alleen als de checkbox niet is aangevinkt.
Oef. Gered van de kroeg. Toch?

De tweede selector, die van na de komma:

```
#wrapper-help:hover:not(:focus) #checkbox-
    help:not(:checked) ~ #wrapper-icons #icons
```

Bij hoveren over het vraagteken moet de icoon met de pin worden getoond, dat is de theorie. Helaas is dat in de praktijk iets ingewikkelder.
Om te beginnen mag de icoon met de pin niet worden getoond als de uitleg is geopend, want dan moet het sluitkruisje worden getoond. Daar zorgt de `#checkbox-help:not(:checked)` weer voor, net als bij de eerste selector hierboven.
Omdat dit niet voor de Tab-toets is bedoeld, ontbreekt `#focus-for-tab`.
En dan begint het gedonder. Je zou moeten kunnen volstaan met `#wrapper-help:hover`: als over het element met id=”wrapper-help” wordt gehoverd. Want daar gaat het om. Helaas blijkt dat iOS hoveren hetzelfde behandelt als een aanraking. Als je op iOS het vraagteken aanraakt, wordt het icon met de pin getoond. En dat blijft getoond worden, totdat je iets anders aanraakt dat focus kan krijgen, iets als een knop of een link.
Door als extra voorwaarde op te nemen dat `#wrapper-help` geen focus mag hebben, als de icon met de pin wordt getoond, kun je dit oplossen. Het heeft verder geen invloed op normaal hoveren op de desktop, want dat staat los van focus.
Om op iOS op focus te kunnen testen, blijkt `#wrapper-help` een tabindex te moeten hebben. Toevallig staat er al een `tabindex="-1"`, omdat die nodig is voor een klein probleem op Android 4.4.2, zoals beschreven bij Tabindex. De volgorde van de Tab-toets wordt niet verstoord, want als een element een `tabindex="-1"` heeft, wordt dat element overgeslagen door de Tab-toets.

```
#wrapper-help #focus-for-tab:focus + #checkbox-help:not(:checked)
    ~ #wrapper-icons #icons::after, #wrapper-help:hover:not
    (:focus) #checkbox-help:not(:checked) ~ #wrapper-icons
    #icons::after {display: block;}
```

Deze selector is precies hetzelfde als die gelijk hierboven, maar dan voor `#icons::after` in plaats van gewoon voor `#icons`. Met behulp van `::after` wordt een pseudo-element gemaakt bij `#icons`, dat wordt gebruikt om tekst weer te geven boven het icoontje met de pin. Die tekst is bij #icons::after met `display: none;` verborgen en wordt nu zichtbaar gemaakt.

```
#focus-for-tab:focus ~ #helptext {left: 0;}
```

Het element met id=”focus-for-tab” is een lege , die alleen maar aanwezig is om de hulp te kunnen openen met behulp van de Tab-toets. Omdat de een `tabindex="0"` heeft, wordt hij gewoon bezocht bij gebruik van de Tab-toets, en kan focus krijgen.
Als deze focus heeft, zet dan het element met id=”helptext” (de uitleg) links in het browservenster. Deze uitleg is bij #helptext links buiten het scherm geparkeerd, maar wordt nu dus zichtbaar.

```
#checkbox-help:not(:checked) + #wrapper-icons #icons {background-
    position: 0;}
```

`background-position: 0;` zorgt ervoor dat het vraagteken wordt getoond.

`#checkbox-help:not(:checked)`: alleen als het element met id=”checkbox-help”, de checkbox, niet is aangevinkt.

`+ #wrapper-icons #icons`: het element met id=”icons” dat binnen een element met id=”wrapper-icons” ligt. Waarbij `#wrapper-icons` in de html vanwege de + gelijk na `#checkbox-help` moet komen en dezelfde ouder moet hebben.

Het tonen van het vraagteken is feitelijk de normale situatie. (Weinig mensen zullen immers de uitleg liever bekijken dan een video, neem ik aan. Hoe wonderschoon deze uitleg ook is geschreven en opgemaakt en zo, al zeg ik het zelf.)

Alleen als de checkbox is aangevinkt, moet het sluitkruisje worden getoond. Dat wordt iets lager afgehandeld. En bij hoveren moet het icon met de pin worden getoond. Dat wordt hierboven afgehandeld.

```
#checkbox-help:checked + #wrapper-icons + #helptext {left: 0;}
```

De uitleg is bij #helptext links buiten het scherm geparkeerd. Door deze aan de linkerkant van het browservenster te zetten, wordt hij nu zichtbaar.

`#checkbox-help:checked`: doe dit als de checkbox is aangevinkt. De + wil zeggen dat `#helptext` in de html gelijk op `#wrapper-icons` moet volgen, en die weer gelijk op `#checkbox-help`. Ook moeten ze alle drie dezelfde ouder hebben.

```
#checkbox-help:checked + #wrapper-icons #icons {background-
    position: 50px;}
```

Met deze achtergrondpositie wordt het sluitkruisje getoond op de plaats, waar eerst het vraagtekentje stond. De selector is precies hetzelfde als die hierboven, alleen eindigt dit op `#icons`, want in dat element staat de achtergrond-afbeelding met het sluitkruisje.

```
#wrapper-help:hover:not(:focus) #helptext {left: 0;}
```

De uitleg is bij #helptext links buiten het scherm geparkeerd. De bedoeling is dat deze, bij hoveren over het vraagteken, aan de linkerkant van het browservenster wordt gezet en dus zichtbaar wordt. Waarvoor eigenlijk `#wrapper-help:hover #helptext` voldoende zou zijn, zou je denken.

Helaas blijkt dat iOS hoveren hetzelfde behandeld als een aanraking. Als je op iOS het vraagteken aanraakt, wordt de uitleg getoond. En die blijft getoond worden, totdat je iets anders aanraakt dat focus kan krijgen, iets als een knop of een link.

Door als extra voorwaarde op te nemen dat `#wrapper-help` geen focus mag hebben, als de icon met de pin wordt getoond, kun je dit oplossen. Het heeft verder geen invloed op normaal hoveren op de desktop, want dat staat los van focus.

Om op iOS op focus te kunnen testen, blijkt `#wrapper-help` een tabindex te moeten hebben. Toevallig staat er al een `tabindex="-1"`, omdat die nodig is voor een klein probleem op Android 4.4.2, zoals beschreven bij Tabindex. De volgorde van de Tab-toets wordt niet verstoord, want als een element een `tabindex="-1"` heeft, wordt dat element overgeslagen door de Tab-toets.

Video en tekst en links onder en boven de video's

```
h2, .origineel, .groot, .klein {max-width: 474px;}
.origineel {font-size: 0.7em;}
```

Voor kleinere browservensters is bij h2, h3, .origineel, .groot .klein allerlei css opgegeven voor kopjes, links naar de originele video's, e.d. Die worden hier deels aangepast voor grotere vensters.

```
video {width: 480px; height: 320px;}
```

Breedte en hoogte voor de <video>'s. Eigenlijk zou dit niet nodig moeten zijn, of hoogstens de breedte. Maar op iOS worden de metadata (waaronder dingen als hoogte en breedte van de video) pas gedownload, als de video wordt afgespeeld. (Het lijkt erop dat dit sinds heel kort met de speelduur niet meer zo is.) De video in iOS wordt standaard op 300x150 px getoond. Bij het afspelen wordt die grootte niet aangepast, terwijl dan toch hoogte en breedte bekend zijn. Vandaar dat voor de video's hoogte en breedte worden opgegeven. (Dit is trouwens het enige dat blijkt te werken: `.videobox` een variabele grootte geven en <video> een hoogte en breedte van 100% werkt niet, breedte en hoogte van <video> `auto` geven werkt niet, enz.)

```
.groot {display: block;}
.klein {display: none;}
```

In de elementen met class="groot" staan de download-links voor de grote video's, in de elementen met class="klein" die voor de kleine video's.
Hierboven zijn bij .groot voor kleinere browservensters de links naar de grote video's verborgen. Hier worden de links naar de kleine video's verborgen en juist de links naar de grote video's zichtbaar gemaakt.

css voor vensters breder dan 1000 px

```
@media screen and (min-width: 1000px)
```

De css die hier tot nu toe staat, geldt ook voor browservensters breder dan 1000 px. Dat geldt ook voor de css voor vensters breder en hoger dan 480 px. Elk venster dat minimaal 480 px breed is, is immers ook 1000 px breed, en zo'n venster zal ook hoger zijn dan 480 px.
De css die hieronder staat, geldt alleen voor browservensters breder dan 1000 px. Voor een deel is dit nieuwe css, voor een deel wordt hierboven staande css aangepast.
De opbouw van de regel staat beschreven bij css voor vensters breder en hoger dan 480 px, het enige verschil is dat het hier om een minimumbreedte van 1000 px gaat en dat er geen voorwaarde voor de hoogte van het venster geldt.

```
.videobox {width: 50%; float: left; margin-top: 30px;}
```

De elementen met class="videobox". De 's waarin de video's met bijbehorende titel, links, e.d. staan.
Het zijn 's, dus komen ze elk op een nieuwe regel te staan. In browservensters breder dan 1000 px kunnen er twee naast elkaar komen te staan, want de video's zijn maar 480 px breed.
De ' worden half zo breed als het venster van de browser. Naar links floaten, zodat er twee naast elkaar komen te staan. Als de video dan binnen de wordt gecentreerd, staan de video's altijd netjes binnen het venster van de browser verdeeld.
Kleine marge aan de bovenkant voor wat ruimte tussen de video's met bijbehorende titels e.d.

css voor vensters breder dan 1500 px

```
@media screen and (min-width: 1500px)
```

De css die hier tot nu toe staat, geldt ook voor browservensters breder dan 1500 px. Dat geldt ook voor de css voor vensters breder en hoger dan 480 px, en de css voor vensters breder dan 1000px. Elk venster dat minimaal 480 px of 1000 px breed is, is immers ook 1500 px breed, en zo'n venster zal ook hoger zijn dan 480 px..
De css die hieronder staat, geldt alleen voor browservensters breder dan 1500 px. Voor een deel is dit nieuwe css, voor een deel wordt hierboven staande css aangepast.
De opbouw van de regel staat beschreven bij css voor vensters breder en hoger dan 480 px, het enige verschil is dat het hier om een minimumbreedte van 1500 px gaat en dat er geen voorwaarde voor de hoogte van het venster geldt.

```
#wrapper-help {left: 50%; margin-left: -740px;}
```

In de met id="wrapper-help" staat het vraagtekentje, waaronder de pop-up met de uitleg zit. In heel brede browservensters komt deze onhandig ver van de video's af te staan, omdat absoluut is gepositioneerd op 5 px van links. Ook de uitleg zelf komt te ver van de video's af te staan.
Zet de met de uitleg horizontaal in het midden van het venster, ongeacht hoe breed dit is. Verplaats daarna de (met het daarin zittende vraagteken, uitleg, enz.) weer 740 px terug naar links. Nu staat de op een vaste afstand vanaf het midden van het browservenster en kan dus nooit te ver naar links komen te staan.

css voor door het script ingevoegde elementen

Het grootste deel van deze css wordt altijd gebruikt, zonder eisen aan breedte of hoogte van het browservenster. Dit is dus anders dan bij het script, wat alleen volledig wordt gebruikt in vensters met een bepaalde minimumbreedte en hoogte. Dat maakt niets uit, want de css wordt alleen gebruikt voor elementen die door het script worden ingevoegd. En omdat het script geen elementen invoegt in vensters smaller of lager dan 480 px, kan de css in die kleinere vensters ook niet worden gebruikt. (Hoe die controle op 480 px werkt, en hoe je die maat kunt aanpassen of de hele controle kunt uitschakelen, staat bij data-smscr="480".)
(Een klein nadeel is er wel: in kleinere vensters wordt ook css gedownload, die vervolgens niet wordt gebruikt. Ik til daar echter niet zo heel erg zwaar aan, omdat 'n heel kort stukje video al tig keer zo groot is als de stylesheet. Als je verbinding zo duur of slecht is dat dit 'n probleem is, dan zul je zeker geen video kijken, lijkt me.)
De elementen waar deze css bij hoort, zijn niet te zien in de gewone broncode. In de gewone broncode staat alleen maar, wat je daar zelf in hebt gezet. Om de elementen bij deze css te zien, moet je de gegenereerde code bekijken.
De id's en classes van de elementen waar deze css bij hoort, worden ingevoegd door het script. Je kunt daar meer over vinden bij Overzicht van door het script ingevoegde bedieningselementen, classes en id's. Hoe je de namen van die id's en classes eventueel kunt wijzigen, is te vinden bij Class, ClassId en Id (voor elementen met respectievelijk alleen een class, een class én een id, en alleen een id).

Keuze standaard- of aangepaste videospeler

```
#kiezen-div {background: white; color: black; min-width: 480px;
    max-width: 740px; margin: 10px auto 0; border: black solid
    1px; border-radius: 8px;}
```

Het element met id="kiezen-div". Bovenaan de pagina kun je kiezen voor tussen de standaard- en de aangepaste videospeler. Hier wat opmaak voor de `div#kiezen-div`, waarin die keuze staat.

```
@media screen and (max-width: 850px) {
      #kiezen-div {
            min-width: 0;
            margin: 10px 60px 0;
      }
      #helptext {
            width: 480px;
            max-width: none;
            position: absolute;
            top: 50px;
      }
}
```

Deze css is alleen bedoeld voor browservensters die hoogstens 850 px breed zijn. Dat is te zien aan de eerste regel: `(max-width: 850px)`. Hoe deze regel precies is opgebouwd, is te zien bij css voor vensters breder en hoger dan 480 px.
Alle css die tussen `@media screen and (max-width: 850px) {` en de onderste, afsluitende `}` staat, hoort bij deze maximale breedte.

De css voor `#kiezen-div`:

Bij #kiezen-div, de waarbinnen de keuze voor standaard- of aangepaste videospelers staat, is een minimale breedte van 480 px opgegeven en alleen een marge aan de bovenkant. Dat is bedoeld voor smallere vensters en daar gaat het goed, maar bij bepaalde breedtes tussen de 480 px en 850 px komt de tekst, waarmee je standaard- of aangepaste videospelers kiest, over het vraagtekentje van de uitleg te staan. Om dat te voorkomen, wordt links en rechts een marge van 60 px aan `#kiezen-div` gegeven.
In een venster van bijvoorbeeld 530 px breed zijn die marges links en rechts samen 120 px breed. Als de minimumbreedte van `#kiezen-div` 480 px zou blijven, wordt de totale breedte 60 + 480 + 60 = 600 px. Oftewel: breder dan het venster, waardoor je horizontaal moet gaan scrollen. Om dat te voorkomen wordt de minimumbreedte bij `#kiezen-div` weggehaald.
In vensters breder dan 850 px speelt dit probleem niet, dus daar is deze css niet nodig.

De css voor `#helptekst`, de waar de eigenlijke helptekst in staat:

Bij #helptext is een reeks instellingen voor de tekst van de uitleg opgegeven, die problemen opleveren in browservensters die niet al te breed zijn. En omdat deze regel voor een maximale breedte van 850 px er toch is, kunnen we die gelijk gebruiken om de tekst van de uitleg wat beter te laten passen in dit soort vensters.
De breedte die eerder is opgegeven is 700 px, dat wordt verminderd tot 480 px voor vensters smaller dan 850 px. Dan past het altijd binnen het venster.
De maximale breedte van 90% wordt weggehaald, want anders wordt de uitleg in smalle vensters wel heel smal.
De uitleg is fixed gepositioneerd, scrolt dus niet mee. Dat is geen goed idee in smalle vensters, want mogelijk kun je dan niet de hele uitleg lezen, dus de positie wordt veranderd in absoluut en de uitleg wordt op 50 px vanaf de bovenkant neergezet.
In vensters breder dan 850 px spelen deze problemen niet, dus daar is deze css niet nodig.

`.kiezen-par {text-indent: -10px; margin: 0 0 0 20px; padding: 3px; position: relative;}`

De keuzes tussen standaard- of aangepaste videospelers staan elk binnen een eigen <p> met class=”kiezen-par”, waarvoor hier wat opmaak wordt gegeven. Een afbeelding van deze <p>’s staat hier iets onder.

`.kiezen-label {display: inline-block; padding-right: 20px;}`

De <label>’s die bij de radioknoppen voor het kiezen tussen standaard- en aangepaste videospelers horen hebben een class=”kiezen-label”.
De padding rechts maakt wat lege ruimte, waarin de radioknop neergezet kan worden. Zonder deze lege ruimte zou de tekst doorlopen tot onder de radioknop. Als je op een <label> bij een radioknop klikt, wordt de radioknop aan- of uitgevinkt. Dat werkt ook zo op een padding bij een <label>. Maar niet bij een marge. Als je ’n marge in plaats van ’n padding zou gebruiken, krijg je daardoor ’n ruimte tussen <label> en bijbehorende radioknop, waar klikken niet werkt. Door een padding te gebruiken, werkt klikken ook op de lege ruimte tussen <label> en knop.

`.kiezen-par`, de <p> waar elke keuze in staat, heeft even een blauwe achtergrond gekregen. De labels hebben een roze achtergrond gekregen. Duidelijk is te zien dat de radioknoppen boven de padding van de label zijn gepositioneerd.

`.kiezen-knop {position: absolute; right: 3px;}`

De elementen met class=”kiezen-knop”. Dit zijn de eigenlijke radioknoppen, waarmee kan worden gekozen tussen standaard- en aangepaste videospeler. Deze worden absoluut gepositioneerd op de lege ruimte rechts van de tekst in de bij de knop horende <label>, zoals op de afbeelding iets hoger is te zien.

`#kiezen-knop-2 {bottom: 6px;}`

Het element met id=”kiezen-knop-2”. Dit is de tweede radioknop, de knop waarmee wordt gekozen voor de in de browser ingebouwde standaardspeler.
Op deze hoogte past de tweede radioknop iets beter bij de bijbehorende tekst.

Melding speelduur nog onbekend

Als de speelduur van de video om een of andere reden nog onbekend is, werken bepaalde functies nog niet goed. Als die functies toch worden gebruikt, verschijnt een melding. Tekst, id’s, e.d. van deze melding kunnen in het script worden aangepast.
Openen en sluiten van de melding worden volledig door het script geregeld. Een uitgebreide beschrijving staat bij Speelduur nog onbekend. Daar staat ook, hoe je de tekst van de melding en de tekst op de knop kunt aanpassen, of eventueel helemaal kunt uitschakelen. Op deze plaats wordt alleen de css voor de melding besproken.
De hele melding staat binnen een . De tekst zelf staat, binnen die , in een <p>, gevolgd door een <button>. Omdat deze drie elementen elk een id hebben, kunnen ze gewoon met css van uiterlijk worden veranderd.

`#duration-div`

Het element met id=”duration-div”. De waarbinnen de hele melding staat.

`background: #eee; width: 400px; font-size: 1.5em; text-align: center;`

Achtergrondkleurtje, breedte, grotere letter, tekst horizontaal centreren.

`margin-left: -208px;`

Hieronder wordt de melding fixed gepositioneerd en horizontaal halverwege het venster van de browser geplaatst. De breedte van de melding is 400 px. Daar komen links en rechts nog een padding van 5 px en een border van 3 px bij. De totale breedte wordt dan 3 + 5 + 400 + 5 + 3 = 416 px.

Als je nu de melding de helft hiervan terugzet naar links, staat de melding horizontaal altijd in het midden, ongeacht de breedte van het venster.

`border: red solid 3px; border-radius: 10px; padding: 5px;`

Rode rand, ronde hoeken, kleine ruimte tussen rand en tekst.

`opacity: 0.9;`

Klein beetje doorzichtig maken.

`position: fixed; top: 100px; left: 50%;`

Fixed positioneren. Omdat deze melding vrij klein is, levert dat nergens problemen op. Scrollen is niet nodig om deze melding te kunnen lezen, dus ook als fixed niet goed werkt, zoals op sommige touchscreens, zullen er geen problemen optreden.

100 px vanaf de bovenkant en horizontaal halverwege het venster van de browser neerzetten. De reden van het laatste staat iets hierboven bij `margin-left: -208px;`.

`z-index: 1000;`

Tamelijk hoge z-index, zodat de melding nooit ergens onder kan verdwijnen.

`#duration-par {margin: 0;}`

Het element met id=”duration-par”. De <p> waar de eigenlijke tekst van de melding in staat. Een <p> heeft standaard een marge aan boven- en onderkant. Die zijn hier niet welkom.

`#duration-button`

Het element met id=”duration-button”. De <button> waarmee de melding kan worden gesloten.

`display: block;`

Een <button> is een inline-element. Door er een blok-element van te maken, komt het op een nieuwe regel te staan en kan `margin` worden gebruikt.

`width: auto;`

De tekst binnen de <button> is ’Melding sluiten’. iOS zet dat op twee regels, alle andere browsers passen de breedte van de <button> netjes aan. Hiermee staat de tekst ook in iOS op één regel.

`margin: 0 auto;`

Omdat voor onder en links geen waarde is opgegeven, krijgen deze dezelfde waarde als boven en rechts. Hier staat dus eigenlijk `0 auto 0 auto` in de volgorde boven – rechts – onder – links. Boven en onder geen marge, links en rechts `auto` wat hier hetzelfde betekent als evenveel. De <button> staat dus altijd horizontaal gecentreerd, ongeacht de breedte van het browservenster.

(Feitelijk werkt dit alleen bij blok-elementen die een breedte hebben. Dat is hier niet het geval. Bij <button> werkt het kennelijk ook, als geen breedte is gegeven. Alleen in Android browser staat de <button> gewoon links.)

Bedieningselementen algemeen (pagina 1)

```
.videobox[data-duration="unknown"] .to-begin, .videobox[data-
    duration="unknown"] .five-back, .videobox[data-
    duration="unknown"] .ten-back, .videobox[data-
    duration="unknown"] .ten-forward, .videobox[data-
    duration="unknown"] .five-forward, .videobox[data-
    duration="unknown"] .to-end, .videobox[data-
    duration="unknown"] .fullscreen, .videobox[data-
    duration="unknown"] .image-slider {opacity: 0.3;}
```

Een hele serie selectors, maar niet echt heel ingewikkeld. Als je even `[data-duration="unknown"]` weglaat, staan hier hele simpele selectors:

```
.videobox .to-begin, .videobox .five-back, .videobox
    .ten-back, .videobox .ten-forward, .videobox .five-
    forward, .videobox .to-end, .videobox
    .fullscreen, .videobox .image-slider
```

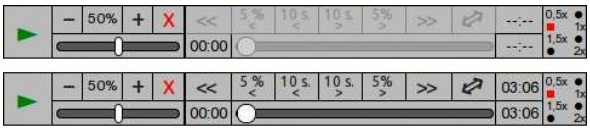
Dit zijn de knoppen Naar begin, Vijf procent terug, Tien seconden terug, Tien seconden vooruit, Vijf procent vooruit, Naar einde, Fullscreen, en de sleepbalk voor afspelen. Deze knoppen en de sleepbalk hebben alleen enig nut, als de speelduur van de video bekend is. En dat is niet altijd het geval bij openen van de pagina. Die speelduur wordt ’n paar keer opgevraagd, maar soms lukt het pas als de video daadwerkelijk wordt afgespeeld.
Bij openen van de pagina wordt aan de elementen met class=”videobox” door het script `data-duration="unknown"` toegevoegd:

```
<div class="videobox" data-duration="unknown">
```

(Alle html die niet echt nodig is, is hierboven even weggelaten.) Zodra de speelduur bekend is, wordt dit veranderd in `data-duration="known"`. Door te testen op de waarde van `data-duration` kan aan nog niet werkende bedieningselementen een ander uiterlijk worden gegeven. En dat is precies wat hierboven gebeurt. De eerste selector volledig:

```
.videobox[data-duration="unknown"] .to-begin
```

Het deel tussen `[]` moet letterlijk aanwezig zijn. Deze selector betekent dus: elementen met class=”to-begin” die binnen een element met class=”videobox” zitten, maar alleen als het element met class=”videobox” het attribuut `data-duration="unknown"` heeft. Zolang de speelduur onbekend is, zal dit zo zijn. Zodra de speelduur bekend is, verandert het attribuut in `data-duration="known"` en werkt de selector niet meer.
Zolang de speelduur onbekend is, worden deze acht bedieningselementen iets doorzichtig weergegeven.



Boven de weergave van de bediening als de speelduur nog onbekend is. Alles om voor of achteruit te gaan en de knop voor fullscreen zijn doorzichtig.
Onderaan de weergave als de speelduur bekend is.

Je kunt natuurlijk ook iets heel anders doen dan doorzichtig maken. Wat nog niet werkt helemaal weglaten. Of er met behulp van `::after` ’kssst, gaat henen, engerd’ op zetten. Of ze met behulp van `transform` ondersteboven neerzetten. Of tien centimeter boven de video.

```
.videobox[data-duration="known"] {-webkit-animation: bugfix
    infinite 1s;}
```

Door een bug in oudere versies van Android browser worden de hier gelijk boven gedeeltelijk doorzichtig gemaakte bedieningselementen niet ondoorzichtig gemaakt, zodra de speelduur bekend is. Deze regel lost dat op. Meer hierover bij @-webkit-keyframes bugfix.

```
.controls {background: #bbb; width: 480px; height: 42px; margin: 0
    auto; border: black solid; border-width: 0 1px 1px;}
```

Alle bedieningselementen staan binnen een met class=”controls”. Dat is het grijze blok op de onderstaande afbeelding. Deze wordt door het script altijd gelijk na het <video>-element in de html ingevoegd, maar kan probleemloos overal worden neergezet met behulp van `position` of zoiets. Op onderstaande afbeelding is alleen `margin: 0 auto;` even weggehaald. Hierdoor is te zien dat de bedieningselementen echt volkomen los van de video staan.

Door alle bedieningselementen bij elkaar in één te zetten, kan aan deze het aria-attribuut `role="group"` worden gegeven, waardoor schermlezers weten dat het hier om een groep bij elkaar horende elementen gaat.
De voor deze bedoelde css is verder weinig interessant.
Door het script wordt ook nog wat onmisbare css ingevoegd in de vorm van een inline-style. Omdat deze inline-styles door het script worden ingevoegd, zijn ze alleen te zien in de gegenereerde code.
Bij openen van de pagina wordt `div.controls` gewoon getoond, en dus alle bedieningselementen ook. Dat is standaard, dus daar is geen css voor nodig. Als je nu kiest voor de standaardvideospeler, voegt het script een inline-style toe: `style="display: none;"`. Waardoor `div.controls` en daarmee alle daarin zittende bedieningselementen worden verborgen.
Als je nu weer kiest om de aangepaste videospeler te gebruiken, wordt in deze inline-style `display: none;` veranderd in `display: block;`, waardoor de en dus de bedieningselementen weer zichtbaar worden.
Zodra je dus een van de radioknoppen hebt gekozen, zul je altijd een van deze twee inline-styles zien.

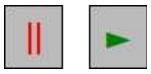
Een andere inline-style is al bij het openen van de pagina aanwezig:

```
style="-ms-touch-action: none; touch-action: none; -moz-
    user-select: none; -ms-user-select: none; -webkit-
    user-select: none; -user-select: none;"
```

Dit is nodig voor een goede werking van het script. Wat deze css precies doet, kun je zien bij controlsDivNodeList[i].setAttribute(”style”...

Speel-pauzeerknop (pagina 1)

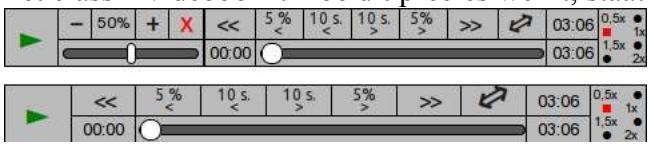
```
.play {background: transparent; width: 40px; height: 42px; float:
    left; border: none; border-right: black solid 1px; border-
    radius: 0;}
```



Alle elementen met class=”play”. Dit zijn de Speel-pauzeerknoppen, <button>’s van het type ’button’. De standaardweergave daarvan verschilt per browser. Daarom worden achtergrondkleur en ronde hoeken weggehaald en breedte, hoogte en border hier expliciet opgegeven. Nu ziet het er in alle browsers hetzelfde uit.
Alleen rechts is een border nodig, deze sluit netjes aan op borders van andere elementen. De knop wordt naar links gefloat, zodat er meerdere knoppen e.d. naast elkaar kunnen komen.
Deze knop hoort bij de bediening voor het afspelen. Het is de enige knop die niet in de subgroep `div.image` staat, waar alle andere knoppen voor het afspelen wel in staat. Dat is gedaan, omdat dit de belangrijkste knop is. Door deze apart te zetten, kun je de knop probleemloos overal neerzetten.

```
.videobox[data-sound="no"] .play {border-right: none;}
```

Op iOS kan de geluidssterkte alleen met de fysieke knop op de tablet worden veranderd, niet met behulp van de knoppen in de videospeler. Het script test of het geluid gewijzigd kan worden en als dat niet mogelijk is, wordt `data-sound="no"` toegevoegd aan de ’s met class=”videobox”. Hoe dit precies werkt, staat uitgebreider beschreven bij data-sound.



Boven de normale bediening, onder de bediening op iOS, waar de geluidsregeling ontbreekt.

`[data-sound="no"]` test op de aanwezigheid van dit attribuut en deze waarde. Alleen als `data-sound="no"` aanwezig is, is deze selector van kracht. Met andere woorden: op iOS kan de geluidsregeling worden verborgen, of er anders uitzien, of wat dan ook.
Op de eerste pagina met videospelers wordt de geluidsregeling op iOS helemaal verborgen. Maar daardoor veranderen de maten van alle andere bedieningselementen natuurlijk ook, dus daar zijn wat aanpassingen voor nodig, zoals het weglaten van de rechterborder bij de Speel-pauzeerknop.

```
.play::after {content: "||"; color: red; font-size: 22px; speak:
    none;}
```

De Speel-pauzeerknop heeft een class=”play”. Met behulp van `::after` wordt een pseudo-element aangemaakt, waarmee boven de knop tekst wordt neergezet: twee verticale lijntjes. De verticale lijntjes zijn twee gewone tekens, net zoals letters. Ze worden rood en tamelijk groot weergegeven.
Als de video niet afspeelt, wordt een extra class toegevoegd, waardoor het uiterlijk van de knop kan worden gewijzigd. Dit wordt gelijk hieronder beschreven.
`speak: none;` is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

```
.not-playing::after {content: "\25ba"; color: green;}
```

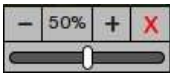
De Speel-pauzeerknop heeft altijd een class=”play”. Maar zodra een video wordt gepauzeerd of aan het eind is aangekomen, wordt aan de Speel-pauzeerknop van die video door het script een extra class toegevoegd: `.not-playing`. Deze extra class wordt weer verwijderd, zodra de video afspeelt.
Door in de selector deze extra class op te nemen, kan het uiterlijk van de Speel-pauzeerknop worden gewijzigd, als de video niet speelt. Met behulp van `::after` wordt een driehoekje

boven de knop gezet, groen gekleurd. Dit driehoekje is een gewoon teken, net zoals een letter, maar het zit niet op het toetsenbord. Daarom wordt het via een zogenaamde utf-8-code ingevoegd: `25ba`. Dat is, simpel gezegd, een volgnummer voor een bepaald teken. Omdat een computer nou eenmaal niet echt slim is, zou gewoon '25ba' worden weergegeven. Daarom staat er aan het begin een zogenaamde 'escape code': `\`. Die geeft aan dat, wat erop volgt, geen letterlijke tekst is, maar een code.

Knoppen volume (pagina 1)

De sleepbalk voor geluid en alle knoppen die met geluid te maken hebben, staan binnen één gezamenlijke . Dit geeft de mogelijkheid aan deze het aria-attribuut `role="group"` te geven, zodat schermlezers weten dat het hier om een groep bij elkaar horende elementen gaat.

```
.sound {width: 108px; height: 42px; float: left; position:
    relative;}
```



De knoppen voor Zachter, Harder en Geluid aan/uit, de weergave van de geluidssterkte en de sleepbalk voor de geluidssterkte zitten samen in een met class="sound". Hierdoor kan de bediening van de geluidsregeling door schermlezers als één groep worden gezien.
De wordt naar links gefloat, zodat hij naast de Speel-pauzeerknop komt te staan.

```
.videobox[data-sound="no"] .sound {display: none;}
```

Op iOS wordt de geluidsregeling verborgen, omdat die daar toch niet werkt. Meer daarover is te vinden bij .videobox[data-sound="no"] .play.

```
.sound button, .sound span {background: transparent; width: 25px;
    height: 23px; float: left; text-align: center; border: black
    solid; border-width: 0 1px 1px 0; border-radius: 0; position:
    relative;}
```

In `div.sound` zitten o.a. drie <button>'s (Harder, Zachter, Geluid aan/uit) en twee 's (weergave en voorlezen geluidssterkte). Deze zien er grotendeels hetzelfde uit, dus kunnen ze in één keer veel dezelfde css krijgen.
De standaardweergave van een <button> verschilt per browser. Een groot deel van deze css zorgt er dan ook alleen voor dat ze er overal hetzelfde uitzien. Ze worden naar links gefloat, zodat ze naast elkaar komen te liggen.

```
.sound span {width: 32px; height: 22px;}
```

Voor deze elementen geldt ook de hier gelijk boven bij `.sound button, .sound span` opgegeven css, voor zover die hier niet wordt veranderd.
Er zitten twee 's in `div.sound`, waarvan er eentje alleen maar voor het voorlezen van de geluidssterkte dient. Die wordt verderop onzichtbaar linksbuiten het scherm geparkeerd, dus het uiterlijk daarvan is niet belangrijk.
De waarin de geluidssterkte wordt weergegeven, is wel zichtbaar. De meeste css is gelijk hierboven bij `.sound button, .sound span` al opgegeven, hier worden alleen breedte en hoogte aangepast.

```
.softer::after {content: "--"; font-size: 20px; letter-spacing:
    -3px; line-height: 16px; speak: none;}
```

De knop voor Zachter heeft een class="softer". Met behulp van `::after` wordt een

pseudo-element gemaakt, waarmee tekst boven de knop wordt gezet: twee koppeltekens. Door ze met behulp van `letter-spacing` iets dichter bij elkaar te zetten, zien ze eruit als één lang streepje.
`speak: none;` is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

```
body .aria-volume {position: absolute; left: -20000px;}
```

De elementen met class=”aria-volume” zijn ’s, die alleen dienen om in schermlezers op bepaalde momenten de geluidssterkte voor te lezen. `body` is nodig om de selector genoeg specificiteit (’gewicht’) te geven om eerdere css te overrulen.
Door de ’s ver links buiten het scherm te parkeren, worden ze wel gelezen door een schermlezer, maar verstoren ze de lay-out niet. Als je ze zou verbergen met `display: none;` zou een schermlezer ze ook volledig negeren.
Het voorlezen van de gewijzigde geluidssterkte wordt volledig door het script geregeld.

```
.percentage {font-size: 12px; line-height: 20px;}
```

Voor deze elementen geldt ook de iets hierboven bij `.sound button, .sound span` en `.sound span` opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”percentage”. De ’s waarin de geluidssterkte wordt weergegeven. Verder weinig spannends hier. Het daadwerkelijke invullen van het percentage gebeurt door het script.

```
.louder::after {content: "+"; font-size: 20px; line-height: 16px;
speak: none;}
```

De knop voor Harder heeft een class=”louder”. Met behulp van `::after` wordt een pseudo-element gemaakt, waar tekst boven de knop wordt gezet: een plusteken.
`speak: none;` is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

```
.sound .mute {border-right: none;}
```

De Aan-uitknop voor geluid heeft een class=”mute”. Als de rechterborder hier blijft staan, wordt de border 1 px te dik, dus wordt hij weggehaald.
`.sound` is nodig, om de selector genoeg specificiteit (’gewicht’) te geven om eerdere css te overrulen.

```
.mute::after {content: "x"; color: red; font-size: 20px; line-
height: 16px; speak: none;}
```

De Aan-uitknop voor geluid heeft een class=”mute”. Met behulp van `::after` wordt een pseudo-element gemaakt, waarmee tekst boven de knop wordt gezet: gewoon de letter ’x’. Door deze groot en rood weer te geven, gaat het op een uitzet-kruisje lijken.
`speak: none;` is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

```
.mute.muted::after {content: "o"; color: green;}
```

De Aan-uitknop voor het geluid heeft altijd een class=”mute”. Maar zodra het geluid wordt uitgezet, wordt aan de Aan-uitknop voor het geluid door het script een extra class toegevoegd: `.muted`. Deze extra class wordt weer verwijderd, zodra het geluid weer wordt aangezet.

Door in de selector deze extra class op te nemen, kan het uiterlijk van de Aan-uitknop worden gewijzigd, als het geluid wordt uitgezet. Met behulp van `::after` wordt de letter 'o', groen gekleurd, weergegeven boven de knop.

Sleepbalk volume (pagina 1)

 Feitelijk is er helemaal geen echte sleepbalk. Er zijn drie geneste 's.
De grijze rechthoek op de afbeelding is de buitenste . Deze heeft alleen maar nut voor de css. Deze heeft een class="sound-slider".
De middelste is de donkerder balk op de afbeelding. Dit is het deel dat reageert op aanraking of klikken. Deze heeft een class="sound-slider-beam".
De middelste is de witte knop op de afbeelding. Deze heeft een class="sound-slider-button". Deze is alleen voor de sier aanwezig: de sleepbalk werkt ook prima zonder deze knop.

`.sound-slider-beam`

 De elementen met class="sound-slider-beam". Dit zijn 's, die de balk van de sleepbalk vormen. De knop van de sleepbalk komt hieronder aan de beurt.
Na het laden van de pagina wordt door het script de breedte van de balk van de sleepbalk ingelezen. Dat is nodig om de knop van de sleepbalk op de juiste plaats neer te zetten. Die plaats is afhankelijk van de geluidssterkte: bij 0% geluidssterkte staat de knop helemaal links op de balk, bij 100% staat de knop helemaal rechts.

De breedte van de balk van de sleepbalk die wordt gebruikt is de opgegeven breedte (`width`), zonder padding, borders of margin. Een padding heeft geen enkele invloed op wat dan ook, dus het is zinloos die op te geven. Margin en borders zijn ook onbelangrijk, omdat je die niet voor slepen gebruikt: alleen slepen op de balk zelf heeft effect.
Voor de breedte van de knop van de sleepbalk wordt de breedte (`width`) plus een eventuele padding plus een eventuele border door het script berekend. Een marge telt niet mee voor de breedte van de knop, maar verstoort wel de goede plaatsing van de knop. (Als je de knop links en rechts niet tot begin en einde van de balk van de sleepbalk wilt laten komen, kun je bij de vierde videospeler op de tweede pagina zien, hoe dat gedaan kan worden.)
In de uiterste stand van de knop komt de buitenkant van de border van de knop tegen de binnenkant van de border van de balk van de sleepbalk te staan. (Als er geen border is, moet je die er maar even bij bedenken.)

Na het laden van de pagina word dus op bovenbeschreven wijze door het script de breedte van de balk van de sleepbalk en de breedte van de knop van de sleepbalk ingelezen. Het script berekent vervolgens de juiste positie van de knop. Als de geluidssterkte wordt aangepast, wordt die positie opnieuw berekend.
Omdat het script alles berekend, kun je dus elke breedte die je wilt opgeven voor balk en knop. Als je in de selector id's gebruikt, kun je zelfs aan sleepbalken en -knoppen in verschillende spelers op dezelfde pagina een verschillende breedte geven (dit is op een aantal pagina's met spelers het geval).

Op bovenstaande afbeelding heeft de bovenste sleepbalk een onwijs brede zwarte border. De rode sleepknop is uiterst smal. De onderste sleepbalk heeft een smalle border, de sleepknop daarentegen heeft links en rechts een brede border. De knop is ook 'n pietsie breder dan die van de bovenste sleepbalk.

Hoewel het uiterlijk van beide sleepbalken nogal verschilt, staat bij beide bij een geluidssterkte van 0% de knop links tegen de border van de sleepbalk, bij 100% staat de knop tegen de rechterborder, en bij 50% staat het midden van de knop precies in het midden van de sleepbalk.
(Het voert te ver om hier uit te leggen, hoe dat wordt berekend. Voor de liefhebber: dat is te vinden in `soundBeginSliding()` in het script.)

De met de sleepknop wordt absoluut gepositioneerd ten opzichte van de met de balk van de sleepbalk. Omdat de knop absoluut wordt gepositioneerd ten opzichte van de met de balk, móét de met de balk absoluut, relatief of fixed zijn gepositioneerd. Dit is zo belangrijk, dat het script controleert op de positie van `.sound-slider-beam`. Als die positie statisch is, wordt een inline-style toegevoegd: `style="position: relative;"`. Als `.sound-slider-beam` al een fixed, absolute of relatieve positie heeft, wordt er niets ingevoegd, want anders zou het script ongewild de lay-out kunnen verstoren.
Deze door het script ingevoegde css is niet te zien in de gewone broncode, maar alleen in de gegenereerde code.

`background: #555;`
: Donkerder achtergrond.

`clear: both;`
: Onder de gefloate knoppen voor harder, zachter, e.d. neerzetten.

`width: 100px; height: 6px;`
: Hoogte en breedte van de .
Over het belang van de breedte voor de plaatsing van de sleepknop staat bovenaan gelijk onder `.sound-slider-beam` meer.

`margin: 0 auto;`
: horizontaal centreren in de ruimte die is bestemd voor de balk van de sleepbalk.
Over de invloed van de breedte van de marge staat bovenaan gelijk onder `.sound-slider-beam` meer.

`border: black solid 1px; border-radius: 5px;`
: Randje met ronde hoeken rondom de balk.
Over de invloed van de breedte van de border staat bovenaan gelijk onder `.sound-slider-beam` meer.

`position: relative;`
: De knop van de sleepbalk wordt gepositioneerd ten opzichte van de de sleepbalk, dus ten opzichte van `div.sound-slider-beam`. Dat kan alleen als `.sound-slider-beam` zelf absoluut, relatief of fixed is gepositioneerd.
Over het belang van een positie staat bovenaan gelijk onder `.sound-slider-beam` meer.

`top: 6px;`
: Op deze hoogte staat de met de balk van de sleepbalk verticaal in het midden.

`.sound-slider-button`
: De elementen met class=”sound-slider-button”: de knop van de sleepbalk. De balk van de sleepbalk is hierboven besproken.
De knop wordt met behulp van een `position: absolute;` door het script op de juiste plaats neergezet. Er wordt gepositioneerd ten opzichte van `.sound-slider-beam`, de

 die de balk van de sleepbalk vormt. Deze knop is ook een , die in `div.sound-slider-beam`, de met de balk, zit.
Het samenspel tussen knop en balk van de sleepbalk wordt gelijk hierboven bij `.sound-slider-beam` besproken. Als je wilt weten, hoe de knop werkt, is het is belangrijk dat ook even te lezen.

Voor een goede werking van de sleepknop is een kleine hoeveelheid css absoluut onmisbaar. Om er zeker van te zijn dat die ook inderdaad aanwezig is, voegt het script die toe in de vorm van een inline-style.
In het script kan bovenin een geluidssterkte worden opgegeven, die wordt gebruikt als de pagina wordt geopend. (Hoe je die geluidssterkte bij opening kunt wijzigen, is te vinden bij soundStartVolume.) De knop van de sleepbalk voor het geluid wordt op de plaats gezet, die bij die geluidssterkte hoort.
Bij openen van de pagina is die geluidssterkte echter nog niet bekend, en dus de positie van de knop ook niet, omdat het script die positie nog niet heeft berekend. In eerste instantie wordt de knop dan ook helemaal links gezet, alsof het geluid helemaal uit staat. Kort daarna wordt de knop dan op de juiste positie gezet. Oftewel: de knop springt iets na het openen van de pagina naar de juiste positie. Om dat verspringen te voorkomen, wordt de knop bij openen van de pagina verborgen met behulp van de inline-style `style="visibility: hidden;"`. Zodra de juiste positie is berekend, wordt de knop dan weer zichtbaar gemaakt met de inline-style `style="visibility: visible;"`.
Om de knop op de juiste plaats neer te kunnen zetten, is een absolute positie onmisbaar. Daarom wordt door het script de volgende inline-style ingevoegd: `style="position: absolute;"`.
Tenslotte wordt een inline-style `style="left: ...px;"` ingevoegd om de knop op de juiste plaats neer te zetten. Op de plaats van de puntjes wordt de door het script berekende waarde ingevuld. Bij elke verandering van de geluidssterkte, wordt deze waarde door het script aangepast. De knop beweegt dus mee als de geluidssterkte verandert.
Deze door het script ingevoegde css is niet te zien in de gewone broncode, maar alleen in de gegenereerde code.

`background: white;`
: Witte achtergrond.

`width: 5px;`
: Breedte.
Over de breedte staat iets hierboven gelijk onder `.sound-slider-button` meer.

`height: 14px;`
: Hoogte.

`border: black solid 1px;`
: Randje.
Over de border staat iets hierboven gelijk onder `.sound-slider-button` meer.

`border-radius: 5px;`
: Ronde hoeken.

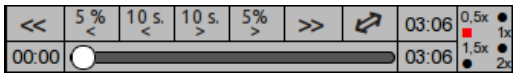
`top: -5px;`
: Op deze hoogte staat de knop precies midden boven de sleepbalk.

Knoppen afspelen (pagina 1)

Alle knoppen die met afspelen te maken hebben, behalve de Speel-pauzeerknop, staan binnen één gezamenlijke . Dit geeft de mogelijkheid aan deze het aria-attribuut

`role="group"` te geven, waardoor schermlezers weten dat het om een bij elkaar horende groep elementen gaat.

`.image`



De elementen met class=”image”. De knoppen Naar begin, Vijf procent terug, Tien seconden terug, Tien seconden verder, Vijf procent verder, Naar einde, Fullscreen, de snelheidsregeling, de weergave van totale, verstreken en resterende speelduur, en de sleepbalk voor weergave zitten samen in een met class=”image”. Hierdoor kan de bediening van de weergave door schermlezers als één groep worden gezien.

`width: 330px; height: 42px; float: left; border-left: black solid 2px; position: relative;`

Breedte en hoogte geven. Naar links floaten, zodat de naast de met de bediening van de geluidssterkte komt te staan. Randje.
Nakomelingen van een element kunnen alleen ten opzichte van dat element worden gepositioneerd, als dat element zelf een relatieve, absolute of fixed positie heeft. Omdat daar verder niets bij wordt opgegeven, heeft de hier opgegeven relatieve positie geen enkele invloed op de zelf.

`.videobox[data-sound="no"] .image {width: 430px; border-left: black solid 1px;}`

Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound=”no”] .play.

`.image button {background: transparent; width: 36px; height: 23px; float: left; border: black solid; border-width: 0 1px 1px 0; border-radius: 0; position: relative;}`

De <button>’s in de elementen met class=”image”. Dit zijn de knoppen die voor het afspelen worden gebruikt. Deze knoppen hebben voor een groot deel dezelfde css. Die wordt hier in één keer opgegeven.
De standaardweergave van een <button> verschilt per browser. Een groot deel van deze css zorgt er dan ook alleen voor dat ze er overal hetzelfde uitzien. Ze worden naar links gefloat, zodat ze naast elkaar komen te liggen.
Nakomelingen van een element kunnen alleen ten opzichte van dat element worden gepositioneerd, als dat element zelf een relatieve, absolute of fixed positie heeft. Omdat daar verder niets bij wordt opgegeven, heeft de hier opgegeven relatieve positie geen enkele invloed op de <button>’s zelf.

`.videobox[data-sound="no"] button {width: 49px;}`

Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound=”no”] .play.

`button.to-begin, button.to-end {width: 38px;}`

Voor deze elementen geldt ook de iets hierboven bij `.image button` opgegeven css, voor zover die hier niet wordt gewijzigd.
De knop Naar begin heeft een class=”to-begin”, de knop Naar einde heeft een class=”to-end”. Van beide wordt de breedte iets aangepast om de hele handel passend te krijgen.

```
.videobox[data-sound="no"] button.to-begin, .videobox[data-
    sound="no"] button.to-end {width: 48px;}
```

Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound=”no”] .play.

```
.to-begin::after, .to-end::after {content: "<<"; width: 32px;
    height: 23px; font-size: 18px; letter-spacing: -0.1em; line-
    height: 23px; position: absolute; top: 0; left: 0; speak:
    none;}
```

De knop Naar begin heeft een class=”to-begin”, de knop Naar einde heeft een class=”to-end”.

Met behulp van `::after` wordt een pseudo-element gemaakt, waarmee de haakjes op de knop worden weergegeven. De css hiervoor is voor beide knoppen hetzelfde. Maar voor `.to-end` moeten die haakjes uiteraard de andere kant op wijzen, dat wordt verderop geregeld. De ’<<’ bestaat uit twee gewone karakters, die met behulp van `letter-spacing` iets dichter bij elkaar worden geplaatst.

Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een relatieve, fixed of absolute positie heeft. Dat zijn hier `button.to-begin` respectievelijk `button.to-end`.

`speak: none;` is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

```
.to-end::after {content: ">>";}
```

Naar het einde. De knop om naar het eind te gaan heeft een class=”to-end”. Vrijwel alle css is gelijk hierboven al opgegeven bij `.to-begin::after, .to-end::after`. Alleen de tekst op de knop (de inhoud van `content`) moet worden aangepast.

```
.five-back::after, .ten-back::after, .ten-forward::after, .five-
    forward::after {content: "5 %\A<"; width: 32px; height: 23px;
    font-size: 12px; line-height: 9px; white-space: pre; padding-
    top: 2px; position: absolute; top: 0; left: 0; speak: none;}
```

De knop Vijf procent terug heeft een class=”five-back”, de knop Tien seconden terug heeft een class=”ten-back”, de knop Tien seconden verder heeft een class=”ten-forward” en de knop Vijf procent verder heeft een class=”five-forward”.

Met behulp van `::after` worden pseudo-elementen gemaakt, waarmee de tekst op de knoppen wordt weergegeven. De css hiervoor is voor alle vier de knoppen hetzelfde. Maar voor `.ten-back`, `.ten-forward` en `.five-forward` moet die tekst uiteraard worden aangepast, dat wordt iets hieronder geregeld.

De tekst op de knop bestaat uit gewone karakters: ’5% <’. Het enige vreemde is de ’\A’. Hiermee geef je aan dat een nieuwe regel begint. De knop is te smal om ’5% <’ op één regel te zetten, dus wordt ’%’ eronder gezet. De ’\’ zorgt ervoor dat de browser weet dat het teken erna, de ’A’, niet moet worden weergegeven, maar een opdracht is. Op deze manier een nieuwe regel beginnen werkt alleen, als ook `white-space: pre;` is opgegeven.

Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een relatieve, fixed of absolute positie heeft. Dat zijn hier respectievelijk `button.five-back`, `button.ten-back`, `button.ten-forward` en `button.five-forward`.

`speak: none;` is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

```
.videobox[data-sound="no"] .to-begin::after, .videobox[data-
    sound="no"] .five-back::after, .videobox[data-
    sound="no"] .ten-back::after, .videobox[data-
    sound="no"] .ten-forward::after, .videobox[data-
    sound="no"] .five-forward::after, .videobox[data-
    sound="no"] .to-end::after {width: 48px;}
```

Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound="no"] .play.

```
.ten-back::after {content: "10 s.\A<";}
```

Tien seconden terug. De knop om tien seconden terug te gaan heeft een class="ten-back". Vrijwel alle css is hierboven al opgegeven bij .five-back::after..., alleen de tekst op de knop (de inhoud van `content`) is anders.

```
.ten-forward::after {content: "10 s.\A>";}
```

Tien seconden verder. De knop om tien seconden vooruit te gaan heeft een class="ten-forward". Vrijwel alle css is hierboven al opgegeven bij .five-back::after..., alleen de tekst op de knop (de inhoud van `content`) is anders.

```
.five-forward::after {content: "5%\A>";}
```

Vijf seconden verder. De knop om vijf seconden vooruit te gaan heeft een class="five-forward". Vrijwel alle css is hierboven al opgegeven bij .five-back::after, alleen de tekst op de knop (de inhoud van `content`) is anders.

```
.fullscreen::before
```


Fullscreen. De knop om fullscreen af te spelen heeft een class="fullscreen". Met behulp van `::before` wordt een pseudo-element gemaakt, waarmee een symbool op de knop wordt gezet.

```
content: "\21d4";
```

Met behulp van `::before` wordt een dubbele pijl op de knop gezet. Die dubbele pijl is een gewoon teken, net zoals een letter, maar hij zit niet op het toetsenbord. Daarom wordt de pijl via een zogenaamde utf-8-code ingevoegd: `21d4`. Dat is, simpel gezegd, een volgnummer voor een bepaald teken. Omdat een computer nou eenmaal niet echt slim is, zou gewoon '21d4' worden weergegeven. Daarom staat er aan

Pijlmishandeling

De gebruikte pijl ⇔ heet 'left right double arrow'. Normaal genomen staat deze horizontaal, maar hier wordt de pijl met behulp van `transform` iets gedraaid.


Rechts de pijl, zoals die hoort te zijn. Links de aan morbide obesitas lijdende weergave in de hieronder genoemde browsers.

Eigenlijk wilde ik de hier afgebeelde dubbele pijl gebruiken: de 'left right arrow' ↔ (utf-8-code `2194;`) . Door twee van die pijlen met behulp van `transform` in verschillende richting te draaien, kan een kruis worden gemaakt met de vier pijlpunten naar de hoeken gericht. Dat is veel mooier, dan wat nu wordt gebruikt. Maar om een of andere reden geven in Android 4.4.2 Opera, Chrome en Android browser deze pijlen dermate wanstaltig vet weer, dat ze niet bruikbaar zijn.
(Android 4.4.2 kwam op de valreep binnen. In 4.0.2 gebeurde precies hetzelfde. Ergens moet een of andere ontwerper zich dus kennelijk esthetisch aangetrokken voelen tot zwaar mishandelde pijlen.)
Nog 'n aanvulling: in deze tekst wordt de dubbele pijl rood gekleurd in genoemde browsers. Iets is er mis met die pijl...

het begin een zogenaamde ’escape code’: `\`. Die geeft aan dat, wat erop volgt, geen letterlijke tekst is, maar een code.

`width: 31px; font-size: 26px; line-height: 18px; position: absolute; top: 0; left: 0;`
Wat aanpassingen om de pijl goed in de knop neer te zetten.

`speak: none;`
Is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

`-ms-transform: rotate(-30deg); -webkit-transform: rotate(-30deg); transform: rotate(-30deg);`
Hier staat in feite drie keer hetzelfde: `transform: rotate(-30deg);`
Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Met behulp van `transform` kunnen elementen o.a. verplaatst, vervormd en gedraaid worden, afhankelijk van de bijbehorende functie. In dit geval is dat `rotate()`: draai. Er wordt `-30deg` gedraaid, min 30 graden. Omdat er een minteken voor staat wordt niet naar rechts, maar naar links gedraaid, tegen de klok in.
Er zitten 360 graden in een cirkel: een draai van 360 graden zie je niet, omdat de pijl dan weer volledig recht staat. Je kunt dus ook 330 gewoon draaien: `330deg`, dat heeft precies hetzelfde effect als min 30 graden.

`.videobox[data-sound="no"] .fullscreen::before {width: 45px; font-size: 32px; line-height: 14px;}`
Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound=”no”] .play.

`.fullscreen[data-fullscreen="on"]::after`
Bij opening van de pagina wordt door het script aan de knop voor Fullscreen het attribuut `data-fullscreen="off"` toegevoegd. Als de video fullscreen moet worden afgespeeld, wordt door het script eerst gekeken, of de browser `requestFullScreen()` ondersteunt. Als dat zo is, wordt het fullscreen afspelen volledig intern door de browser afgehandeld.
Als de browser `requestFullScreen()` (nog) niet ondersteunt, wordt door het script ’off’ veranderd in ’on’: `data-fullscreen="on"`. Zodra fullscreen weer wordt verlaten en de video weer normaal wordt weergegeven, wordt ’on’ weer veranderd in ’off’.
(Los hiervan wordt door het script ook een kleine hoeveelheid css ingevoegd, waardoor de video ook fullscreen kan worden bekeken, als de browser `requestFullScreen()` niet kent. Wat er precies wordt ingevoegd e.d., is te vinden bij css die tijdelijk wordt ingevoegd, zoals alleen tijdens fullscreen afspelen.)
Omdat alleen tijdens het fullscreen afspelen de waarde van `data-fullscreen` ’on’ is, kan hierop in een selector worden getest:

`.fullscreen[data-fullscreen="on"]`: het element met class=”fullscreen”, maar alleen als `data-fulsscreen` de waarde ’on’ heeft.

`::after`: met behulp van een door `::after` aangemaakt pseudo-element kan iets op het scherm worden gezet. Als browsers fullscreen zelf afhandelen, vermelden die hoe je weer uit fullscreen kunt komen. Maar als de browser dat niet zelf kan afhandelen, gebeurt dat niet, omdat het script fullscreen alleen maar min of meer imiteert.

Daarom wordt een tekst op het scherm gezet met aanwijzingen, hoe je weer uit fullscreen kunt komen.
Speciaal voor zo'n melding is bovenaan het venster van de browser een ruimte van 30 px leeg gelaten, als fullscreen wordt afgespeeld. Die melding langzaam laten verdwijnen, zodat de video tot bovenaan kan worden weergegeven, is geen goed idee. Oudere browsers die `requestFullScreen()` niet kennen, zullen ook geen nieuwere css als `animation` kennen.

`content: "Raak video aan, klik op video of druk Escape in om video weer te verkleinen";`

De inhoud van de melding die op het scherm wordt gezet.

`background: white; width: 100%; height: 30px; font-size: 1.5em; text-align: center; padding: 2px 0;`

Wat opmaak voor de melding. Het script laat aan de bovenkant een strook van 30 px hoogte vrij, daarom is de hoogte van deze melding ook 30 px: dat past precies in de vrije ruimte.

`position: fixed; top: 0; left: 0; z-index: 200000;`

`.fullscreen`, de knop voor Fullscreen, is zelf absoluut gepositioneerd. Daarom kan de melding niet ten opzichte van deze knop worden gepositioneerd, want dan komt de melding niet bovenin het browservenster te staan. Daarom wordt fixed gepositioneerd: ten opzichte van het venster van de browser.
De z-index is onwijs hoog. Het script voegt om technische redenen een hele hoge z-index in voor <video>. De melding moet daarboven komen te staan, vandaar deze hoge z-index. (Meer over wat het script doet is te vinden bij css die tijdelijk wordt ingevoegd, zoals alleen tijdens fullscreen afspelen.)

Snelheidsregeling (pagina 1)

De snelheidsregeling bestaat uit vier radioknoppen met vier bijbehorende <label>'s. Deze staan samen in een met class="speed". Deze staat weer binnen `div.image`, de waar de hele bediening voor het afspelen van de video in staat (op de Speel-pauzeerknop na.)
Door de hele handel in één te zetten, kan aan deze het aria-attribuut `role="radiogroup"` worden gegeven, waardoor schermlezers weten dat het om een groep bij elkaar horende radioknoppen gaat.

`.speed {width: 35px; height: 42px; font-size: 0.6em; margin-left: -1px; border: none; border-left: black solid 1px; padding-top: 1px; position: absolute; right: 0;}`

De elementen met class="speed". Dit zijn de 's waar de bediening voor de snelheid in staat. De css is verder nauwelijks interessant.
(Uiteraard is deze snelheidsregeling veel te klein voor touchscreens. Op andere pagina's is soms een grotere snelheidsregeling aanwezig, die ook met bootwerkerskolenschoppen te bedienen zou moeten zijn.)

`.videobox[data-sound="no"] .speed {width: 41px;}`

Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound="no"] .play.

`.speed input`

Alle <input>'s die binnen een element met class="speed" staan. Dit zijn de radioknoppen die de snelheid regelen.

`position: absolute; left: -20000px;`

De ruimte voor de snelheidsregeling is heel klein. Daar passen gewoon geen vier knoppen in. Bovendien worden de radioknoppen in verschillende browsers verschillend weergegeven, met o.a. een verschillende grootte.
Verbergen met `display: none;` of `visibility: hidden;` is geen goed idee, want dan worden ze door schermlezers genegeerd. Daarom worden ze links buiten het scherm geparkeerd. Ze werken nog steeds, maar je ziet ze niet meer.
De bijbehorende <label>'s staan gewoon op het scherm, dus aanraken en klikken blijven gewoon werken. Niet op de knoppen – het is vrij zinloos om 'n meter links van het scherm in de lucht te gaan lopen graaien –, maar wel op de via het `for`-attribuut aan de knop gekoppelde <label>.
Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `div.speed`.

`.speed label {display: block; width: 33px; height: 9px; padding: 0 0 1px 2px;}`

Alle <label>'s die binnen een element met class="speed" staan. De <label>'s die bij de radioknoppen voor snelheidsregeling horen.
<label> is een inline-element: de <label>'s komen achter elkaar te staan en je kunt ze ook geen breedte e.d. geven. Met behulp van `display: block;` worden ze veranderd in blok-elementen, waardoor ze onder elkaar komen te staan en ze breedte, hoogte, e.d. kunnen krijgen.
Met deze waarden staat de snelheidsregeling in alle browsers min of meer fatsoenlijk.

`.speed-default-label, .speed-double-label {text-align: right; padding: 0 2px 1px 0;}`

Voor deze elementen geldt ook de gelijk hierboven bij `.speed label` opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class="speed-default-label" en class="speed-double-label". De <label>'s die horen bij de radioknoppen voor normale en dubbele snelheid.
Het script zet op alle vier de <label>'s een tekstje: '0,5x', '1x', '1,5x' en '2x'. (Deze teksten kun je wijzigen. Hoe dat moet, staat bij Text.) Deze tekstjes worden gewoon aan de linkerkant van de <label> neergezet.
De ruimte voor de snelheidsregeling is op de eerste pagina veel te klein. Om toch alle vier de <label>'s neer te kunnen zetten, wordt bij deze twee <label>'s de tekst naar rechts verschoven.

`.speed label::after {content: "\25cf"; font-size: 14px; position: absolute; top: -2px; right: 2px; speak: none;}`

Met behulp van `::after` wordt bij alle <label>'s binnen `.speed` een pseudo-element aangemaakt, waarmee een zwart rondje in de <label> wordt gezet. Omdat de radioknoppen zelf buiten het scherm zijn geparkeerd, wordt via de <label> een soort knop neergezet.
Dit rondje is een gewoon teken, net zoals een letter, maar het zit niet op het toetsenbord. Daarom wordt het via een zogenaamde utf-8-code ingevoegd: `25cf`. Dat is, simpel gezegd, een volgnummer voor een bepaald teken. Omdat een computer nou eenmaal niet echt slim is, zou gewoon '25cf' worden weergegeven. Daarom staat er aan het begin een zogenaamde 'escape code': `\`. Die geeft aan dat, wat erop volgt, geen letterlijke tekst is, maar een code.
Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `div.speed`.

`speak: none;` is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

```
.speed .speed-default-label::after {top: 8px; right: 25px;}
.speed .speed-one-half-label::after {top: 18px;}
.speed .speed-double-label::after {top: 28px; right: 25px;}
```

Voor deze elementen geldt ook de gelijk hierboven bij `.speed label::after` opgegeven css. Alleen de positie van de zwarte rondjes in de tweede, derde en vierde <label> moet worden aangepast, omdat ze anders allemaal over elkaar heen komen te staan.

```
.speed input:checked + label::after {content: "\25a0"; color: red;
    font-size: 13px; speak: none;}
```

Iets hoger is bij `.speed label::after` op alle <label>'s binnen `.speed` een zwart rondje neergezet. Als de radioknop aangevinkt is, moet dat op een of andere manier zichtbaar zijn. Omdat de radioknoppen zelf links buiten het scherm zijn geparkeerd, zie je de standaardmarkering voor een aangevinkte radioknop niet.
Daarom wordt deze markering op de <label> gezet, die bij de aangevinkte radioknop hoort:
`input:checked`: als de <input> is aangevinkt.
`+ label::after`: doe dan iets met het pseudo-element dat door `::after` bij de direct daaropvolgende <label> is aangemaakt.
`content: "\25a0";`
De utf-8-code voor een vierkantje. Het zwartje rondje wordt vervangen door een vierkantje, dat rood is. Omdat niet alleen de kleur verandert, maar ook de vorm, is dit ook duidelijk voor mensen die moeite met het onderscheiden van kleuren hebben.
`speak: none;`
Is gericht op schermlezers. Dit werkt nog in geen enkele browser, maar kwaad kan het niet, en misschien gaat het ooit werken.

Sleepbalk afspelen (pagina 1)

 De sleepbalk voor het afspelen werkt precies hetzelfde als de Sleepbalk voor het volume. Alleen moet je '0%' vervangen door 'begin van de video', en '100%' door 'eind van de video'.
Verder zijn er nog een paar kleine verschillen. De precieze werking van deze sleepbalk staat niet in `soundBeginSliding()`, maar in `imageBeginSliding()` in het script.
De class voor de met de sleepbalk is 'image-slider', de class voor de met de balk van de sleepbalk is 'image-slider-beam', en de class voor de met de knop van de sleepbalk is 'image-sliderbutton'.
De sleepbalk voor het afspelen ziet er iets anders uit dan de sleepbalk voor het geluid, dus de css is ook iets anders, maar dat heeft geen echte invloed op de werking. De css voor de balk van de sleepbalk:

```
.image .image-slider-beam {background: #555; clear: both;
    width: 211px; height: 6px; margin: 0 5px 0 35px; border:
    black solid 1px; border-radius: 5px; position: absolute;
    top: 29px; left: 6px;}
```

En de css voor de knop van de sleepbalk:

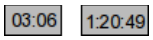
```
.image-slider-button {background: white; width: 14px; height:
    14px; border: black solid 1px; border-radius: 7px; top:
    -5px;}
```

Bij de sleepknop voor afspelen is er nog een klein verschil met de sleepknop voor de geluidssterkte. Omdat bij openen van de pagina de verstreken speelduur altijd op 0 staat, staat de sleepknop altijd links op de sleepbalk. (Bij de sleepknop voor de geluidssterkte hoeft dat niet zo te zijn.) Omdat deze sleepknop niet verspringt na openen van de pagina, is er – anders dan bij de sleepknop voor de geluidssterkte – geen reden deze knop tijdelijk te verbergen.

```
.videobox[data-sound="no"] .image-slider-beam {width: 288px;
    margin-left: 44px;}
```

Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound=”no”] .play.

Verstreken speelduur (pagina 1)



De verstreken speelduur wordt weergegeven in een met class=”elapsed”. Zodra de verstreken speelduur minder is dan 1 uur, wordt de speelduur weergegeven zoals links op de afbeelding, zonder uren. Zodra de speelduur meer is dan 59 minuten 59 seconden, wordt het aantal uren en een ’:’ voor de minuten en seconden geplaatst, zoals rechts op de afbeelding is te zien. Dit wordt door het script geregeld.
Als je weet dat de video langer dan ’n uur is, kun je daar in de css rekening mee houden door gewoon de iets breder te maken. Zodra de verstreken speelduur dan boven het uur komt, verschijnen de uren en wordt de hele gevuld.
Zolang de totale speelduur van de video nog onbekend is, wordt altijd --:-- ingevuld bij verstreken speelduur.
Zodra de speelduur van een video bekend is, wordt aan het element met class=”videobox” (de waar video, bediening, e.d. in staan) het attribuut `data-hour` toegevoegd. Als de video minimaal 1 uur duurt, is de waarde daarvan ’yes’: `data-hour="yes"`. Als de video korter dan ’n uur duurt, is de waarde ’no’: `data-hour="no"`.
Met behulp van de selector `.videobox[data-hour="yes"]` of z’n tegenhanger (`.videobox[data-hour="no"]`), kun je de weergave met behulp van css aanpassen.
De weergave zelf is niet aan te passen, die is altijd in het formaat ’00:00’ of ’0:00:00’. Aanpassen is ook niet nodig, omdat dit ’n internationaal bekend formaat is.
(Bij openen van de pagina wordt ook nog een attribuut `data-duration="unknown"` aan `.videobox` toegevoegd. Zodra de speelduur bekend is, verandert dit in `data-duration="known"`. Door dit in een selector te gebruiken, kun je eventueel hier ook nog het uiterlijk mee aanpassen.)

```
.elapsed {width: 37px; font-size: 0.8em; line-height: 19px;
    text-align: center; border-right: black solid 1px; position:
    absolute; bottom: 0; left: 0;}
```

De elementen met class=”elapsed”. Dit zijn ’s, waarbinnen de verstreken speelduur wordt weergegeven. De css voor deze is verder weinig interessant. Gewoon wat waarden om de speelduur enigszins fatsoenlijk op z’n plaats te krijgen.

```
.videobox[data-sound="no"] .elapsed {width: 47px;}
```

Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound=”no”] .play.

```
.aria-elapsed {position: absolute; left: -20000px;}
```

De elementen met class=”aria-elapsed” zijn ’s, die alleen dienen om in schermlezers de verstreken speelduur voor te lezen, als deze door de gebruiker is gewijzigd. Door ze ver

links buiten het scherm te parkeren worden ze wel gelezen door een schermlezer, maar verstoren ze de lay-out niet. Als je ze zou verbergen met `display: none;` zou een schermlezer ze volledig negeren, dus dat is geen goed idee.
Het voorlezen van de gewijzigde verstreken speelduur wordt volledig door het script geregeld.

Resterende speelduur (pagina 1)

Voor de resterende speelduur geldt precies hetzelfde als voor de Verstreken speelduur hierboven, maar dan 'omgekeerd'. Zolang de resterende speelduur 'n uur of langer is, is de weergave '0:00:00'. Zodra het minder dan 'n uur is, wordt de weergave '00:00'.
De class van deze is 'remaining'.

```
.remaining {width: 38px; font-size: 0.8em; line-height: 20px;
    text-align: center; border-left: black solid 1px; position:
    absolute; right: 36px; bottom: 0;}
```

De elementen met class="remaining". De 's waarbinnen de resterende speelduur staat. Ook deze css is verder weinig interessant.
Gelijk hieronder bij Speelduur staat voor deze nog een aanpassing voor iOS, waar de geluidsregeling ontbreekt en het uiterlijk dus iets anders is.

Speelduur (pagina 1)

Voor de totale speelduur geldt precies hetzelfde als voor de Verstreken speelduur hierboven, maar deze tijd verandert uiteraard niet. Als de speelduur 'n uur of langer is, is de weergave '0:00:00'. Als de speelduur minder dan 'n uur is, is de weergave '00:00'.
De class van deze is 'duration'.

```
.duration {width: 38px; font-size: 0.8em; line-height: 22px;
    text-align: center; border-bottom: black solid 1px; position:
    absolute; top: 0; right: 36px;}
```

De elementen met class="duration". De 's waarbinnen de totale speelduur staat.
Mijn hartelijke verontschuldigingen dat het niet opwindender wordt, maar ook deze css is dodelijk saai. Ach, altijd nog beter dan economie, want met deze cijfers weet je in ieder geval wat er gaat gebeuren, en bij economen weet je dat niet. Het schijnt dat astrologen nog betrouwbaarder zijn met hun voorspellingen...

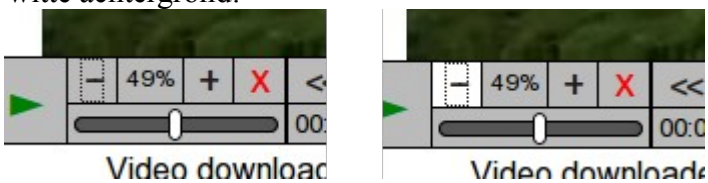
```
.videobox[data-sound="no"] .remaining, .videobox[data-
    sound="no"] .duration {width: 47px; right: 42px;}
```

Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast.
Meer uitleg bij .videobox[data-sound="no"] .play.

Focus (pagina 1)

Knoppen, links, e.d. kunnen focus hebben. Dat wil zeggen dat er bij gebruik van 'n bepaalde toets iets kan gebeuren. Als bijvoorbeeld een tekstveld focus heeft, kun je tekst gaan typen. Als bijvoorbeeld een link focus heeft en er wordt op Enter gedrukt, wordt de link gevolgd. Als bijvoorbeeld de knop Zachter van de videospeler focus heeft en de spatiebalk wordt ingedrukt, wordt de geluidssterkte minder. Welke toetsen precies werken bij welke knop, staat bij Werking van knoppen en sleepbalken.
Met behulp van de Tab-toets (of een soortgelijke toets) kunnen elementen die focus kunnen krijgen één voor één worden afgelopen (en met Shift + Tab ga je terug). Dit is belangrijk als iemand om een of andere reden de muis niet kan of wil gebruiken. Voor de volgorde waarin

de elementen die focus kunnen krijgen worden bezocht, is de tabindex belangrijk. Meer daarover is te vinden bij Tabindex.
Dit speelt uiteraard alleen, als je een toetsenbord gebruikt. Op touchscreens mag je aannemen dat de gebruiker weet, waar hij met zijn of haar vinger het scherm afraakt. (En als de gebruiker dat niet weet, valt te vrezen dat een simpele markering ook geen redding kan bieden.)
Als iemand op deze manier de videospeler bedient, links volgt, enz., is het belangrijk om te weten, waar je bent. Daarom krijgt een element dat focus heeft standaard een markering van de browser. Normaal genomen is dit een klein kadertje rondom het element met focus, zodat de gebruiker bijvoorbeeld weet welke link wordt gevolgd, als op Enter wordt gedrukt.
Die markeringen zijn heel belangrijk voor de toegankelijkheid, maar sommige lijken ontworpen door een slechtziende kleurenblinde met een ernstige persoonlijkheidsstoornis of, iets subtieler: ze zijn niet echt heel erg mooi. Voor links zijn ze prima, maar binnen de videospeler zijn ze lelijk. Bovendien vallen sommige, afhankelijk van de browser, nauwelijks op.
Daarom wordt in de videospeler de markering voor focus vervangen door een eigen markering: een witte achtergrond:



Hierboven staat links de knop met de standaardmarkering voor focus in Firefox, rechts met als extra de witte achtergrond. Firefox heeft een heel onopvallende markering: een stippellijn (deze is op de afbeeldingen ook te zien). Oftewel: je zoekt je ongans, welk kreng focus heeft. Opera op Linux bijvoorbeeld heeft daarentegen een door bovenvermelde getormenteerde ontwerper bedachte focus, die je niet kúnt missen, al wil je hem helemaal niet zien.
Een witte achtergrond valt in ieder geval op.

`.controls button:focus, input:focus, .sound-slider-beam:focus, .controls .image-slider-beam:focus`

Deze selectors beslaan de meeste bedieningselementen van de videospeler, als deze focus hebben. (De <label>'s voor de snelheid en de knoppen van de sleepbalken worden later apart behandeld bij .sound-slider-beam:focus .sound-slider-button, .speed input:focus + label en .image-slider-beam:focus .image-slider-button.)

`.controls button:focus` bestrijkt alle <button>'s. Omdat eerder alle <button>'s met `.controls button` een achtergrondkleur hebben gekregen, moet `.controls` hier ook worden gebruikt. Met alleen `button:focus` zou de selector te weinig specificiteit ('gewicht') hebben om de eerdere regel te overrulen.

`input:focus` bestrijkt alle <input>'s.

`.sound-slider-beam:focus` en `.image-slider-beam:focus` zijn nodig, omdat dit 's zijn. Ze worden dus niet bestreken door `button` of `input`. In de 's met deze classes zitten de balken van de sleepbalken voor geluid en beeld. De achtergrond van deze balken wordt wit, als ze focus hebben.
In Internet Explorer 9, 10 en 11 krijgen sleepbalk en sleepknop niet altijd een andere achtergrondkleur. Zie verder bij Bekende problemen (en oplossingen).

`background-color: white;`

Achtergrondkleur wit (alleen als het element focus heeft dus).

Firefox op OS X en Safari op OS X krijgen, als enige browsers, geen focus als je op ’n knop klikt. Dat is opzettelijk zo door Apple gedaan en heeft niets met dit voorbeeld te maken. Meer daarover bij Bekende problemen (en oplossingen).

`outline: none;`

De normale markering voor focus kan nu verdwijnen.
De markering voor focus is niet in elke browser weg te krijgen, althans: niet zonder veel extra moeite. De meest lelijke zijn met het hier gebruikte `outline: none;` te onderdrukken. Maar voor Firefox is ook nog het pseudo-element `::-moz-focus-inner` nodig. En voor Safari schijnt `!important` toegevoegd te moeten worden. Dat zou betekenen dat de regel hieronder veel en veel langer zou worden. Daar vind ik het niet belangrijk genoeg voor. Maar als je dat wel belangrijk vindt, kun je met enig zoeken op internet wel vinden hoe je de standaardmarkering voor focus in álle browsers weg krijgt.

```
.sound-slider-beam:focus .sound-slider-button {background: red;}
.image-slider-beam:focus .image-slider-button {background: red;}
```

Links de sleepbalk voor geluid, rechts de sleepbalk voor afspelen bij focus.

Eerste regel: geef een rode achtergrond aan het element met class=”sound-slider-button” dat binnen een element met class=”sound-slider-beam” ligt, maar alleen als `.sound-slider-beam` focus heeft.
Tweede regel: precies hetzelfde, maar dan voor de sleepbalk voor het afspelen.
Vertaald: geef de met de knop van de sleepbalk een rode achtergrond, maar alleen als de balk van de sleepbalk focus heeft.
In Internet Explorer 9, 10 en 11 krijgen sleepbalk en sleepknop niet altijd een andere achtergrondkleur. Zie verder bij Bekende problemen (en oplossingen).

```
.speed input:focus + label {background: white;}
```

Als een <input> die binnen een element met class=”speed” focus heeft, geef dan een witte achtergrond aan de direct op de <input> volgende <label>.
De snelheidsregeling zit binnen een met class=”speed”. Hij bestaat uit vier radioknoppen, die elk worden gevolgd door de bij de knop horende <label>. Door te kijken of de <input> focus heeft, kan iets worden gedaan met de bij de <input> horende <label>, in dit geval een witte achtergrond geven. Omdat de bij de <input> horende <label> altijd gelijk op de <input> volgt, zorgt de + ervoor dat altijd de juiste <label> wordt bereikt.

Pagina 2

De css die hier wordt beschreven, hoort bij de stylesheet ’afbeelding-103-2-dl.css’. Deze stylesheet hoort bij de tweede pagina met videospelers. Omdat het bij vijf pagina’s met grotendeels verschillende videospelers om heel veel css gaat, wordt hier alleen de css besproken, die afwijkt van die in ’afbeelding-103-1-dl.css’. De css daarin is te vinden bij Pagina 1.

Hoewel de videospelers op de tweede pagina wel wat variëren, hebben ze allemaal min of meer deze basisvorm.

Algemeen (pagina 2)

Verborgen bedieningselementen (pagina 2)

```
.controls .aria-volume, .controls .percentage, .sound-slider, main
    .controls .fullscreen, .speed, .controls .elapsed, .aria-
    elapsed, .duration {display: none;}
```

Deze regel zorgt ervoor dat een deel van de bedieningselementen volledig wordt verborgen, en ook door schermlezers wordt genegeerd. Sommige selectors zijn langer dan andere, omdat ze voldoende specificiteit (’gewicht’) moeten hebben om andere selectors te overrulen.

Achter de schermen blijft alles gewoon werken, want het script wordt niet veranderd. Als je de gegenereerde code bekijkt, zie je dat de elementen nog steeds gewoon aanwezig zijn, maar niet meer te zien zijn op het scherm.

Omdat in de selectors classes worden gebruikt, worden deze elementen op de hele pagina verborgen. Je kunt ze ook in één of enkele videospelers verbergen door gebruik te maken van id’s. Door gebruik te maken van een id kun je eventueel ook in één of enkele spelers verborgen elementen toch weer zichtbaar maken.

De volgende elementen worden verborgen:

`.controls .aria-volume`: voorlezen door schermlezers van nieuwe geluidssterkte na verandering.

`.controls .percentage`: weergeven van de geluidssterkte.

`.sound-slider`: sleepbalk voor geluidssterkte.

`main .controls .fullscreen`: knop Fullscreen.

`.speed`: volledige snelheidsregeling.

`.controls .elapsed`: weergave van verstreken speelduur.

`.aria-elapsed`: voorlezen door schermlezers van nieuwe verstreken speelduur na verandering.
`.duration`: totale speelduur.

Symbolen voor bedieningselementen (pagina 2)

Voor de afbeeldingen op de knoppen is een zogenaamd webfont gebruikt. (Een font is een vorm van een bepaalde lettersoort, bijvoorbeeld Helvetica Bold.) Normaal genomen moet een font op de computer van de gebruiker zijn geïnstalleerd, om het te kunnen gebruiken. Maar een webfont wordt, net als bijvoorbeeld een afbeelding, gedownload bij het openen van de pagina.
Normaal genomen bestaat een font uit (heel) veel tekens. Alle hoofd- en kleine letters, cijfers, accenten, enz. Soms zitten er duizenden tekens in. Voor deze videospelers zijn maar ’n paar symbolen nodig (om precies te zijn: twaalf. Het zou ’n beetje mallotig zijn om voor die paar symbolen een compleet font te gebruiken: honderden of duizenden tekens downloaden om er maar twaalf te gebruiken.
Dat hoeft gelukkig ook niet.
Voor het maken van dit webfont is gebruik gemaakt van de site icomoon.io. Op die site kun je kiezen uit een groot aantal icoontjes. De gekozen icoontjes, en niets meer, worden vervolgens opgeborgen in een font: icomoon. In dit font zit dus niet meer dan de twaalf gebruikte icoontjes. Om redenen waar ik nog op terugkom, zijn er twee formaten van dit font nodig. De een is 2 kB groot, de ander 2,3 kB, terwijl ’n beetje font al gauw honderden kB’s groot is, en soms zelfs vele MB’s. Deze fonts zijn kleiner dan bijvoorbeeld veel thumbnails, dus ze worden snel gedownload.

Er bestaat een soort systeem voor letters, cijfers, symbolen, enz., waarin elk teken een soort volgnummer krijgt: utf-8. Bij het maken van het webfont hebben alle symbolen een eigen volgnummer (utf-8-code) gekregen. Het font kan worden gebruikt door als het ware die utf-8-code aan te roepen.
Als je aan ’n symbool ’n willekeurige utf-8-code gaat geven, heb je kans op botsingen. Als ik de utf-8-code ’0041’ gebruik voor een luidsprekertje, worden alle hoofdletters A plotsklaps veranderd in luidsprekertjes. 0041 is namelijk al in gebruik voor de hoofdletter A. Het zou kunnen dat je je daar niet helemaal populair mee maakt, want dat leest wat lastig. Bovendien zijn er twaalf symbolen in gebruik, dus met ’n beetje geluk verander je de helft van het alfabet in de meest wilde moderne kunst.
Om dit te voorkomen heeft utf-8 een paar series gereserveerd voor eigen gebruik: Privat Use Area. De gebruikte utf-8-codes zitten in die reeks. Bijkomend voordeel is dat schermlezers utf-8-codes uit die reeksen niet voorlezen.
(Als je meer wilt weten over utf-8, kun je dat o.a. hier vinden: Charset, tekenset, utf-8, entiteiten, accenten, en dergelijke.)

```
@font-face {
     font-family: "icomoon";
     src: url("../103-fonts/icomoon.woff") format("woff"),
          url("../103-fonts/icomoon.ttf") format("truetype");
}
```

Om een webfont te kunnen gebruiken, moet de regel met `@font-face` beginnen. Wat daarna komt, staat tussen `{` en `}`. De eerste regel geeft een naam aan het font: icomoon. Nu kan deze naam binnen `font-family` worden gebruikt op precies dezelfde manier als een normaal font.
Daaronder staan twee regels met het adres van het webfont. Hiervandaan worden ze gedownload. Er staan twee regels, omdat er twee verschillende formaten van hetzelfde font

worden gedownload. Niet elke browser kan al met het nieuwste formaat woff overweg. Browsers bekijken elke url, en als er dan een formaat wordt gevonden dat bruikbaar is, wordt dat gekozen. Als een browser met woff uit de voeten kan, wordt dat dus gebruikt, en anders ttf.
Vaak worden nog vier formaten opgegeven, omdat oudere browsers geen van deze twee formaten kennen. Maar die browsers kennen ook het hele <video>-element niet, dus als je nog zulke oude browsers wilt ondersteunen, heb je hoe dan ook niets aan dit hele verhaal over <video>.
Als een font ook voor vet, cursief, e.d. wordt gebruikt, zijn vaak veel meer fonts en aanvullende regels code nodig. Hier speelt dat echter niet, omdat het webfont alleen in de meest normale vorm wordt gebruikt: standaardstijl en standaardgewicht.
Er wordt ook niet gekeken of het font al aanwezig is op de computer en dus niet gedownload hoeft te worden, omdat het onmogelijk is dat een zelf samengesteld font al aanwezig is. Als dit font eenmaal is gedownload, wordt het opgeslagen in de cache van de browser (als dat niet is uitgeschakeld door de gebruiker). Het hoeft dan niet opnieuw gedownload te worden. Dit werkt precies hetzelfde als bij dingen als afbeeldingen.

De symbolen zijn eigenlijk gewone tekens, net zoals een letter, maar ze zitten niet op het toetsenbord. Daarom worden ze via een zogenaamde utf-8-code ingevoegd, bijvoorbeeld : `e602`, waarbij `::after {content: …}` wordt gebruikt. Als je echter gewoon `content: "e602"` zou schrijven, zou gewoon ’e602’ worden weergegeven. Daarom staat er aan het begin een zogenaamde ’escape code’: `\`. Die geeft aan dat, wat erop volgt, geen letterlijke tekst is, maar een code: `content: "\e602"`.

Er is altijd een kleine kans dat het webfont niet kan worden gedownload. Een kleine kans, zeker omdat het zulke kleine bestandjes zijn, maar het kan. In dat geval zouden er helemaal geen symbolen op de knoppen zitten, wat de bediening er niet makkelijker op maakt. Voor een schermlezer maakt het niets uit, want die heeft toch niets aan die symbolen. In dit geval zijn mensen die goed kunnen zien dus in het nadeel. Mensen die een muis gebruiken zien nog steeds de title, als die aanwezig is. (Hoe je die title kunt wijzigen, is te vinden bij Title.)
Om dat enigszins op te lossen, wordt ook een achtergrond-afbeelding meegegeven aan de knoppen. Die achtergrond-afbeelding bestaat uit precies dezelfde symbooltjes als het webfont, samen op een doorzichtige achtergrond gezet en zelf ook doorzichtig: een gewone achtergrond-afbeelding dus (’iconenfont-fallback.png’). Zo’n afbeelding die uit meerdere afbeeldingen bestaat, heet een ’sprite’. Nu hoeft voor twaalf achtergrond-afbeeldingen maar één keer de server te worden aangeroepen. Met behulp van `background-position` zet je de juiste afbeelding dan op de goede plaats.
Als het webfont laadt, wat vrijwel altijd het geval zal zijn, zie je er vrijwel niets van, omdat het webfont boven de afbeelding staat. Als het webfont niet laadt, heb je in ieder geval de doorzichtige symbolen nog. De symbolen zijn doorzichtig, omdat het onmogelijk is de achtergrond-afbeeldingen en de symbolen uit het webfont exact boven elkaar te krijgen. Als ze doorzichtig zijn, zie je ze alleen, als je weet dat ze er zijn.
Dit werkt niet in alle browsers even goed, want bijvoorbeeld Firefox geeft in zo’n geval de utf-8-code weer, dus daar zie je de doorzichtige achtergrond-afbeelding niet goed. Maar in een aantal browsers werkt dit wel goed, en het is beter dan niets.

css voor vensters breder dan 700 px (pagina 2)

```
@media screen and (min-width: 700px) {
	main nav + p {
		-moz-column-count: 2;
		-moz-column-gap: 1em;
		-webkit-column-count: 2;
		-webkit-column-gap: 1em;
		column-count: 2;
		column-gap: 1em;
	}
}
```

Deze css is alleen geldig voor browservensters die minimaal 700 px breed zijn. De opbouw van de regel staat beschreven bij css voor vensters breder en hoger dan 480 px, het enige verschil is dat het hier om een minimumbreedte van 700 px gaat en dat er geen voorwaarde voor de hoogte van het venster geldt.

`main nav + p` is het inleidende tekstje bovenaan de pagina (de <p> die gelijk op de <nav> volgt, die binnen <main> staat). Als het browservenster breed wordt, worden de regels te lang om nog makkelijk te kunnen lezen. Daarom wordt de tekst in twee kolommen neergezet.
Hier staat in feite drie keer hetzelfde: `column-count: 2; column-gap: 1em;`
Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Internet Explorer 9 kent geen kolommen, dus daar staat gewoon ’n lange regel. Er zijn wel zogenaamde polyfills te vinden, waardoor ook dit fossiel met kolommen overweg kan, maar die moeite heb ik niet genomen. Als je zoekt op ’polyfill column’ kun je zo wat vinden. Of het probleemloos werkt, is een ander verhaal: goed testen is bij polyfills absoluut nodig, want er zit veel kaf onder het koren.

css voor vensters breder dan 1200 px (pagina 2)

```
@media screen and (min-width: 1200px) {
	main nav + p {
		-moz-column-count: 3;
		-webkit-column-count: 3;
		column-count: 3;
	}
}
```

Precies hetzelfde verhaal als gelijk hierboven voor browservensters breder dan 700 px, maar nu voor vensters breder dan 1200 px. Het aantal kolommen wordt hier 3.

Speel-pauzeerknop (pagina 2)

```
.play {background: transparent; width: 40px; height: 42px; float:
	left; border: none; border-right: black solid 1px; border-
	radius: 0; position: relative;}
```


De css is precies hetzelfde als die bij de Speel-pauzeerknop op de eerste pagina, alleen wordt deze knop relatief gepositioneerd, zodat de inhoud van `::after` iets verderop gepositioneerd kan worden ten opzichte van deze knop.

`.play::after`

Met behulp van `::after` wordt een pseudo-element gemaakt, waarmee het symbool op de knop wordt gezet.

`content: "\e602";`

De utf-8-code voor het pauzeren-symbool, zoals beschreven bij Symbolen voor bedieningselementen.

`background: url(../103-images/iconenfont-fallback.png) -2px 4px no-repeat;`

Voor het geval het webfont niet laadt, wordt hetzelfde symbool doorzichtig als achtergrond-afbeelding gebruikt, zoals beschreven bij Er is altijd een kleine kans...

`color: red; width: 35px; height: 43px; line-height: 38px; font-family: icomoon; font-size: 30px; position: absolute; top: 0; left: 1px;`

'n Hele serie eigenschappen om voor 'n goede plaatsing boven de knop te zorgen. Omdat het om een gewoon teken gaat, kunnen ook eigenschappen als `color`, `font-size`, e.d. worden gebruikt. Bij `font-family` kan gewoon de bij `@font-face` aan het webfont gegeven naam worden gebruikt.

```
.not-playing::after {content: "\e601"; background: url(../103-images/iconenfont-fallback.png) -44px 4px no-repeat; color: green;}
```

Behalve dat het hier om een groen driehoekje gaat, is het verhaal precies hetzelfde als gelijk hierboven bij `.play::after`.

Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Hierdoor kan, als de video niet speelt, een ander symbool worden getoond, dan wanneer de video wel speelt.

Knoppen volume (pagina 2)

```
.sound {width: 66px; height: 42px; float: left; position: relative;}
```

De 's met class="sound", de 's waar de geluidsregeling in staat. De css is grotendeels hetzelfde als bij Knoppen volume op de eerste pagina. Alleen de breedte is iets kleiner, omdat de sleepbalk en de knoppen voor Harder en Zachter ontbreken:

```
.sound button {background: transparent; display: block; width: 25px; float: left; text-align: center; border: black solid; border-width: 0 1px 1px 0; border-radius: 0; position: relative;}
```

De <button>'s binnen de elementen met class="sound".

Ook de css voor de knoppen voor de geluidsregeling is grotendeels hetzelfde als die bij Knoppen volume op de eerste pagina. Alleen is hier geen css voor 's aanwezig, want die zijn op deze pagina niet aanwezig in de voor de geluidsregeling. Ook ontbreekt een hoogte, omdat de drie knoppen voor de geluidsregeling elk een andere hoogte krijgen. Er is nog één ander verschil: de knoppen krijgen hier `display: block;`. <button>'s zijn inline-elementen, waardoor ze geen eigenschappen als breedte kunnen krijgen. Maar door ze naar links te floaten, worden ze 'n soort blok-element en kunnen ze toch breedte e.d. krijgen. Alleen krijgt iets verderop de knop voor Zachter `float: none;`, waardoor deze geen breedte e.d. meer kan krijgen. Om dit op te lossen, worden de inline-knoppen gewoon veranderd in blok-elementen.

`.sound .softer {height: 22px; float: none;}`

 Voor deze elementen geldt ook de gelijk hierboven bij `.sound button` opgegeven css, voor zover die hier niet wordt gewijzigd.
De elementen met class=”softer” binnen elementen met class=”sound”.
De knop voor Harder moet niet naast, maar onder deze knop voor Zachter komen te staan. Door met `float: none;` het floaten bij deze knop te verwijderen, wordt de knop voor Harder onder deze knop gezet in plaats van ernaast. Met een hoogte van 22 px passen de knoppen voor Harder en Zachter precies boven elkaar.

`.sound .louder {height: 21px;}`

 Voor deze elementen geld ook de iets hierboven bij `.sound button` opgegeven css, voor zover die hier niet wordt gewijzigd.
De elementen met class=”louder” binnen elementen met class=”sound”. De knop voor Harder wordt 1 px minder hoog dan de erboven staande knop voor Zachter, waardoor ze precies boven elkaar passen. Als je niet toevallig een adelaar bent, zie je dit verschil in hoogte niet.

`.softer::after, .louder::after`

Met behulp van `::after` wordt een pseudo-element gemaakt, waarmee een symbool op de knoppen voor Harder en Zachter wordt gezet. Het grootste deel van de css voor beide knoppen is hetzelfde, voor Harder wordt gelijk hieronder het symbool aangepast.

`content: "\e60a";`

De utf-8-code voor het zachter-symbool, zoals beschreven bij Symbolen voor bedieningselementen.

`background: url(../103-images/iconenfont-fallback.png) -182px no-repeat;`

Voor het geval het webfont niet laadt, wordt hetzelfde symbool doorzichtig als achtergrond-afbeelding gebruikt, zoals beschreven bij Er is altijd een kleine kans...

`width: 24px; font-family: icomoon; font-size: 16px; line-height: 18px; position: absolute; top: 0; left: 0;`

’n Hele serie eigenschappen om voor ’n goede plaatsing boven de knop te zorgen. Omdat het om een gewoon teken gaat, kunnen ook eigenschappen als `font-size`, e.d. worden gebruikt. Bij `font-family` kan gewoon de bij `@font-face` aan het webfont gegeven naam worden gebruikt.

`.louder::after {content: "\e609"; background: url(../103-images/iconenfont-fallback.png) -164px no-repeat;}`

De meeste css is gelijk hierboven bij `.softer::after, .louder::after` al opgegeven. Hier wordt alleen het symbool voor de knop Harder aangepast.

`.sound .mute {width: 40px; height: 43px; margin-top: -22px; border-right: none;}`

 Voor deze elementen geld ook de iets hierboven bij `.sound button` opgegeven css, voor zover die hier niet wordt gewijzigd.
De elementen met class=”mute” die binnen elementen met class=”sound” staan. De Aan-uitknop voor geluid wordt even groot als de Spelen-pauzerenknop. Met behulp van een negatieve marge wordt hij op de juiste plaats neergezet.

.mute::after

Met behulp van ::after wordt een pseudo-element gemaakt, waarmee het symbool op de knop wordt gezet.

content: "\e60c";

De utf-8-code voor het Geluid uit-symbool, zoals beschreven bij Symbolen voor bedieningselementen.

background: url(../103-images/iconenfont-fallback.png) -84px no-repeat;

Voor het geval het webfont niet laadt, wordt hetzelfde symbool doorzichtig als achtergrond-afbeelding gebruikt, zoals beschreven bij Er is altijd een kleine kans...

color: red; display: block; width: 40px; font-family: icomoon; font-size: 28px; line-height: 28px; margin-left: -4px;

'n Hele serie eigenschappen om voor 'n goede plaatsing boven de knop te zorgen. Omdat het om een gewoon teken gaat, kunnen ook eigenschappen als font-size, color, e.d. worden gebruikt. Bij font-family kan gewoon de bij @font-face aan het webfont gegeven naam worden gebruikt.

```
.mute.muted::after {content: "\e60b"; background: url(../103-images/iconenfont-fallback.png) -124px no-repeat; color: green;}
```

Behalve dat het hier om een groene luidspreker gaat, is het verhaal precies hetzelfde als gelijk hierboven bij .mute::after.

Aan de Aan-uitknop voor geluid wordt door het script een class="muted" toegevoegd, als het geluid uitstaat. Hierdoor kan, als het geluid uit staat, een ander symbool worden getoond, dan wanneer het geluid aanstaat.

Knoppen afspelen (pagina 2)

```
.image {width: 372px; height: 42px; float: left; border-left: black solid 2px; position: relative;}
.videobox[data-sound="no"] .image {width: 439px; border-left: black solid 1px;}
.image button {background: transparent; width: 52px; height: 23px; float: left; border: black solid; border-width: 0 1px 1px 0; border-radius: 0; position: relative;}
.videobox[data-sound="no"] button {width: 64px;}
button.to-begin, button.to-end {width: 58px;}
```

In bovenstaande css is het enige verschil met de css voor de Knoppen voor afspelen op de eerste pagina de breedte van de elementen. Omdat niet alle bedieningselementen worden gebruikt, moet de breedte van de wel gebruikte elementen worden aangepast.

.to-begin::after, .five-back::after, .ten-back::after, .ten-forward::after, .five-forward::after, .to-end::after

Met behulp van ::after worden pseudo-elementen gemaakt, waarmee symbolen op de knoppen voor het afspelen worden gezet. Omdat de meeste css voor alle knoppen hetzelfde is, wordt die hier in één keer opgegeven. Waar dat nodig is, wordt de css later aangepast voor bepaalde knoppen.

```
content: "\e605";
```
De utf-8-code voor het Terug-naar-begin-symbool, zoals beschreven bij Symbolen voor bedieningselementen.

```
background: url(../103-images/iconenfont-fallback.png) -200px -6px no-repeat;
```
Voor het geval het webfont niet laadt, wordt hetzelfde symbool doorzichtig als achtergrond-afbeelding gebruikt, zoals beschreven bij Er is altijd een kleine kans...

```
display: block; height: 23px; font-family: icomoon; font-size: 18px; line-height: 19px;
```
'n Hele serie eigenschappen om voor 'n goede plaatsing boven de knop te zorgen. Omdat het om een gewoon teken gaat, kunnen ook eigenschappen als `font-size` e.d. worden gebruikt. Bij `font-family` kan gewoon de bij `@font-face` aan het webfont gegeven naam worden gebruikt.

```
.videobox[data-sound="no"] .to-begin::after, .videobox[data-sound="no"] .five-back::after, .videobox[data-sound="no"] .ten-back::after, .videobox[data-sound="no"] .ten-forward::after, .videobox[data-sound="no"] .five-forward::after, .videobox[data-sound="no"] .to-end::after {width: 40px;}
.videobox[data-sound="no"] .to-begin::after {background-position: -205px}
```
Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound="no"] .play.

```
.five-back::after {content: "\e603"; background: url(../103-images/iconenfont-fallback.png) -310px -6px no-repeat;}
.ten-back::after {content: "\e607"; background: url(../103-images/iconenfont-fallback.png) -420px -6px no-repeat;}
.ten-forward::after {content: "\e608"; background: url(../103-images/iconenfont-fallback.png) -470px -6px no-repeat;}
.five-forward::after {content: "\e604"; background: url(../103-images/iconenfont-fallback.png) -366px -6px no-repeat;}
.to-end::after {content: "\e606"; background: url(../103-images/iconenfont-fallback.png) -248px -6px no-repeat;}
```
De meeste css is iets hierboven al opgegeven bij `.to-begin::after,...` Hier wordt alleen nog het juiste symbool op de juiste knop gezet.

```
.videobox[data-sound="no"] .to-end::after {background-position: -253px;}
```
Op iOS is geen geluidsregeling aanwezig en moeten maten e.d. daaraan worden aangepast. Meer uitleg bij .videobox[data-sound="no"] .play.

Sleepbalk afspelen (pagina 2)

```
.image .image-slider-beam {background: #555; clear: both; width:
     360px; height: 6px; margin: 0 5px 0 35px; border: black solid
     1px; border-radius: 5px; position: absolute; top: 29px; left:
     -30px;}
.videobox[data-sound="no"] .image-slider-beam {width: 410px;
     margin-left: 44px;}
.image-slider-button {background: white; width: 14px; height:
     14px; border: black solid 1px; border-radius: 8px; top: -5px;}
```

Zelfde als op de eerste pagina bij Sleepbalk afspelen. Alleen de breedte voor de balk van de sleepbalk wordt aangepast.

Resterende speelduur (pagina 2)

```
.remaining {width: 48px; height: 22px; float: left; font-size:
     0.9em; line-height: 21px; text-align: center; border-bottom:
     black solid 1px;}
.videobox[data-sound="no"] .remaining {width: 55px;}
```

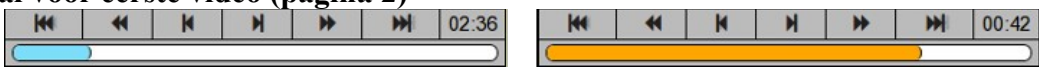
Weinig spannend allemaal. Omdat de plaatsing van de met de resterende speelduur anders is dan op de eerste pagina, moet van alles in de css worden veranderd.

Focus (pagina 2)

```
.controls button:focus, input:focus, .controls .image-slider-
     beam:focus {background-color: white; outline: none;}
.image-slider-beam:focus .image-slider-button {background: red;}
```

Het aangeven welke knop focus heeft, werkt op precies dezelfde manier als bij Focus op de eerste pagina, behalve dat de selectors voor de sleepbalk voor geluidssterkte en de selectors voor de snelheidsregeling ontbreken, omdat die knoppen op deze pagina zijn verborgen.

Speciaal voor eerste video (pagina 2)



Rechtsboven staat de resterende speelduur, die correspondeert met de lengte van de gekleurde achtergrond van de sleepbalk. Hoe verder de video is afgespeeld, hoe langer de gekleurde achtergrond van de balk.

De css die voor alle videospelers hetzelfde is, maakt gebruik van algemene selectors, zoals classes, die elementen van meerdere videospelers bestrijken. Door gebruik te maken van id's, kun je één of enkele videospelers een ander uiterlijk geven.

De eerste videospeler heeft een sleepbalk voor afspelen, die vanaf links wordt gevuld met blauw. Als de sleepbalk focus heeft, is de achtergrond oranje. (Oranje en lichtblauw, een wonderschone combinatie! Wie riep daar dat ik geen kleurgevoel heb?) Wat op papier niet is te zien: als de video speelt, fluctueert het blauw of oranje langzaam.

`#videobox-1 .image-slider-beam`

Voor dit element geldt ook de eerder bij .image .image-slider-beam opgegeven css, voor zover die hier niet wordt veranderd.

Elementen met class = "image-slider-beam" die binnen het element met id="videobox-1" liggen. `#videobox-1` is de , waarbinnen de eerste videospeler zit. Er is maar één element binnen `#videobox-1` met een class="image-slider-beam": de balk van de sleepbalk.

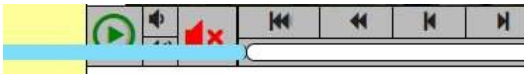
`background: white;`

Witte achtergrond.

`height: 11px;`

De eerder bij .image .image-slider-beam opgegeven hoogte van 6 px is te laag voor dit effect.

`overflow: hidden;`



Hoewel er geen sleepknop is te zien op deze sleepbalk, is er wel degelijk 'n knop aanwezig. Je ziet die knop echter niet, omdat de knop geen breedte, geen border, helemaal niets heeft. De natte droom van de NSA: aanwezig, maar niet zichtbaar. (Ik heb het hier steeds over een 'knop', maar feitelijk is dat dus gewoon een , met behulp van css vermomd als knop.)
De blauwe (bij focus oranje) kleur is de achtergrond van een pseudo-element, dat met behulp van `::before` aan de sleepknop is toegevoegd. Die `::before` heeft wel een breedte.
Aan het begin van de video staat de (onzichtbare) sleepknop helemaal links. De bijbehorende `::before` dus ook. Standaard staat `overflow` op `visible`, zodat ook tekst e.d. die niet binnen een element past, in ieder geval wordt weergegeven. Normaal genomen is dat ook prima. Maar hier wordt in dat geval `::before`, die links buiten de met de sleepbalk staat, zichtbaar. Wat de ietwat eigenaardige blauwe puist op de afbeelding oplevert. Met behulp van `overflow: hidden;` wordt alleen het deel dat binnen de sleepbalk staat zichtbaar.

`border-radius: 6px;`

Ronde hoeken.

`top: 26px;`

Omdat de sleepbalk iets hoger is dan opgegeven bij .image .image-slider-beam, moet de sleepbalk iets hoger komen te staan, dan daar is opgegeven. Nu staat de sleepbalk netjes verticaal in het midden.

```
#videobox-1 .image-slider-button {width: 0; border: none;}
```

Voor dit element geldt ook de eerder bij .image-slider-button opgegeven css, voor zover die hier niet wordt veranderd.
Elementen met class = "image-slider-button" die binnen het element met id="videobox-1" liggen. `#videobox-1` is de , waarbinnen de eerste videospeler zit. Er is maar één element binnen `#videobox-1` met een class="image-slider-button": de sleepknop van de sleepbalk.
Door de knop een breedte van 0 px te geven, wordt hij onzichtbaar. Als dan ook nog de bij .image-slider-button opgegeven border wordt weggehaald, is de knop volledig onzichtbaar.

```
@-webkit-keyframes pulse {
    0%, 100% {background: #4ccce8;}
    50% {background: #7eddf9;}
}
@keyframes pulse {
    0%, 100% {background: #4ccce8;}
    50% {background: #7eddf9;}
}
```

Hier staat in feite twee keer hetzelfde, als je `-webkit-` bij de bovenste regel even wegdenkt. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Met behulp van `@keyframes` kun je een animatie maken. Lengte e.d. van de animatie wordt later opgegeven met behulp van `animation`.

`@keyframes`: dat is gewoon de naam van de eigenschap. De eigenlijke animatie staat tussen `{` en `}`, daarom wordt aan het begin een `@` gezet.

`pulse`: de naam van de animatie. Deze wordt later bij `animation` gebruikt, zodat duidelijk is welke animatie gebruikt moet worden.

`@keyframes pulse {…}`: een animatie met de naam 'pulse'. Tussen de `{` en `}` komt de eigenlijke animatie te staan. De naam had ook kerstman of Marx kunnen zijn, dat is helemaal vrij.

De eigenlijke animatie wordt in procenten opgegeven (de trefwoorden `from` en `to` kunnen ook worden gebruikt, maar dat doe ik nooit.) Bij `animation` verderop wordt opgegeven dat de animatie 2 seconden moet duren.
Bij 0% (aan het begin van de 2 seconden) en bij 100% (bij het eind van de 2 seconden) moet de achtergrond de kleur `#4ccce8` krijgen. Dat is lichtblauw. Op 50% van de animatie, dat is halverwege de 2 seconden, moet de achtergrond de kleur `#7eddf9` krijgen: 'n heel klein beetje donkerder lichtblauw. Het gaat hier om de achtergrond van het element dat verderop met behulp van `animation` deze animatie gaat aanroepen.
De achtergrond verandert dus over een tijdsbestek van 2 seconden van lichtblauw naar iets donkerder lichtblauw terug naar lichtblauw. En omdat verderop is opgegeven dat de animatie eindeloos doorgaat, wordt dan weer naar donkerder lichtblauw gegaan, naar lichtblauw, enz.
(Er zit wel 'n grens aan de animatie: de animatie speelt alleen als de sleepbalk focus heeft. Dat wordt hieronder allemaal geregeld. Net als een andere achtergrondkleur bij focus.)

Omdat hier specifiek de achtergrond is opgegeven, wordt alleen de achtergrond veranderd door de animatie. Een animatie kan veel meer eigenschappen aanpassen, maar dat is hier niet de bedoeling.

`#videobox-1 .image-slider-button::before`

Elementen met class = "image-slider-button" die binnen het element met id="videobox-1" liggen. `#videobox-1` is de , waarbinnen de eerste videospeler zit. Er is maar één element binnen `#videobox-1` met een class="image-slider-button": de sleepknop van de sleepbalk.
Met behulp van `::before` wordt een pseudo-element gemaakt. Dit pseudo-element wordt van een achtergrondkleur voorzien en beweegt mee met de (onzichtbare) met de sleepknop. Hierdoor ontstaat een sleepbalk, waarvan de achtergrond steeds verder wordt gekleurd, naarmate de video verder afspeelt.

`content: "";`

De inhoud van het pseudo-element. Dat is evenveel als na jaren van graaiende bestuurders en bankiers en bezuinigingen het spaargeld van een gemiddelde uitkeringsgerechtigde: niets, nada, noppes, leeg. Maar `content` moet wel aanwezig zijn, dus krijgt het een inhoud van niets. Wat je aangeeft met twee aanhalingstekens met niets ertussen.

`-webkit-animation: pulse 2s infinite; animation: pulse 2s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: pulse 2s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
`pulse` is de naam van de gelijk hierboven bij `@keyframes` opgegeven animatie. Bij die animatie is het veranderen van de achtergrond opgegeven, en dat wordt hier daadwerkelijk uitgevoerd.

De speelduur is 2 seconden. `infinite` wil zeggen dat de animatie eindeloos doorgaat. (In werkelijkheid speelt hij alleen als de sleepbalk focus heeft, dat wordt hieronder geregeld.) Andere waarden, zoals sneller aan het begin en einde, worden niet gebruikt.

`background: #7eddf9;`

Oudere browsers, waaronder Internet Explorer 9, kennen `@keyframes` niet en negeren die eigenschap. En dus ook de daar opgegeven achtergrondkleur, waardoor ze geen achtergrondkleur zouden krijgen. Daarom wordt hier ook nog op de 'gewone' manier een achtergrondkleur opgegeven.

`width: 380px;`

De sleepbalk is 360 px breed. Met 380 px breedte kan dit pseudo-element dus ruim de hele sleepbalk afdekken, als de video aan het eind is.

`height: 11px;`

De sleepbalk is 11 px hoog, dus dit element – met de daarin zittende achtergrond – moet ook 11 px hoog worden.

`border-right: black solid 1px;`

Zwarte border aan rechterkant. Die border staat precies op de grens tussen de blauwe (of oranje) en de witte achtergrond.

`border-radius: 0 5px 5px 0;`

Border ronde hoekjes geven, die even groot zijn als de ronde hoekjes van de sleepbalk.

`position: absolute;`

Om het pseudo-element op de goede plaats neer te kunnen zetten.
Van zichzelf is dit element een inline-element. Dat betekent o.a. dat je het geen breedte kunt geven. Door het element absoluut te positioneren, verandert het ook in een soort blok-element, waardoor eigenschappen als hoogte en breedte gebruikt kunnen worden.

`top: 5px;`

Op de juiste hoogte zetten.

`right: 0;`

Meestal zet je zo'n pseudo-element vanaf links neer, dus achter de (onzichtbare) sleepknop. Maar dan zou het juist zichtbaar zou zijn, met de achtergrond, als de video aan het begin staat. Hiermee wordt het rechts van de sleepknop neergezet. (Ja, dat klopt. Als je hier `left: 0;` gebruikt, en je verandert `border-right` in `border-left`, en je draait de hoeken om bij `border-radius`, dan krijg je het omgekeerde effect: het witte deel van de sleepbalk wordt juist langer, naarmate de video verder is afgespeeld.)

`#play-1.not-playing ~ #image-1 .image-slider-button::before`

Als de video niet speelt, wordt door het script de class 'not-playing' toegevoegd aan de Speel-pauzeerknop. Hier kun je gebruik van maken om het uiterlijk aan te passen, afhankelijk van of de video wel of niet speelt.

`#play-1.not-playing`: het element met id="play-1" en class="not-playing". De Speel-pauzeerknop van de eerste video, als deze niet speelt.

`~ #image-1`: het element met id="image-1" dat ergens in de html volgt op het voor ~ staande element. Enige eis is verder dat `#image-1` en het element waar het op moet volgen dezelfde ouder hebben. En dat is hier zo: ze hebben beide als ouder `div#controls-1`. Daarom is `#image-1` ook nodig. Het gaat eigenlijk om `.image-slider-button`, maar die heeft niet dezelfde ouder als `#play-1`.

`.image-slider-button::before`: het met behulp van `::before` gemaakte pseudo-element bij de sleepknop.

Alles bij elkaar: doe iets met het pseudo-element bij de sleepknop die binnen het element met id=”image-1” ligt, waarbij `#image-1` in de html ergens op `#play-1` moet volgen, maar alleen als `#play-1` ook de class ’not-playing’ heeft.

`#videobox-1 .image-slider-button::before`

Iets minder langdradig: doe iets met het hierboven bij #videobox-1 .image-slider-button::before gemaakte pseudo-element bij de sleepknop, maar alleen als de video niet speelt.

`-webkit-animation: none; animation: none;`

Hier staat in feite twee keer hetzelfde: `animation: none;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Het weinig indrukwekkende resultaat van de wel indrukwekkende selector: stop de animatie. Oftewel: de achtergrond verandert niet van kleur, als de video niet speelt.

```
@-webkit-keyframes focus-pulse {
    0%, 100% {background: #ffa500;}
    50% {background: #dd8300;}
}
@keyframes focus-pulse {
    0%, 100% {background: #ffa500;}
    50% {background: #dd8300;}
}
```

Precies hetzelfde verhaal als hierboven bij @webkit-keyframes pulse, alleen is de naam van de animatie hier geen ’pulse’, maar ’focus-pulse’ en zijn de achtergrondkleuren niet lichtblauw en iets donkerder lichtblauw, maar oranje en iets donkerder oranje.

```
#videobox-1 .image-slider-beam:focus .image-slider-
    button::before {-webkit-animation: focus-pulse 2s infinite;
    animation: focus-pulse 2s infinite; background: #ffa500;}
```

Voor het door `::before` gemaakte pseudo-element is alle css al opgegeven bij #videobox-1 .image-slider-button::before. De enige verandering in de css is de gebruikte animatie: hier wordt ’focus-pulse’ gebruikt in plaats van de eerder opgegeven ’pulse’. De achtergrond is oranje in plaats van lichtblauw.

Verder is de selector anders. In het midden is er bijgekomen `.image-slider-beam:focus`: als de balk van de sleepbalk focus heeft (er is op geklikt of hij is met behulp van de Tab-toets bereikt.) Deze selector geldt dus alleen, als de sleepbalk focus heeft.

Normaal genomen is de achtergrond van het door `::before` gemaakte pseudo-element lichtblauw, maar als de sleepbalk focus heeft is de achtergrond oranje.

En voor de browsers die `animation` niet kennen, wordt de achtergrondkleur weer met `background` opgegeven.

```
#play-1.not-playing ~ #image-1 .image-slider-beam:focus .image-
    slider-button::before {-webkit-animation: none; animation:
    none; background: #ffa500;}
```

Dit is vrijwel hetzelfde als bij #play-1.not-playing ~ #image-1 .image-slider-button::before. Het laat de animatie stoppen, als de video niet speelt. Alleen is hier `.image-slider-`

`beam:focus` aan de selector toegevoegd, waardoor dit alleen werkt als de sleepbalk focus heeft.

Speciaal voor tweede video (pagina 2)

De css die voor alle videospelers hetzelfde is, maakt gebruik van algemene selectors, zoals classes, die elementen van meerdere videospelers bestrijken. Door gebruik te maken van id's, kun je één of enkele videospelers een ander uiterlijk geven.
De tweede videospeler geeft in de linkerbovenhoek en op de knop van de sleepbalk de verstreken speelduur weer in procenten. Op de afbeelding is 56% van de video afgespeeld. Je kunt dit percentage in principe overal neerzetten, het hoeft niet in de linkerbovenhoek. Verder is de achtergrondkleur van de Speel-pauzeerknop veranderd en heeft de sleepknop van de sleepbalk een afwijkende breedte.

```
#videobox-2 {position: relative;}
```

De met id="videobox-2" is de , waarbinnen de tweede videospeler zit.
Het hieronder met behulp van `::before` aangemaakte pseudo-element moet absoluut worden gepositioneerd ten opzichte van deze . Dat kan alleen als die zelf absoluut, relatief of fixed is gepositioneerd, wat hier wordt gedaan. Omdat verder niets wordt opgegeven, heeft dit geen enkele invloed op de zelf.

```
#videobox-2::before
```

Met behulp van `::before` wordt bij het element met id="videobox-2" een pseudo-element aangemaakt, dat hier wordt gebruikt om de verstreken speelduur in procenten weer te geven. In procenten, want dat wordt namelijk door het script ingevoegd als data-attribuut: `data-played`. Als je de verstreken speelduur in tijd wilt weergeven, kan dat al, want daar is een speciaal element voor (een met class="elapsed").

```
content: attr(data-played) "%";
```

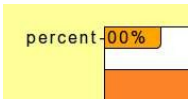
Met `content` wordt de inhoud van het pseudo-element opgegeven. Dat is hier de waarde van een attribuut (daar staat `attr()` voor) met de naam `data-played`. Een attribuut dat met `data-` begint, is een zelf bedacht attribuut. Daar is `data-` aan het begin voor gereserveerd. `data-played` wordt door het script toegevoegd aan de elementen met een class="videobox", dus ook aan `#videobox-2`, want die heeft behalve de id 'videobox-2' ook de class 'videobox'.

De waarde van `data-played` is de verstreken speelduur, uitgedrukt in 'percent-' gevolgd door het percentage. Bij 10% bijvoorbeeld is dat 'percent-10'. Tijdens het afspelen van de video wordt dit percentage voortdurend bijgewerkt.
`attr(data-played)` laat dus bij een video die voor 10% is afgespeeld, 'percent-10' zien.
Als laatste staat achter `content` nog `"%"`. Dit wordt gewoon achter de inhoud van `data-played` gezet, dus bij een video die voor 10% is afgespeeld wordt de volledige `content` 'percent-10%'. Waar vermoedelijk alleen tweetalig opgevoede mensen met heimwee naar hun opvoeding enthousiast over zijn, dus hieronder wordt via snode trucs 'percent-' er weer afgesloopt, waardoor alleen '10%' overblijft.

`width: 42px;`

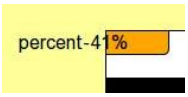
Breedte van het pseudo-element.

`overflow: hidden;`



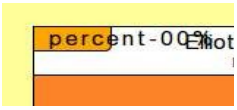
Normaal genomen wordt alle inhoud van een element weergegeven, ook als dit er niet in past. Normaal genomen wil je dat ook, want het is dan misschien niet zo netjes, maar je ziet in ieder geval alles.
In dit geval wil ik alleen zien, wat in het element past. Door 'percent-' buiten het element te zetten, kan ik op die manier 'percent-' verbergen. Op de afbeelding is `overflow: hidden;` even weggehaald en kun je zien, dat 'percent-' inderdaad gewoon aanwezig is, en alleen maar wordt verborgen.

`letter-spacing: 2px;`



Hoewel je 'percent-' niet ziet (als `overflow` op `hidden` staat), wordt het percentage neergezet vanaf de letter 'p'. Die 'p' wordt op een bepaalde plek neergezet. Zonder de ruimte tussen de tekens te verhogen, komt het percentage buiten het pseudo-element te staan en valt grotendeels weg. Met een kleine verhoging komt het deel na 'percent-' precies goed te staan.
Het was nogal lastig dit in alle browsers goed op z'n plaats te krijgen. Dit is één van de daarvoor gebruikte eigenschappen. Daar is trouwens geen geniale truc voor: gewoon uitproberen tot het overal goed staat.
Op de afbeelding is de letter-spacing even weggehaald, waardoor het grootste deel van het getal links buiten het oranje vakje valt.

`text-indent: -84px;`



Zet de tekst 84 px naar links. Als dat niet gebeurt, krijg je het resultaat dat hiernaast is te zien. Ook mooi hoor, daar niet van, maar toch misschien iets té apart.
In combinatie met `overflow: hidden;` verbergt dit 'percent-'.

`margin-left: -241px;`

Het hele pseudo-element 241 px naar links zetten.

Het percentage moet linksboven in de hoek van de videospeler komen te staan. En daar ontstaat een probleem. De titel e.d. zijn even breed als de video, zodat je 'n mooi blok krijgt. Maar `div#videospeler-2`, waar de hele handel in staat, is even breed als 50% van het browservenster. Op de afbeelding is even 'n border rondom `#videospeler-2` gezet, wat de afstand tussen de buitenkant van `#videospeler-2` en titel, video, e.d. zichtbaar maakt.
Het met `::before` gemaakte pseudo-element hoort bij `#videospeler-2`. Dat kan niet anders, de inhoud van `data-played` nodig is, en dat is nou eenmaal een attribuut van `#videospeler-2`. Maar het moet op de juiste plaats binnen de videospeler staan, en die heeft andere maten dan `#videospeler-2`. Bovendien is

de breedte van de video e.d. een vaste maat, terwijl `#videospeler-2` een breedte van de helft van het browservenster heeft, en dus varieert.
Maar er is redding. De video, titel, enz. staan in het midden van `#videospeler-2`, ongeacht de breedte daarvan.
Hieronder wordt met behulp van `left: 50%;` het pseudo-element horizontaal halverwege `#videospeler-2` gezet. Als je het nu de halve breedte van de video (met een verrekening voor borders e.d.) terug naar links zet, staat het precies linksboven in de hoek. Ongeacht de breedte van het browservenster, want halverwege (`left: 50%;`) is altijd halverwege, hoe breed `#videospeler-2` ook is.

`border: black solid 1px; border-bottom-right-radius: 5px;`
Randje geven en rechtsonder een ronde hoek.

`padding-left: 10px;`
Een van de eigenschappen die nodig zijn om het in alle browsers op de juiste plaats te krijgen.

`position: absolute; top: 0; left: 50%;`
Op de goede plaats zetten. Er wordt gepositioneerd ten opzichte van `div#videospeler-2`, want dat is de eerste voorouder die een absolute, relatieve of fixed positie heeft. De `left: 50%;` wordt iets hierboven bij `margin-left: -241px;` verklaard.

`background: orange;`
Toch heb ik nog iets masochistisch, diep van binnen, kennelijk. Waarom toch zoveel oranje? De kleur die ik met voorsprong het lelijkst vind en ik ben nog republikein ook. Nou ja, het valt wel op, en daar gaat het hier om.

`#play-2`


Voor dit element geldt ook de eerder bij Speel-pauzeerknop opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”play-2”. Dat is de Speel-pauzeerknop van de tweede video. Deze krijgt een andere achtergrondkleur. Voor de duidelijkheid is op de afbeelding het symbool even weggehaald, zodat je alleen de achtergrond ziet.

`background: #bbb;`
Hieronder wordt een gradiënt (verlopende kleur) als achtergrond opgegeven. Internet Explorer 9 (en andere oudere browser) kennen dit niet. Voor die browsers wordt een grijze achtergrond opgeven, dezelfde achtergrondkleur als de andere knoppen hebben.

`background: -webkit-radial-gradient(#fff, #4f3cfc);`
`background: radial-gradient(#fff, #4f3cfc);`
Hier staat in feite twee keer hetzelfde: `background: radial-gradient(#fff, #4f3cfc);`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Een gradiënt is een verlopende kleur. In dit geval is het een heel simpele gradiënt: een cirkel die in het midden wit is en naar de rand toe geleidelijk naar paars verkleurt. Omdat het ’n simpele gradiënt is, worden er maar ’n paar van de mogelijke waarden gebruikt.
`radial-gradient` geeft aan dat het om een ronde gradiënt gaat. Tussen de haakjes staan de beginkleur `#fff` (wit) en de eindkleur `#4f3cfc` (paars), met een komma ertussen. Gradiënts kunnen enorm ingewikkeld worden, maar voor zo’n simpele heb je niet meer nodig dan dit.

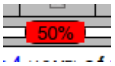
```
#play-2:focus {background: white;}
```

Het element met id=”play-2”. Dat is de Speel-pauzeerknop van de tweede video. Als deze focus heeft, maak dan de achtergrond wit.
Dit is eerder al opgegeven met de selector .controls button:focus, ..., maar de hierboven staande selector `#play-2` heeft meer specificiteit, meer gewicht, en wint van die eerdere selector. Daarom wordt hier ’n selector gebruikt, die evenveel specificiteit heeft als de hierboven staande. (Feitelijk zelfs meer, omdat `:focus` ook gewicht in de schaal legt.)

```
#image-slider-button-2 {width: 40px;}
```

Voor dit element geldt ook de eerder bij .image-slider-button opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”image-slider-button-2”. Dat is de sleepknop van de sleepbalk voor afspelen die bij de tweede video hoort. 40 px breed maken. Het script verrekent deze breedte automatisch, waardoor de knop altijd goed wordt neergezet.

```
#image-slider-button-2::after {display: block; font-size: 13px;
    line-height: 14px; text-align: center;}
```



Met behulp van `::after` wordt bij het element met id=”image-slider-button-2” (de sleepknop voor afspelen bij de tweede video) een pseudo-element gemaakt, waarin de verstreken speelduur in procenten wordt weergegeven. Bovenstaande css geeft een pseudo-element, dat precies boven de sleepknop voor het afspelen staat. Door de achtergrond even rood te maken, wordt het hele pseudo-element op de afbeelding zichtbaar. Dit pseudo-element is nog helemaal leeg, de weergave van het percentage wordt gelijk hieronder geregeld met behulp van `content`.

```
#videobox-2[data-played="percent-00"] .image-slider-button::after
    {content: "0%";}
#videobox-2[data-played="percent-01"] .image-slider-button::after
    {content: "1%";}
#videobox-2[data-played="percent-02"] .image-slider-button::after
    {content: "2%";}
```

(…) nog 95 dezelfde regels met een steeds hoger getal (…)

```
#videobox-2[data-played="percent-98"] .image-slider-button::after
    {content: "98%";}
#videobox-2[data-played="percent-99"] .image-slider-button::after
    {content: "99%";}
#videobox-2[data-played="percent-100"] .image-slider-button::after
    {content: "100%";}
```

Dit kan alleen maar door een gediplomeerde gek zijn bedacht. En dat is ook zo, want ik heb het zelf bedacht. Honderdeneen regels code om ’n percentage te kunnen laten zien. Het is natuurlijk ook absoluut niet de bedoeling dat zoiets ooit in het echt gebruikt zou worden. Maar het is ’n aardige illustratie van wat er mogelijk is, als je met css via JavaScript toegang hebt tot iets als een <video>-element.
En wat levert deze indrukwekkende berg aan code op? 2 (Twee!) cijfers en 1 (Één!) procentteken. Die wel meebewegen met de sleepknop, dat nog wel. En aan het eind leveren deze 101 regels code zelfs wel 3 (Drie!) cijfers op! En ’n procentteken! Dat is meer dan 25 regels code om één teken tevoorschijn te toveren. En dan zeggen ze dat Opstelten langdradig is...

`#videobox-2`: het element met id=”videobox-2”. Dat is de waar de tweede videospeler in staat.

`[data-played="percent-00"]`: Er moet een attribuut ’data-played’ aanwezig zijn dat de waarde ’percent-00’ heeft. (En in latere regels ’percent-01’, ’percent-02’, enz., t/m ’percent-100’.) Dit attribuut wordt door het script ingevoegd. De waarde is het percentage van de verstreken speelduur van de video. Dit wordt door het script bijgehouden.

`.image-slider-button::after`: het element met class=’image-slider-button’ dat binnen `div#videobox-2` ligt. Dat is er maar eentje: de met de sleepknop voor het afspelen.

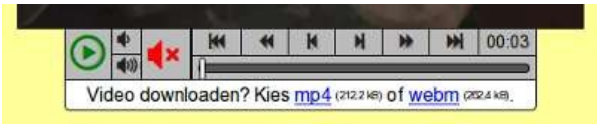
De css voor het pseudo-element dat met `::after` wordt gemaakt is al hierboven opgegeven, alleen `content` miste nog.

Als het attribuut `data-played` bij `#videobox-2` een waarde heeft van ’percent-00’ (de video staat helemaal aan het begin), geef dan het met behulp van `::after` bij `.image-slider-button` gemaakte pseudo-element een `content` van ’0%’. En die ’0%’ verschijnt dan dus op de speelknop.

Als 10% van de video is afgespeeld, is de waarde van `data-played` ’percent-10’ en wordt de inhoud van `content` ’10%’. Enz.

Omdat de knop van de sleepbalk mee beweegt met het verstrijken van de speelduur, beweegt het met behulp van `::after` gemaakte pseudo-element ook mee, en staat het verstreken percentage speelduur dus altijd op de knop, waar deze ook staat.

Speciaal voor derde video (pagina 2)



Weinig spectaculair: alleen de bedieningselementen en de download-links onder de video zijn smaller dan de video zelf. En de sleepknop is iets te veel afgevallen.

```
#controls-3 {width: 400px;}
```

Het element met id=”controls-3”. De waar de bedieningselementen voor de derde videospeler in zitten. De video is 480 px breed. Door deze 400 px breed te maken, worden de bedieningselementen aan weerszijden 40 px smaller dan de video zelf.

```
#controls-3 + .groot {width: 394px;}
```

Het element met class=”groot” dat in de html gelijk volgt op het element met id=”controls-3”. Dit is de <p> waarbinnen de download-links zitten. Ook dit wordt smaller gemaakt. Omdat deze <p> links en rechts een padding van 3 px heeft, wordt hij 6 px smaller gemaakt dan de `div#controls-3`, waarin de bedieningselementen zitten, zodat ze uiteindelijk even breed zijn.

```
#image-3 {width: 292px;}
.videobox[data-sound="no"] #image-3 {width: 359px;}
#image-3 button {width: 40px;}
#videobox-3[data-sound="no"] .image button {width: 50px;}
```

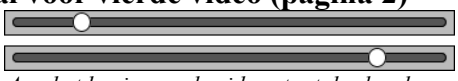
```
#videobox-3[data-sound="no"] .to-begin::after, #videobox-3[data-
     sound="no"] .five-back::after, #videobox-3[data-
     sound="no"] .ten-back::after, #videobox-3[data-
     sound="no"] .ten-forward::after, #videobox-3[data-sound="no"]
     .five-forward::after, #videobox-3[data-sound="no"] .to-
     end::after {width: 26px;}
#videobox-3[data-sound="no"] .remaining {width: 59px;}
button#to-begin-3, button#to-end-3 {width: 42px;}
#to-begin-3::after {background-position: -210px;}
#five-back-3::after {background-position: -315px;}
#ten-back-3::after {background-position: -425px;}
#ten-forward-3::after {background-position: -475px;}
#five-forward-3::after {background-position: -370px;}
#to-end-3::after {background-position: -258px;}
#image-3 #image-slider-beam-3 {width: 280px;}
#videobox-3[data-sound="no"] .image #image-slider-beam-3 {width:
     330px;}
```

Voor veel van bovenstaande elementen is eerder al css opgegeven bij Knoppen afspelen, die ook hier geldt, voor zover die hier niet wordt veranderd.
Een hele reeks kleine aanpassingen, die allemaal met de smallere knoppen e.d. te maken hebben. De selectors met `data-sound="no"` zijn bedoeld voor iOS, waar geen geluidsregeling aanwezig is en maten e.d. daaraan moeten worden aangepast. Meer uitleg bij .videobox[data-sound=”no”] .play.

```
#image-slider-button-3 {width: 3px; height: 17px; top: -6px;}
```

Voor dit element geldt ook de eerder bij Sleepbalk afspelen opgegeven css, voor zover die hier niet wordt veranderd.
Hier wordt de sleepbalk iets smaller en hoger gemaakt dan eerder opgegeven. Vanwege de afwijkende hoogte moet ook `top` worden aangepast.
De afwijkende breedte wordt automatisch verrekend door het script, waardoor de knop toch altijd op de goede plaats staat.

Speciaal voor vierde video (pagina 2)



Aan het begin van de video staat de sleepknop zoals op de bovenste sleepbalk, aan het eind zoals op de onderste. Er blijft ruimte tussen knop en begin of eind van de balk.

Bij deze videospeler gaat de knop van de sleepbalk voor afspelen niet helemaal tot links en rechts: er blijft altijd een ruimte op de balk van de sleepbalk over. Daarnaast heeft de sleepknop weer een afwijkende maat.

```
#remaining-4 {position: absolute; top: 0; right: 0;}
```

Voor dit element geldt ook de eerder bij Resterende speelduur opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”remaining-4”. Dit is de waarbinnen in de vierde video de resterende speelduur staat.
De met de balk van de sleepbalk heeft bij Sleepbalk afspelen een absolute positie gekregen, waardoor hij voor andere elementen als het ware niet meer aanwezig is. Gelijk hieronder krijgt de sleepbalk van deze videospeler een relatieve positie, waardoor hij weer wel aanwezig is. En als dank daarvoor `#remaining-4` van z’n plaats mept. Door `#remaining-4` absoluut te positioneren, staat deze weer op de juiste plaats.

`#image-slider-beam-4`

Voor dit element geldt ook de eerder bij Sleepbalk afspelen opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”image-slider-beam-4”. De met de balk van de sleepbalk in de vierde video.
Hoewel het er mogelijk op lijkt dat er iets met de sleepknop is gedaan, om de lege ruimte links en rechts te krijgen, is dat niet het geval. De met de balk van de sleepbalk is ingekort, waarna er met behulp van `::before` en `::after` pseudo-elementen zijn gemaakt, die links en rechts van de balk zijn neergezet. Door die pseudo-elementen goed op de echte sleepbalk aan te laten sluiten, lijkt het één lange sleepbalk.

`width: 260px;`

Om ruimte te maken voor de pseudo-elementen links en rechts van de sleepbalk, moet de sleepbalk worden ingekort. De eerder opgeven breedte van 360 px wordt met 100 verminderd, waardoor er links en rechts ruimte komt voor de ’nep’-sleepbalken.

`margin: 0 auto;`

Hieronder wordt de eerder opgegeven absolute positie veranderd in een relatieve, waardoor de positionering niet meer werkt. Daar zou je trouwens toch niets aan hebben, omdat de breedte is aangepast.
Boven en onder geen marge, links en rechts `auto`, wat in dit geval betekent: evenveel. De sleepbalk staat dus netjes in het midden van z’n ouder `div#image-slider-4`.

`border-radius: 0;`

Om de pseudo-elementen links en rechts van de sleepbalk goed aan te laten sluiten, moet de ronde hoek bij de sleepbalk worden weggehaald.

`position: relative;`

Bij Sleepbalk afspelen is de sleepbalk met behulp van absoluut positioneren op de juiste plaats gezet. Hier is het makkelijker dat met behulp van het hierboven gebruikte `margin 0 auto;` te doen. Maar dat werkt niet als het element een absolute positie heeft. Daarom wordt dat hier in een relatieve positie veranderd.

`top: 5px; left: 0;`

Nog twee kleine correcties ten opzichte van de eerder opgegeven css.

`#image-slider-beam-4::before`

Met behulp van `::before` wordt bij het element met id=”image-slider-beam-4” een pseudo-element gemaakt. Op de afbeelding is dat het rode vlak links. Het is hier even rood gemaakt, maar in de videospeler heeft het dezelfde kleur als de balk van de sleepbalk, waardoor het lijkt alsof het één geheel is. Lijkt, want de sleepknop kan nooit op het rode deel van de sleepbalk komen.
(Voor het gele vlak rechts geldt precies hetzelfde, maar dan in spiegelbeeld.)

`content: "";`

Het pseudo-element heeft geen inhoud. Maar `content` moet wel aanwezig zijn, dus krijgt het een inhoud van niets.

`background: #555;`

Zelfde achtergrondkleur als de eigenlijke sleepbalk.

`width: 50px;`

50 px links van de eigenlijke sleepbalk laten uitsteken.

`height: 6px;`

Even hoog als de eigenlijke sleepbalk maken.

`border: black solid 1px;`
Zelfde border als de eigenlijke sleepbalk geven.
`border-right: none;`
Streepje tussen pseudo-element en sleepbalk voorkomen.
`border-radius: 5px 0 0 5px;`
Linksboven en linksonder ronde hoeken geven.
`position: absolute;`
Om het pseudo-element goed neer te kunnen zetten.
Van zichzelf is het pseudo-element een inline-element. Daardoor kunnen eigenschappen als hoogte en breedte niet worden gebruikt. Door het element absoluut te positioneren, verandert het in een soort blok-element, waardoor eigenschappen als hoogte en breedte gebruikt kunnen worden.
`top: -1px; left: -50px;`
Goed laten aansluiten op de eigenlijke sleepbalk.

```
#image-slider-beam-4::after {content: ""; background: #555; width:
    50px; height: 6px; border: black solid 1px; border-left: none;
    border-radius: 0 5px 5px 0; position: absolute; top: -1px;
    right: -50px;}
```

Met behulp van `::after` wordt bij het element met id=”image-slider-beam-4” een pseudo-element gemaakt. De beschrijving is exact hetzelfde als bij `#image-slider-beam-4::before` gelijk hierboven, alleen zijn een aantal eigenschappen in spiegelbeeld, omdat het hier om de verlenging rechts van de sleepbalk gaat.

```
#image-slider-beam-4:focus::before, #image-slider-beam-
    4:focus::after {background: white;}
```

Als de sleepbalk focus heeft, wordt de achtergrond wit. Dat moet dus ook gebeuren bij de twee hierboven met behulp van `::before` en `::after` gemaakte pseudo-elementen.

```
#image-slider-button-4 {width: 13px; z-index: 100;}
```

De eerder bij Sleepbalk afspelen opgegeven css geldt ook voor dit element, voor zover die hier niet wordt veranderd.
De eerder opgegeven breedte wordt iets kleiner. De afwijkende breedte wordt automatisch verrekend door het script, waardoor de knop toch altijd op de goede plaats staat.
De z-index is nodig voor Safari op OS X. Zonder z-index verdwijnt de sleepknop onder het pseudo-element aan de linkerkant, als daarop wordt geklikt.

Speciaal voor vijfde video (pagina 2)


Deze video heeft alleen een brede sleepknop. Wat niet echt handig is, omdat de sleepknop nooit buiten de sleepbalk komt, en je dus weinig ruimte over hebt om de sleepknop te bewegen. Maar technisch kan het.

```
#image-slider-button-5 {width: 120px;}
```

Voor dit element geldt ook de eerder bij .image-slider-button opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”image-slider-button-5”: de sleepknop van de vijfde video. 120 px breed maken. De afwijkende breedte wordt automatisch verrekend door het script, waardoor de knop toch altijd op de goede plaats staat.

Speciaal voor zesde video (pagina 2)

De sleepbalk is extra dik en heeft een gradiënt als achtergrond. Daardoor moeten ook allerlei andere maten worden aangepast.

```
#controls-6 {height: 64px;}
#play-6 {height: 64px;}
#play-6::after {background: url(../103-images/iconenfont-
    fallback.png) -1px 15px no-repeat; height: 64px; line-height:
    60px;}
#play-6.not-playing::after {background: url(../103-
    images/iconenfont-fallback.png) -43px 15px no-repeat;}
#sound-6 {height: 64px;}
#sound-6 button {height: 32px;}
#louder-6 {border-bottom: none;}
#softer-6::after, #louder-6::after {line-height: 30px;}
#sound-6 #mute-6 {height: 64px; margin-top: -33px; border-bottom:
    none;}
#image-6 {height: 64px;}
```

Om ruimte te maken voor de dikkere sleepbalk, moeten allerlei maten, posities, e.d. worden aangepast. Dat kan door de aan te passen bedieningselementen aan te roepen met hun id, waardoor deze css alleen geldt voor bedieningselementen binnen de zesde videospeler. Voor al deze elementen geldt de eerder met behulp van een class opgegeven css ook, voor zover die hier niet wordt veranderd.

`#image-slider-beam-6`

Voor dit element geldt ook de eerder bij .image .image-slider-beam opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”image-slider-beam-6”: de sleepbalk (de balk zelf) van de zesde videospeler.

`background: yellow;`

Hieronder wordt een gradiënt opgegeven, een verlopende achtergrondkleur. Oudere browsers kennen dat niet, daarom wordt hier een gewone achtergrondkleur opgegeven. Browsers die het hieronder gebruikte `linear-gradient` niet kennen, geven dan gewoon een egaal gele achtergrondkleur aan de sleepbalk.
Dit geldt in ieder geval voor Internet Explorer 9 en Android browser 4.0.3. (Android browser 4.0.3 kent wel een oudere vorm van `linear-gradient`, maar daar vind ik het niet belangrijk genoeg voor. De nieuwere vorm van `radial-gradient` wordt trouwens vreemd genoeg wel herkend door deze browser.)

```
background: -webkit-linear-gradient(to right, blue, yellow);
    background: linear-gradient(to right, blue, yellow);
```

Hier staat in feite twee keer hetzelfde: `background: linear-gradient(to right, blue, yellow);`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Een gradiënt is een verlopende kleur. In dit geval is het een heel simpele gradiënt: een lineaire gradiënt die van blauw aan de linkerkant naar geel rechts verkleurt. Omdat het ’n simpele gradiënt is, worden er maar ’n paar van de mogelijke waarden gebruikt.

`linear-gradient` geeft aan dat het om een lineaire gradiënt gaat. Standaard verloopt de kleur van boven naar beneden.
`to right` zorgt ervoor dat de kleur hier van links naar rechts verloopt. De beginkleur is `blue` (blauw), de eindkleur `yellow` (geel), gescheiden door een komma.

`height: 30px; border-radius: 15px; top: 28px;`
Grotere hoogte dan eerder is opgegeven. Vanwege die grotere hoogte moeten ook de ronde hoeken en de positie worden aangepast.

```
#image-slider-button-6 {background: darkorange; width: 24px;
    height: 24px; border: mediumspringgreen solid 3px; border-
    radius: 14px; top: 0;}
```


Voor dit element geldt ook de eerder bij .image-slider-button opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”image-slider-button-6”, de knop van de sleepbalk voor weergave van de zesde videospeler.
Gewoon wat aanpassingen om de sleepknop eruit te laten zien als een overjarige opengesprongen steenpuist. De enige reden om de kleur ’mediumspringgreen’ te gebruiken, is de fantastische naam. Dit is dus ’n naam, zoals je die officieel in css3 mag gebruiken. ’Springgreen’ bestaat trouwens ook, ’darkspringgreen’ en ’lightspringgreen’ moeten kennelijk wachten tot css4.

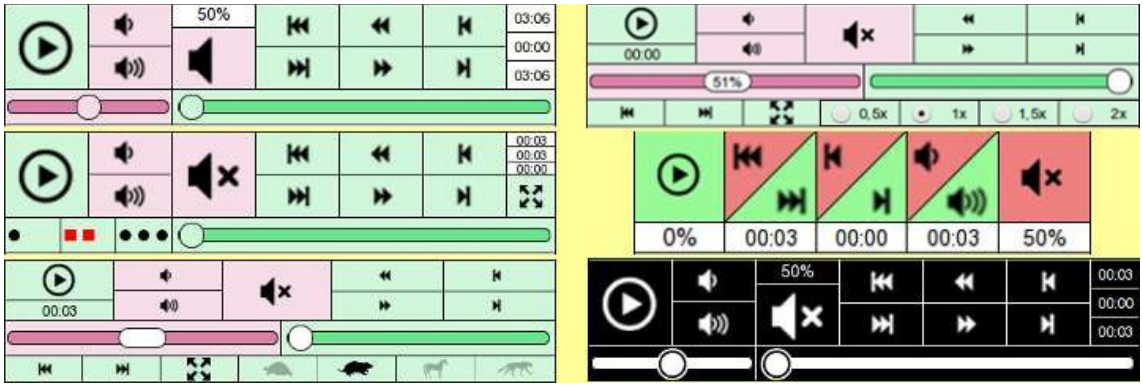
```
#image-slider-beam-6:focus {background: white;}
#image-slider-beam-6:focus .image-slider-button {background: red;}
```

De met de balk en de met de knop van de sleepbalk voor afspelen van de zesde videospeler.
Bij Focus is geregeld dat de balk van de sleepbalk wit en de knop van de sleepbalk rood wordt, als de sleepbalk focus heeft. Voor de sleepbalk wordt daarvoor de selector `.controls .image-slider-beam:focus` gebruikt.
Hier iets boven krijgt de balk echter een achtergrondkleur met behulp van de selector `#image-slider-beam-6`. Omdat deze selector een id heeft, heeft hij meer specificiteit (’gewicht’) dan de eerder gebruikte selector. Daardoor ’wint’ deze selector van de eerdere en wordt ook bij focus de kleur niet veranderd.
Om dat te verhelpen, wordt hier voor deze balk nogmaals een achtergrondkleur opgegeven, waarbij een id in de selector zit: `#image-slider-beam-6:focus`. Nu heeft deze selector genoeg selectiviteit om de eerdere selector te overrulen en verandert ook bij deze balk de achtergrondkleur, als de sleepbalk focus heeft.
Voor de tweede regel geldt precies hetzelfde, maar nu voor de knop van de sleepbalk.

Pagina 3

De css die hier wordt beschreven, hoort bij de stylesheet ’afbeelding-103-3-dl.css’. Deze stylesheet hoort bij de derde pagina met videospelers. Omdat het bij vijf pagina’s met grotendeels verschillende videospelers om heel veel css gaat, wordt hier alleen de css besproken, die afwijkt van die in ’afbeelding-103-1-dl.css’. De css daarin is te vinden bij Pagina 1.

Omdat het uiterlijk van de zes videospelers op deze pagina nogal verschilt, worden van alle zes hieronder de bedieningselementen neergezet.



Algemeen (pagina 3)

Omdat de volgorde van de elementen in de html anders is dan die op het scherm, heeft deze pagina een aangepaste Tabindex.

Verborgen bedieningselementen (pagina 3)

```
.controls .fullscreen, .speed {display: none;}
```

Deze regel zorgt ervoor dat een deel van de bedieningselementen volledig wordt verborgen, en ook door schermlezers wordt genegeerd. Sommige selectors zijn langer dan andere, omdat ze voldoende specificiteit (’gewicht’) moeten hebben om andere selectors te overrulen.

Achter de schermen blijft alles gewoon werken, want het script wordt niet veranderd. Als je de gegenereerde code bekijkt, zie je dat de elementen nog steeds gewoon aanwezig zijn, maar niet langer op het scherm zichtbaar zijn.

Omdat in de selectors classes worden gebruikt, worden deze elementen op de hele pagina verborgen. Je kunt ze ook in één of enkele videospelers verbergen door gebruik te maken van id’s. Door gebruik te maken van een id kun je eventueel ook in één of enkele spelers verborgen elementen toch weer zichtbaar maken.

De volgende elementen worden verborgen:

`.controls .fullscreen`: knop Fullscreen.

`.speed`: volledige snelheidsregeling.

Symbolen voor bedieningselementen (pagina 3)

Op deze pagina zijn achtergrond-afbeeldingen gebruikt voor de bedieningselementen. Er worden op deze pagina 63 verschillende achtergrond-afbeeldingen gebruikt. Het zijn er zoveel, omdat er verschillende soorten videospelers op de pagina staan. Normaal genomen zullen videospelers hetzelfde zijn en zul je dus ongeveer drie keer zo weinig achtergrond-afbeeldingen gebruiken.

Om het aantal aanroepen naar de server te beperken, zijn verschillende achtergrond-afbeeldingen samengevoegd tot één grotere (een ’sprite’), waarbij de juiste achtergrond-afbeelding dan met behulp van `background-position` op de juiste plaats wordt gezet.

Normaal genomen zou je alle 63 afbeeldingen op één grote afbeelding zetten, maar hier heb ik ze in drie groepen verdeeld. Dat was om te voorkomen dat ik helemaal gek zou worden

van de verschillende spelers met verschillende soorten afbeeldingen. Dus in plaats van één aanroep naar de server, gebruikt deze pagina er drie voor de achtergrond-afbeeldingen. Altijd nog ’n stuk beter dan 63.
De gebruikte achtergrond-afbeeldingen zijn te herkennen aan ’background-page-3’ in de naam. Op de afbeelding hieronder zijn ze boven elkaar gezet. Van boven naar beneden: ’buttons-background-page-3.png’, ’buttons-background-page-3-invert.png’ en ’buttons-background-page-3-klein.png’.

De achtergrond is even groen gemaakt, omdat de middelste (witte) sprite anders niet is te zien. In werkelijkheid is de achtergrond doorzichtig.

De achtergrond-afbeelding wordt gewoon op de normale manier gebruikt. Door de afbeelding met behulp van `background-position` het juiste aantal pixels te verschuiven, komt het juiste deel van de achtergrond-afbeelding in de knop te staan. `background-position` wordt niet afzonderlijk gebruikt, maar is opgenomen in de ’shorthand’ `background`, omdat dat veel korter is dan het afzonderlijk gebruiken van `background-image`, `background-repeat`, `background-color` en `background-position`.
Bij bijvoorbeeld de Speel-pauzeerknop van de eerste video wordt het dan:

```
background: url(../103-images/buttons-background-page-
     3.png) -4px no-repeat #d1f7db;
```

Achtergrond-afbeelding, vier px naar links verplaatsen, niet herhalen en als laatste een achtergrondkleur, die hier lichtgroen is.

css voor vensters breder dan 700 px (pagina 3)

```
@media screen and (min-width: 700px) {
     main nav + p {
          -moz-column-count: 2;
          -moz-column-gap: 1em;
          -webkit-column-count: 2;
          -webkit-column-gap: 1em;
          column-count: 2;
          column-gap: 1em;
     }
}
```

In bredere browservensters te lange regels voorkomen door de tekst in kolommen op te delen. Dit is precies hetzelfde als bij css voor vensters breder dan 700 px voor de tweede pagina, waar de beschrijving is te vinden.

css voor vensters breder dan 1200 px (pagina 3)

```
@media screen and (min-width: 1200px) {
	main nav + p {
		-moz-column-count: 3;
		-webkit-column-count: 3;
		column-count: 3;
	}
}
```

In bredere browservensters te lange regels voorkomen door de tekst in kolommen op te delen. Dit is precies hetzelfde als bij css voor vensters breder dan 1200 px voor de tweede pagina, waar de beschrijving is te vinden.

Bedieningselementen algemeen (pagina 3)

```
.controls {width: 480px; height: 105px; margin: 0 auto; border:
	black solid; border-width: 0 1px 1px;}
```

De elementen met class=”controls”. Dit zijn de ’s, waarbinnen de hele bediening staat. Het enige verschil met de eerste pagina is de grotere hoogte en het ontbreken van een achtergrondkleur. Omdat de bedieningselementen op deze pagina nogal verschillende achtergrondkleuren krijgen, wordt dat hier later voor afzonderlijke (groepen) bedienings-elementen geregeld.

Speel-pauzeerknop (pagina 3)

```
.play
```

De elementen met class=”play”: de Speel-pauzeerknoppen. Deze css geldt voor álle Speel-pauzeerknoppen op de pagina. Sommige knoppen zien er anders uit, die krijgen later eigen aanvullende css. De eerste en de derde Speel-pauzeerknop gebruiken dezelfde achtergrond-afbeelding, en de eerste, derde en zesde knop hebben alles hetzelfde, behalve de achtergrond. Door dat allemaal hier al op te geven, spaar je behoorlijk wat css verderop uit.

```
	background: url(../103-images/buttons-background-page-3.png)
		-4px no-repeat #d1f7db;
```

Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
	width: 73px; height: 75px;
```

Hoogte en breedte.

```
	float: left;
```

Makkelijkste manier om meer knoppen e.d. naast elkaar te zetten.

```
	border: black solid; border-width: 0 1px 1px 0;
```

Rechts en onder randje.

```
.not-playing {background: url(../103-images/buttons-background-
	page-3.png) -85px no-repeat #d1f7db;}
```

Behalve dat het hier om een driehoekje gaat, is het verhaal precies hetzelfde als gelijk hierboven bij `.play`. En uiteraard is de achtergrondpositie anders.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Hierdoor kan, als de video niet speelt, een ander symbool worden getoond, dan wanneer de video wel speelt.

Knoppen en percentage volume (pagina 3)

```
.sound {width: 146px; height: 105px; float: left; position:
    relative;}
```

De elementen met class=”sound”. De ’s waar de bedieningselementen voor het geluid in zitten. Behalve ’n andere breedte en hoogte weinig anders dan op de eerste pagina.
Voor zover dit bij één of meer videospelers moet worden aangepast, krijgen die later eigen aanvullende css.

```
.sound button, .sound span {width: 73px; height: 37px; text-align:
    center; border: black solid; border-width: 0 1px 1px 0;
    position: absolute;}
```

De <button>’s binnen de elementen met class=”sound” en de ’s binnen de elementen met class=”sound”. Er is maar één element met class=”sound”: de waarbinnen de bedieningselementen voor het geluid zitten.
Anders dan op de eerste pagina worden de bedieningselementen niet gefloat, en krijgen ze geen relatieve maar een absolute positie. Ze worden met behulp van die absolute positie op de juiste plaats gezet.
Veel css voor de knoppen voor Harder, Zachter en de Aan-uitknop voor het geluid, en voor de met het percentage van de geluidssterkte, is hetzelfde. Die kan hier in één keer worden opgegeven. Eventuele aanpassingen voor individuele elementen komen dan later.
In de eerste, derde en zesde videospeler is het uiterlijk van de bediening voor het geluid grotendeels hetzelfde. Kleinere aanpassingen voor deze videospelers, en grotere voor de andere drie, volgen later.

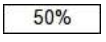
```
.sound .softer {background: url(../103-images/buttons-background-
    page-3.png) -412px no-repeat #f7dde8;}
```


Voor dit element geldt ook de eerder bij .sound button, .sound span opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”softer” die binnen een element met class=”sound” zitter. Dit zijn de knoppen voor Zachter.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
.aria-volume {left: -20000px;}
```

Voor dit element geldt ook de eerder bij .sound button, .sound span opgegeven css, voor zover die hier niet wordt veranderd.
De die wordt gebruikt door schermlezers om de geluidssterkte na wijziging voor te lezen heeft een class=”aria-volume”. Zelfde als op de eerste pagina, alleen mist `position: absolute;` omdat die al bij .sound button, .sound span is opgegeven.

```
.sound .percentage {background: white; width: 72px; height: 20px;
    font-size: 14px; line-height: 18px; top: 0; left: 73px;}
```



Voor dit element geldt ook de eerder bij .sound button, .sound span opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”percentage” die binnen de elementen met class=”sound” liggen. Dit zijn de ’s waarbinnen de geluidssterkte in procenten wordt weergegeven.
Gewoon wat css om ze op te maken.

```
.sound .louder {background: url(../103-images/buttons-background-
    page-3.png) -325px no-repeat #f7dde8; height: 38px; top:
    37px;}
```

Voor dit element geldt ook de eerder bij .sound button, .sound span opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”louder” die binnen een element met class=”sound” zitten. Dit zijn de knoppen voor Harder.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
.sound .mute {background: url(../103-images/buttons-background-
    page-3.png) -165px no-repeat #f7dde8; height: 54px; top: 21px;
    left: 73px;}
```

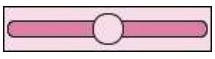
Voor dit element geldt ook de eerder bij .sound button, .sound span opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”mute” die binnen een element met class=”sound” zitten. Dit zijn de Aan-uitknoppen voor geluid.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
.sound .mute.muted {background: url(../103-images/buttons-
    background-page-3.png) -256px no-repeat #f7dde8;}
```

Behalve dat hier het kruisje bij de luidspreker mist, is het verhaal precies hetzelfde als gelijk hierboven bij `.mute`. En uiteraard is de achtergrondpositie anders.
Aan de Geluid aan-uitknop wordt door het script een class=”muted” toegevoegd, als het geluid uitstaat. Hierdoor kan, als het geluid uitstaat, een ander symbool worden getoond, dan wanneer het geluid aanstaat.

Sleepbalk volume (pagina 3)

```
.sound .sound-slider {background: #f7dde8; height: 30px; width:
    145px; border-right: black solid 1px; position: absolute;
    bottom: 0; left: -73px;}
```

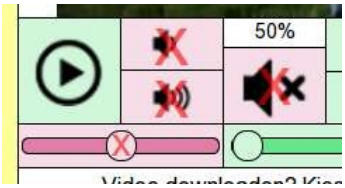


De elementen met class=”sound-slider” die binnen een element met class=”sound” liggen. Dit zijn de ’s, waarbinnen de sleepbalken voor het geluid zitten. Op de afbeelding is het de lichtroze rechthoek. Deze is groter dan de eigenlijke sleepbalk, die op de afbeelding donkerroze is.
Voor de meeste videospelers wordt de meeste css later weer veranderd, maar het is toch de moeite waard hier al van alles neer te zetten. Want omdat veel css later niet wordt veranderd, spaart dit uiteindelijk toch css uit.
Hier worden gewoon wat maten en kleuren opgegeven, en het element wordt op de goede plaats gezet.

```
.sound .sound-slider-beam {background: #df7ea7; width: 139px;
    height: 10px; border: black solid 1px; border-radius: 5px;
    position: absolute; bottom: 9px; left: 2px;}
.sound-slider-button {background: #f7dde8; width: 20px; height:
    20px; border: black solid 1px; border-radius: 10px; top:
    -6px;}
```

De eerste regel is voor de elementen met class=”sound-slider-beam” binnen een element met class=”sound”, de tweede regel is voor de elementen met class=”sound-slider-button” binnen een element met class=”sound”.
De eerste regel is voor de balk van de sleepbalk voor geluid, de tweede regel is voor de sleepknop van de sleepbalk.
Opmaak voor de sleepbalk, waarvan het suikerzoete resultaat op de afbeelding hier vlak boven is te zien.
Het script plaatst de knop altijd op de goede plaats, waarbij wordt gecorrigeerd voor breedte van knop en balk, borders, e.d. Meer daarover bij Sleepbalk volume.

```
.videobox[data-sound="no"] .softer::after, .videobox[data-
    sound="no"] .louder::after, .videobox[data-
    sound="no"] .mute::after, .videobox[data-sound="no"] .sound-
    slider-button::after, #videobox-2[data-sound="no"] #sound-
    slider-button-2::after {content: "x"; color: rgba(255, 0, 0,
    0.6); display: block; height: 37px; font-size: 50px; line-
    height: 28px; text-align: center; margin-top: -1px; speak:
    none;}
```



Op iOS kan de geluidssterkte alleen worden veranderd met de knoppen op het apparaat, niet met de knoppen in de videospeler. Daarom wordt op iOS een rood kruis door de bedienings-elementen voor het geluid gezet.
Als het geluid niet kan worden aangepast binnen de videospeler, krijgt het attribuut `data-sound` van `div.videobox` de waarde ’no’. Meer daarover bij .videobox[data-sound=”no”] .play. Door dit attribuut in de selector op te nemen, kan het uiterlijk worden aangepast.
Met behulp van `::after` wordt een pseudo-element aangemaakt, met behulp waarvan boven de knoppen een rode, grote, enigszins doorzichtige, ’x’ wordt neergezet. Met wat extra css wordt de ’x’ op de goede plaats gezet.
Omdat hier ook op iOS de bedieningselementen voor de geluidssterkte gewoon aanwezig zijn, zijn geen aanpassingen voor de maten van de andere bedieningselementen nodig. (Bij de meeste videospelers is dat wel nodig, omdat daar allerlei bedieningselementen voor de geluidssterkte volledig ontbreken.)

```
.videobox[data-sound="no"] .sound-slider-button::after, #videobox-
    2[data-sound="no"] #sound-slider-button-2::after {font-size:
    28px; line-height: 16px;}
```

Voor deze elementen geldt ook de gelijk hierboven opgegeven css, voor zover die hier niet wordt veranderd.
Het rode kruis op de sleepbalk, dat gelijk hierboven is opgegeven, wordt hier iets kleiner gemaakt, zodat het binnen de sleepknop past.
De tweede selector is specifiek op de tweede videospeler gericht. Deze heeft meer specificiteit (’gewicht’) nodig, omdat bij Speciaal voor tweede video een andere lettergrootte

en regelhoogte worden opgegeven met de selector `#sound-slider-button-2::after`. Deze zou 'winnen' van de eerste selector hierboven, omdat er een id in wordt gebruikt. De tweede selector heeft echter genoeg specifiteit, om te voorkomen dat het kruisje de verkeerde maat krijgt in de tweede videospeler.

Knoppen afspelen (pagina 3)

```
.image {width: 219px; height: 105px; float: left; position: relative;}
```

De elementen met class="image". De 's waar de bedieningselementen voor het afspelen in zitten. De enige verschillen met de eerste pagina zijn andere maten en een missende border.

```
.image button {width: 73px; height: 38px; border: black solid; border-width: 0 1px 0 0; position: absolute;}
```

De <button>'s binnen de elementen met class="image". De knoppen die het afspelen van de video regelen. Er zijn alleen wat maten, borders, e.d. anders dan op de eerste pagina.
Deze instellingen gelden voor de hele pagina. Voor een aantal videospelers worden ze later aangepast.

```
.image .to-begin {background: url(../103-images/buttons-background-page-3.png) -484px no-repeat #d1f7db;}
```

Voor dit element geldt ook de eerder bij .image button opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class="to-begin" binnen de elementen met class="image". De knoppen Terug naar begin.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
.image .five-back {background: url(../103-images/buttons-background-page-3.png) -644px no-repeat #d1f7db; left: 73px;}
```

Voor dit element geldt ook de eerder bij .image button opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class="five-back" binnen de elementen met class="image". De knoppen Vijf procent terug.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
.image .ten-back {background: url(../103-images/buttons-background-page-3.png) -804px no-repeat #d1f7db; left: 146px;}
```

Voor dit element geldt ook de eerder bij .image button opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class="ten-back" binnen de elementen met class="image". De knoppen Tien seconden terug.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
.image .ten-forward {background: url(../103-images/buttons-
    background-page-3.png) -884px no-repeat #d1f7db; border-
    width: 1px 1px 1px 0; left: 146px; top: 37px;}
```

Voor dit element geldt ook de eerder bij .image button opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”ten-forward” binnen de elementen met class=”image”. De knoppen Tien seconden verder.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

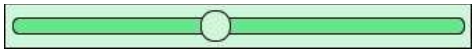
```
.image .five-forward {background: url(../103-images/buttons-
    background-page-3.png) -724px no-repeat #d1f7db; border-
    width: 1px 1px 1px 0; left: 73px; top: 37px;}
```

Voor dit element geldt ook de eerder bij .image button opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”five-forward” binnen de elementen met class=”image”. De knoppen Vijf procent verder.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
.image .to-end {background: url(../103-images/buttons-background-
    page-3.png) -564px no-repeat #d1f7db; border-width: 1px 1px
    1px 0; top: 37px;}
```

Voor dit element geldt ook de eerder bij .image button opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”to-end” binnen de elementen met class=”image”. De knoppen Naar het einde.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

Sleepbalk afspelen (pagina 3)



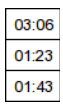
De sleepbalk voor het afspelen is vrijwel hetzelfde als de Sleepbalk afspelen van de eerste pagina. Het grootste verschil zit in de afmetingen.

```
.image-slider {background: #d1f7db; height: 30px; width: 334px;
    position: absolute; bottom: 0; left: -73px;}
.image .image-slider-beam {background: #66e78c; width: 323px;
    height: 10px; border: black solid 1px; border-radius: 5px;
    position: absolute; bottom: 9px; left: 5px;}
.image-slider-button {background: #d1f7db; width: 20px; height:
    20px; border: black solid 1px; border-radius: 10px; top:
    -6px;}
```

De bovenste regel is voor de div, waar de hele sleepbalk in zit. De middelste regel is voor de met de balk van de sleepbalk. De onderste regel is voor de met de knop van de sleepbalk.

Verstreken, resterende en totale speelduur (pagina 3)

```
.elapsed, .remaining, .duration {background: white; width: 42px;
    height: 24px; font-size: 0.8em; line-height: 24px; text-align:
    center; border-bottom: black solid 1px; position: absolute;
    top: 25px; right: -42px;}
```



De elementen met class=”elapsed”, class=”remaining” en class=”duration”. Dit zijn de ’s, waarin de verstreken, resterende en totale speelduur wordt weergegeven. Deze instellingen gelden voor de hele pagina, verderop worden ze voor aparte videospelers aangepast.

Het overgrote deel van de css voor deze drie ’s is hetzelfde. Hieronder wordt alleen voor twee ’s de positie aangepast, zodat ze niet over elkaar heen komen te staan. Witte achtergrond, 42 px breed, 24 px hoog, iets kleinere letter. Regelhoogte 24 px, zodat de tijden verticaal in het midden staan. Horizontaal gecentreerd, border aan de onderkant, absoluut gepositioneerd om ze op de juiste plaats neer te kunnen zetten.

Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf absoluut, relatief of fixed is gepositioneerd. Dat is hier `.image`, de waar de bedieningselementen voor het afspelen in staan.

```
.remaining {top: 50px;}
.duration {top: 0;}
```

De ’s met resterende en totale speelduur op een andere hoogte zetten, dan hier gelijk boven is opgegeven, zodat de drie ’s niet over elkaar heen komen te staan.

Focus (pagina 3)

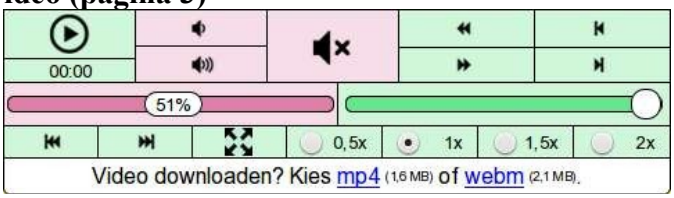
```
.controls button:focus, input:focus, .sound-slider-beam:focus,
    .controls .image-slider-beam:focus {background-color: white;
    outline: none;}
.sound-slider-beam:focus .sound-slider-button {background: red;}
.image-slider-beam:focus .image-slider-button {background: red;}
```

Het aangeven welke knop focus heeft, werkt op precies dezelfde manier als bij Focus op de eerste pagina. Daar is ook de uitleg voor bovenstaande regels te vinden. (Kort samengevat: als ’n element focus heeft, geef er dan ’n witte achtergrond aan. Als de sleepbalk focus heeft, maak dan de sleepknop rood.)

Dit is css die voor álle videospelers op deze pagina geldt. Maar omdat de spelers nogal van uiterlijk verschillen, wordt de css vaak aangepast voor ’n specifieke speler. Die aanpassingen staan bij css speciaal voor ’n bepaalde video.

In Internet Explorer 9, 10 en 11 krijgen sleepbalk en sleepknop niet altijd een andere achtergrondkleur. Zie verder bij Bekende problemen (en oplossingen).

Speciaal voor tweede video (pagina 3)



De verstreken en totale speelduur zijn bij deze speler vervallen. De knop Fullscreen en de snelheidsregeling zijn daarentegen wel aanwezig. Op de knop van de sleepbalk voor geluidssterkte staat het percentage van de geluidssterkte. Ook zijn veel maten aangepast.

Bij deze videospeler wijkt de volgorde van de elementen op het scherm af van de voor deze pagina opgegeven volgorde van de Tabindex. Daarom wordt onder aan de html de tabindex

voor deze videospeler aangepast. Meer daarover bij Het JavaScript onderaan de html-bestanden.

```
#duration-2, #elapsed-2, #aria-elapsed-2, #percentage-2, #sound-
    slider-4, #five-back-4, #five-forward-4, #image-slider-4,
    #duration-5, #elapsed-5, #aria-elapsed-5, #percentage-5
    {display: none;}
```

Door een id te gebruiken, kan bij één of meer videospelers iets worden verborgen, terwijl dat bij de andere videospelers gewoon in gebruik blijft.
Hier worden bij de tweede video de speelduur, de verstreken speelduur, het voorlezen van de verstreken speelduur en de geluidssterkte verborgen.
Om css uit te sparen, wordt dezelfde regel gebruikt om ook bij andere videospelers elementen te verbergen.
Bij de vierde videospeler worden de sleepbalk voor volume, de knoppen Vijf procent terug en Vijf procent verder, en de sleepbalk voor afspelen verborgen.
Bij de vijfde videospeler worden speelduur, verstreken speelduur, voorlezen van de verstreken speelduur en de geluidssterkte verborgen.
Achter de schermen blijft alles gewoon werken, want het script wordt niet veranderd. Als je de gegenereerde code bekijkt, zie je dat de elementen nog steeds gewoon aanwezig zijn, maar niet langer op het scherm zichtbaar zijn.

```
#play-2, #play-5 {background: url(../103-images/buttons-
    background-page-3-klein.png) 8px 1px no-repeat #d1f7db; width:
    95px; height: 35px;}
```

Voor deze elementen geldt ook de eerder bij .play opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met id=”play-2” en id=”play-5”. Dit zijn de Speel-pauzeerknoppen voor de tweede en vijfde videospeler. Omdat de css voor de vijfde speler video grotendeels hetzelfde is als die voor de tweede, wordt die hier ook opgegeven. Waar nodig wordt die dan later aangepast.
De enige verschillen met de eerder opgegeven css zijn de kleinere breedte en hoogte. Daardoor moet ook een kleinere achtergrond-afbeelding worden gebruikt, dan daar is opgegeven.

```
#play-2.not-playing, #play-5.not-playing {background: url(../103-
    images/buttons-background-page-3-klein.png) -73px 1px no-
    repeat #d1f7db;}
```

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Hierdoor kan, als de video niet speelt, een ander symbool worden getoond, dan wanneer de video wel speelt.
Behalve dat het hier om een driehoekje gaat, is het verhaal verder precies hetzelfde als gelijk hierboven bij `#play-2, #play-5`. En uiteraard is de achtergrondpositie anders.

```
#sound-2, #sound-5 {width: 190px; height: 50px;}
```

Voor deze elementen geldt ook de eerder bij .sound opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met id=”sound-2” en id=”sound-5”. Dit zijn de ’s, waarbinnen de bedieningselementen voor de geluidssterkte zitten. Omdat de css voor de vijfde speler video grotendeels hetzelfde is als die voor de tweede, wordt die hier ook opgegeven. Waar nodig wordt die dan later aangepast.

De enige verschillen met de eerder opgegeven css zijn de grotere breedte en kleinere hoogte.

```
#sound-2 .softer, #sound-5 .softer {background: url(../103-
    images/buttons-background-page-3-klein.png) -391px no-repeat
    #f7dde8; width: 95px; height: 26px;}
#sound-2 .louder, #sound-5 .louder {background: url(../103-
    images/buttons-background-page-3-klein.png) -310px no-repeat
    #f7dde8; width: 95px; height: 26px; top: 26px;}
.videobox[data-sound="no"] #softer-2::after, .videobox[data-
    sound="no"] #louder-2::after, .videobox[data-sound="no"]
    #softer-5::after, .videobox[data-sound="no"] #louder-5::after
    {font-size: 34px; line-height: 18px;}
#sound-2 .mute, #sound-5 .mute {background: url(../103-
    images/buttons-background-page-3-klein.png) -154px no-repeat
    #f7dde8; width: 95px; height: 52px; top: 0; left: 95px;}
#sound-2 .mute.muted, #sound-5 .mute.muted {background:
    url(../103-images/buttons-background-page-3-klein.png) -238px
    no-repeat #f7dde8;}
```

Voor deze elementen geld ook de eerder bij Knoppen en percentage volume opgegeven css, voor zover die hier niet wordt veranderd.
De knoppen Zachter, Harder, Geluid aan/uit. Hiervoor geldt precies hetzelfde als iets hierboven bij `#play-2, #play-5` staat: er worden kleinere achtergrond-afbeeldingen gebruikt. Ook de achtergrondpositie wordt aangepast, en bij sommige elementen wordt ook de plaats aangepast.
De derde regel, die met `[data-sound="no"]`, is bedoeld voor iOS, waar het de geluidssterkte alleen met de knoppen op het apparaat kan worden veranderd, zoals beschreven bij .videobox[data-sound=”no”]. Ook de plaats van de ’x’ moet aan de andere maten worden aangepast.

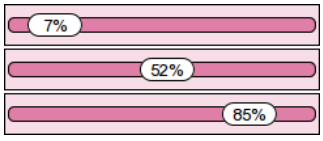
```
#sound-2 .sound-slider, #sound-5 .sound-slider {width: 240px;
    border-bottom: black solid 1px; bottom: -33px; left: -95px;}
```

Voor deze elementen geld ook de eerder bij .sound .sound-slider opgegeven css, voor zover die hier niet wordt veranderd.
De sleepbalk voor de geluidssterkte. Alleen breedte, border en positie worden aangepast.

```
#sound-2 .sound-slider-beam, #sound-5 .sound-slider-beam {width:
    234px;}
```

Voor deze elementen geldt ook de eerder bij .sound .sound-slider-beam opgegeven css, voor zover die hier niet wordt veranderd.
De balk van de sleepbalk voor de geluidssterkte. Alleen de breedte wordt hier aangepast.

```
#sound-slider-button-2, #sound-slider-button-5
```



De geluidssterkte wordt weergegeven op de knop van de sleepbalk voor het geluid.

Voor deze elementen geldt ook de eerder bij .sound-slider-button opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met id=”sound-slider-button-2” en id=”sound-slider-button-5”. De sleepknoppen voor geluid van de tweede en vijfde videospeler. De maten zijn voor beide knoppen gelijk, maar alleen bij de tweede videospeler wordt het percentage van de geluidssterkte op de knop van de sleepbalk weergegeven.

Dit gebeurt met behulp van een door `::after` gemaakt pseudo-element. Om dit voor elkaar te krijgen, zijn voor deze sleepknop wat meer aanpassingen nodig. De sleepknop voor de vijfde videospeler krijgt dit pseudo-element niet, dus daar zie je het percentage niet.

`background: white;`

Witte achtergrond.

`width: 40px; height: 16px`

40 px breed en 16 px hoog, zodat de tekst erop past.

`text-align: center;`

Tekst horizontaal centreren.

`top: -4px;`

Op de goede hoogte zetten.

```
#sound-slider-button-2::after {display: block; font-size: 13px;
    line-height: 16px;}
```

Met behulp van `::after` wordt bij het element met id=”sound-slider-button-2” een pseudo-element gemaakt, waarmee tekst op de sleepknop voor geluid van de tweede videospeler kan worden gezet.

De inhoud van wat er precies op komt te staan, wordt gelijk hieronder opgegeven met behulp van `content`. Hier wordt alleen het uiterlijk bepaald. De inhoud verandert voortdurend en moet dus vaker worden opgegeven, het uiterlijk blijft steeds hetzelfde en kan in één keer worden opgegeven.

Dat uiterlijk is een iets kleinere lettergrootte en een regelhoogte, waarbij de tekst verticaal in het midden van de knop komt te staan.

```
#videobox-2[data-volume="percent-00"] .sound-slider-button::after
    {content: "0%";}
#videobox-2[data-volume="percent-01"] .sound-slider-button::after
    {content: "1%";}
#videobox-2[data-volume="percent-02"] .sound-slider-button::after
    {content: "2%";}
```

(…) nog 95 dezelfde regels met een steeds hoger getal (...)

```
#videobox-2[data-volume="percent-98"] .sound-slider-button::after
    {content: "98%";}
#videobox-2[data-volume="percent-99"] .sound-slider-button::after
    {content: "99%";}
#videobox-2[data-volume="percent-100"] .sound-slider-button::after
    {content: "100%";}
```

Dit is een uiterst omslachtige manier om het percentage weer te geven. Honderdeneen regels code Het is natuurlijk ook absoluut niet de bedoeling dat zoiets ooit in het echt gebruikt zou worden. Maar het is ’n aardige illustratie van wat er mogelijk is, als je met css via JavaScript toegang hebt tot iets als een <video>-element.

`#videobox-2`: het element met id=”videobox-2”. Dat is de waar de tweede videospeler in staat.

`[data-volume="percent-00"]`: Er moet een attribuut ’data-volume’ aanwezig zijn dat de waarde ’percent-00’ heeft. (En in latere regels ’percent-01’, ’percent-02’, enz., t/m ’percent-100’.) Dit attribuut wordt door het script ingevoegd. De waarde is het percentage van de geluidssterkte. Dit wordt door het script bijgehouden, zoals beschreven bij data-volume.

(Meestal zie je de geluidssterkte in tienden. Honderdsten werken echter ook en geven de mogelijkheid procenten te gebruiken om de geluidssterkte in te tonen.)

`.sound-slider-button::after`: het element met class=”sound-slider-button” dat binnen `div#videobox-2` ligt. Dat is er maar eentje: de met de sleepknop van de sleepbalk voor de geluidssterkte. De css voor het pseudo-element dat met `::after` wordt gemaakt, is al hierboven opgegeven, alleen `content` miste nog.

Als het attribuut `data-volume` bij `#videobox-2` een waarde heeft van ’percent-00’ (geluid staat helemaal uit), geef dan het met behulp van `::after` bij `.sound-slider-button` gemaakte pseudo-element een `content` van ’0%’. En die ’0%’ verschijnt dan dus op de speelknop.

Als de geluidssterkte op 11% staat, is de waarde van `data-volume` ’percent-11’ en wordt de inhoud van `content` ’11%’. Enz.

```
#controls-2 button:focus, #mute-2.muted:focus, #videobox-2
    .image .to-begin:focus, #videobox-2 .image .to-end:focus,
    #controls-4 button:focus, #controls-5 button:focus, #mute-
    5.muted:focus, #videobox-5 .image .to-begin:focus, #videobox-
    5 .image .to-end:focus {background-color: white;}
```

De focus is eerder bij Focus geregeld met een vrij algemene regel:

```
.controls button:focus, input:focus, .sound-slider-
    beam:focus, .controls .image-slider-beam:focus
    {background-color: white; outline: none;}
```

Bij allerlei elementen in verschillende spelers wordt echter de achtergrondkleur later aangepast, waardoor deze algemene regel wordt overruled, omdat de latere regels meer specificiteit, meer gewicht, hebben. Iets hierboven bijvoorbeeld wordt met behulp van `#play-2, #play-5` een nieuwe achtergrondkleur gegeven aan de Speel-pauzeerknoppen voor de tweede en vijfde videospeler.

`.controls button:focus`, waarmee de achtergrondkleur ook voor de Speel-pauzeerknoppen wordt opgegeven (ook deze buttons zitten binnen `.controls`), verliest het van het specifiekere `#play-2, #play-5`. Waardoor de met `#play-2, #play-5` opgegeven achtergrondkleur `#d1f7db` (lichtgroen) ook bij focus niet verandert. Dit probleem doet zich bij allerlei bedieningselementen in meerdere videospelers voor. Om de css wat beknopt te houden, wordt dit hier in één keer voor alle spelers verholpen.

`#controls-2 button:focus`: als een <button> in de tweede videospeler focus heeft.

`#mute-2.muted:focus`: als het geluid in de tweede videospeler uitstaat en de Geluid aan-uitknop focus heeft.

`#videobox-2 .image .to-begin:focus`: als de knop Naar begin in de tweede videospeler focus heeft (hier is zelfs nog een extra `.image` nodig om voldoende specifiteit te krijgen).

`#videobox-2 .image .to-end:focus`: als gelijk hierboven, maar voor de knop Naar einde.

`#controls-4 button:focus`: als een <button> in de vierde videospeler focus heeft.

`#controls-5 button:focus`: als een <button> in de vijfde videospeler focus heeft.

`#mute-5.muted:focus`: als het geluid in de vijfde videospeler uitstaat en de Geluid aan-uitknop focus heeft.

`#videobox-5 .image .to-begin:focus`: als de knop Naar begin in de vijfde videospeler focus heeft (hier is zelfs nog een extra `.image` nodig om voldoende specifiteit te krijgen).

`#videobox-5 .image .to-end:focus`: als gelijk hierboven, maar voor de knop Naar einde.

```
#sound-slider-beam-2:focus .sound-slider-button, #sound-slider-
    beam-5:focus .sound-slider-button {background: #f7dde8;}
```

De focus voor de sleepbalk voor geluid is eerder bij Focus geregeld met een vrij algemene regel:

```
.sound-slider-beam:focus .sound-slider-button
    {background: red;}
```

Bij #sound-slider-button-2, #sound-slider-button-5 wordt echter een andere achtergrondkleur opgegeven. Deze selector heeft meer specificiteit, meer 'gewicht', omdat er een id in voorkomt. Daardoor 'wint' deze selector van de meer algemene. Hierdoor verandert de achtergrondkleur ook niet, als de sleepbalk focus heeft. Bovendien wordt de achtergrondkleur verandert in rood, wat bij de tweede videospeler niet zo handig is, omdat hier het percentage van de geluidssterkte op de knop wordt gezet. En zwart op rood leest niet lekker.

Daarom krijgen de tweede en vijfde videospeler hier een selector, die genoeg specificiteit heeft om wel te werken. Bij beide wordt de achtergrondkleur van de sleepknop van de sleepbalk voor geluid verandert in lichtroze, als de sleepbalk focus heeft.

In Firefox en Internet Explorer 9, 10 en 11 krijgen sleepbalk en sleepknop niet altijd een andere achtergrondkleur. Zie verder bij Bekende problemen (en oplossingen).

```
#image-2, #image-5 {width: 195px; height: 52px;}
#videobox-2 .image .to-begin, #videobox-5 .image .to-begin
    {background: url(../103-images/buttons-background-page-3-
    klein.png) -487px no-repeat #d1f7db; width: 68px; height:
    22px; top: 83px; left: -285px;}
#videobox-2 .image .to-end, #videobox-5 .image .to-end
    {background: url(../103-images/buttons-background-page-3-
    klein.png) -566px no-repeat #d1f7db; width: 68px; height:
    22px; border-width: 0 1px 0 0; top: 83px; left: -217px;}
#videobox-2 .fullscreen, #videobox-5 .fullscreen {background:
    url(../103-images/buttons-background-page-3-klein.png) -965px
    no-repeat #d1f7db; display: block; width: 68px; height: 22px;
    top: 83px; left: -149px;}
#image-2 .five-back, #image-5 .five-back {background: url(../103-
    images/buttons-background-page-3-klein.png) -634px no-repeat
    #d1f7db; width: 98px; height: 26px; left: 0;}
#image-2 .ten-back, #image-5 .ten-back {background: url(../103-
    images/buttons-background-page-3-klein.png) -792px no-repeat
    #d1f7db; width: 98px; height: 26px; left: 98px;}
#image-2 .ten-forward, #image-5 .ten-forward {background:
    url(../103-images/buttons-background-page-3-klein.png) -872px
    no-repeat #d1f7db; width: 98px; height: 26px; top: 26px; left:
    98px;}
```

```
#image-2 .five-forward, #image-5 .five-forward {background:
    url(../103-images/buttons-background-page-3-klein.png) -711px
    no-repeat #d1f7db; width: 98px; height: 26px; top: 26px; left:
    0;}
```

Voor al deze elementen geldt de eerder bij Knoppen afspelen opgegeven css, voor zover die hier niet wordt veranderd.

Voor de tweede en vijfde videospeler zijn nogal wat aanpassingen nodig. De hierboven staande aanpassingen zijn allemaal redelijk simpel, ze hebben voornamelijk betrekking op een andere breedte, hoogte, border en/of positie. Omdat de knoppen kleiner zijn dan de eerder opgegeven knoppen, wordt ook een kleinere achtergrond-afbeelding gebruikt ('buttons-background-page-3-klein'), waarbij ook een andere background-position hoort.

Omdat de knop Fullscreen eerder bij Verborgen bedieningselementen is verborgen met `display: none;` krijgt die ook nog `display: block;`. In de tweede en vijfde videospeler kan de video dus fullscreen worden afgespeeld, terwijl dat in de andere spelers niet kan. Verder werkt de knop Fullscreen precies hetzelfde als op de eerste pagina, zoals beschreven is bij .fullscreen[data-fullscreen="on"]::after.

```
#image-2 .image-slider, #image-5 .image-slider {width: 239px;
    border-bottom: black solid 1px; bottom: -31px; left: -44px;}
#image-2 .image-slider-beam, #image-5 .image-slider-beam {width:
    227px;}
#image-2 .image-slider-button, #image-5 .image-slider-button
    {background: white;}
```

Voor deze elementen geldt ook de eerder bij Sleepbalk afspelen opgegeven css, voor zover die hier niet wordt veranderd.

Ook de sleepbalk voor afspelen krijgt in de tweede en vijfde videospeler een andere maat. De bovenste regel is voor de , waarin de sleepbalk staat. De middelste regel is voor de met de balk van de sleepbalk.

In de onderste regel krijgt de met de knop van de sleepbalk een witte achtergrond.

```
#image-2 .image-slider-beam:focus .image-slider-button, #image-
    5 .image-slider-beam:focus .image-slider-button {background:
    #d1f7db;}
```

Voor deze elementen geldt ook de eerder bij Focus opgegeven css, voor zover die hier niet wordt veranderd.

Bij de eerder opgegeven css wordt de achtergrondkleur van de met de sleepknop voor afspelen, als de sleepbalk focus heeft, veranderd naar knalbrandweerrood. Hier wordt opgegeven dat de achtergrondkleur bij focus lichtgroen moet worden. Waarom? Daarom.

Overigens zou die eerdere css ook niet werken, omdat daar als selector `.image-slider-beam:focus .image-slider-button` wordt gebruikt. Gelijk hierboven wordt de achtergrondkleur met `#image-2 .image-slider-beam, #image-5 .image-slider-beam` in wit veranderd. Omdat in die laatste selector een id zit, heeft die meer specificiteit ('gewicht') dan de eerdere selector en 'wint' daardoor: de achtergrondkleur verandert niet bij focus.

De hier gebruikte selector heeft evenveel specificiteit (feitelijk zelfs iets meer door het gebruikte `:focus`) en werkt dus wel.

In Internet Explorer 9, 10 en 11 krijgen sleepbalk en sleepknop niet altijd een andere achtergrondkleur. Zie verder bij Bekende problemen (en oplossingen).

`#remaining-2, #remaining-5 {background: #d1f7db; width: 94px; height: 16px; line-height: 16px; border: black solid; border-width: 0 1px 1px 0; top: 35px; left: -285px;}`

Voor deze elementen geldt ook de eerder bij Verstreken, resterende en totale speelduur opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met id=”remaining-2” en id=”remaining-5”. De ’s in de tweede en vijfde videospeler, waarin de resterende speelduur wordt weergegeven.
Achtergrondkleur, breedte, hoogte, regelhoogte, border en positie worden aangepast aan de andere maten van de bedieningselementen in de tweede en vijfde videospeler.

`#image-2 .speed, #image-5 .speed`

De elementen met een class=”speed” die binnen een element met id=”image-2” of id=”image-5” liggen. Dit zijn de ’s, waarbinnen de snelheidsregeling voor de tweede en de vijfde videospeler zit.
Omdat de css voor de vijfde videospeler vrijwel hetzelfde is, staat die ook hier.

`display: block;`

Bij Verborgen bedieningselementen is de snelheidsregeling voor alle videospelers op deze pagina met behulp van `.speed {display: none;}` verborgen. De class ”speed” geldt voor alle ’s met een snelheidsregeling, dus in principe wordt de snelheidsregeling overal verborgen.
Elk bedieningselement heeft echter ook een id. Door die id te gebruiken, kan bij sommige spelers de snelheidsregeling toch weer zichtbaar worden gemaakt, zoals hier voor de tweede en vijfde videospeler gebeurt.
Maar waar het om gaat: het is mogelijk de radioknoppen zelf te tonen. En met ’n andere indeling zou ook nog wel wat mooier kunnen.

`width: 276px; height: 22px;`

Breedte en hoogte.

`line-height: 22px;`

Met deze regelhoogte komt de tekst bij de radioknoppen ongeveer verticaal in het midden te staan.

`position: absolute; top: 83px; left: -81px;`

Op de goede plaats neerzetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een relatieve, absolute of fixed positie heeft. Dat is hier `div#image-2` of `div#image-5`.

`#speed-2 input`

De <input>’s binnen het element met id=”speed-2”. De radioknoppen die de snelheid van de tweede videospeler regelen.

`height: 20px; margin: 0; padding: 0;`

Verschillende browsers hebben een verschillende standaardhoogte, marge en padding bij <input>. Hiermee worden ze overal het zelfde.

`position: absolute; top: 1px; right: 250px;`

De <input> op de goede plaats zetten. Voor de tweede, derde en vierde <input> wordt `right` later aangepast, anders zouden alle vier de <input>’s over elkaar heen komen te staan.
Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `div#image-2`. Om een of andere mij niet geheel duidelijke reden moet vanaf rechts worden gepositioneerd. Als je vanaf links positioneert, plaatsen Opera, Google Chrome en Safari de radioknoppen veel te ver naar links.

`z-index: 50;`

In de html volgt na de <input> de bijbehorende <label>. Dat <label> wordt over de <input> gezet, waardoor de <input> onzichtbaar wordt. Door een hogere z-index aan de <input>'s te geven, worden deze toch boven de <label> gezet en zijn dus zichtbaar.
Een z-index werkt alleen als het element absoluut, relatief of fixed is gepositioneerd. Hierboven zijn de <input>'s absoluut gepositioneerd, dus aan die voorwaarde is voldaan.

```
#speed-2 .speed-default {right: 180px;}
#speed-2 .speed-one-half {right: 110px;}
#speed-2 .speed-double {right: 40px;}
```

Voor deze elementen geldt ook de gelijk hierboven bij `#speed-2 input` opgegeven css, voor zover die hier niet wordt veranderd.
De radioknoppen voor normale, anderhalve en dubbele snelheid van de tweede videospeler. Om te voorkomen dat alle radioknoppen over elkaar heen komen te staan, worden ze allemaal op 'n eigen plekje gezet. De radioknop voor halve snelheid is gelijk hierboven al op de goede plaats gezet.

`#speed-2 label, #speed-5 label`

De <label>'s binnen de elementen met id="speed-2" en id="speed-5". De <label>'s bij de radioknoppen voor snelheidsregeling in de tweede en vijfde videospeler.
Een groot deel van de css voor deze <label>'s in de vijfde videospeler is het zelfde als voor de <label>'s in de tweede speler. Daarom wordt hier voor beide spelers in één keer de css opgegeven. Waar nodig wordt die dan later voor de vijfde speler aangepast.

`background: #d1f7db;`

Lichtgroene achtergrond.

`width: 57px; height: 22px;`

Breedte en hoogte. De <label>'s vullen de volle ruimte van de knop, zodat klikken en aanraken over de volle breedte van de knop werken. Zonder de breedte zou klikken en aanraken niet werken op de gele achtergrond op de afbeelding.

`font-size: 13px;`

Lettergrootte.

`text-align: right;`

Standaard staat tekst links. Hier wil ik de snelheden rechts van de radioknoppen hebben.

`border-right: black solid 1px;`

Randje rechts.

`padding: 0 10px 0 1px;`

Padding in de volgorde boven – rechts – onder – links. Om de tekst op de juiste plaats te krijgen.

`position: absolute; left: 0;`

Op de juiste plaats zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die een absolute, relatieve of fixed positie heeft. Dat is hier `div#image-2` of `div#image-5`.

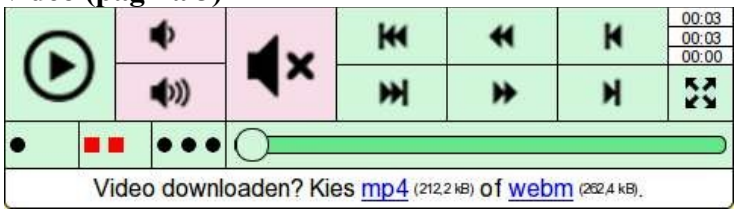
```
#speed-2 .speed-default-label {left: 69px;}
#speed-2 .speed-one-half-label {left: 138px;}
#speed-2 .speed-double-label {width: 58px; left: 207px;}
```

De <label>'s van de tweede, derde en vierde radioknop in de tweede videospeler op de goede plaats zetten. Hier gelijk boven zijn ze absoluut gepositioneerd en helemaal links neergezet. De eerste <label> mag daar blijven staan, de andere krijgen hier een eigen plaatsje.
De laatste <label> is 1 px breder dan hier gelijk boven is opgegeven, zodat de border rechts precies boven de border van `div#controls-2` komt te staan.

```
#speed-2 input:focus + label {background: white;}
```

Bij Focus is voor de videospelers op deze pagina opgegeven, hoe focus herkend kan worden bij de bedieningselementen. Maar dat is daar niet geregeld voor de snelheidsregeling, omdat die bij de meeste spelers ontbreekt. Daarom wordt dat hier alsnog gedaan voor deze speler.
`#speed-2 input:focus`: als een <input> binnen het element met id="speed-2" focus heeft. Als een radioknop voor de snelheidsregeling focus heeft.
`label`: doe dan iets met de gelijk op die <input> volgende <label>. Het script voegt gelijk na de <input> de bijbehorende <label> in, dus dit is altijd de juiste <label>.
Als een <button> focus heeft, maak dan de achtergrond van de gelijk daarop volgende <label> wit.

Speciaal voor derde video (pagina 3)



Snelheidsregeling (met maar drie knoppen) en de knop Fullscreen zijn aanwezig, maar de sleepbalk voor de geluidssterkte ontbreekt. De stand van de snelheid wordt met behulp van een achtergrond-afbeelding aangegeven. Ook zijn veel maten aangepast.

```
#aria-volume-3, #percentage-3 {display: none;}
```

De elementen met id="aria-volume-3" en id="percentage-3". De om in schermlezers een gewijzigde geluidssterkte voor te lezen en de voor weergave van de geluidssterkte in de derde videospeler.
In deze videospeler wordt de geluidssterkte niet weergegeven of voorgelezen, dus verbergen.

```
#sound-3 .mute {height: 75px; top: 0;}
```

Voor dit element geldt ook de eerder bij .sound .mute opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class="mute" binnen het element met id="sound-3". De Aan-uitknop voor geluid van de derde videospeler.
Omdat het percentage van de geluidssterkte hier niet wordt weergegeven, moet de Aan-uitknop voor geluid een grotere hoogte krijgen, dan eerder is opgegeven. Bovenaan neerzetten.

```
#elapsed-3, #remaining-3, #duration-3 {height: 12px; font-size:
    0.7em; line-height: 13px; top: 25px;}
#remaining-3 {top: 13px;}
#duration-3 {top: 0;}
```

Voor deze elementen geldt ook de eerder bij Verstreken, resterende en totale speelduur opgegeven css, voor zover die hier niet wordt veranderd.
Voor de derde videospeler worden verstreken, resterende en totale speelduur alleen wat kleiner gemaakt. Daardoor moet ook de positie e.d. iets worden aangepast.
Om te voorkomen dat ze alle drie over elkaar heen komen te staan, krijgen ze alle drie een eigen `top`.

```
#videobox-3 .fullscreen {background: url(../103-images/buttons-
    background-page-3-klein.png) -977px no-repeat #d1f7db;
    display: block; width: 42px; height: 37px; border-width: 0 0
    1px; top: 38px; right: -42px;}
```

De knop Fullscreen in de derde videospeler.
Voor dit element geldt ook de eerder bij Knoppen afspelen opgegeven css, voor zover die hier niet wordt veranderd.
Hier worden alleen wat simpele aanpassingen aan de grootte, positie, e.d. gedaan. Voor het symbool wordt de achtergrond-afbeelding 'buttons-background-page-3-klein', gebruikt, met de juiste background-position.
Omdat de knop Fullscreen eerder bij Verborgen bedieningselementen is verborgen met `display: none;` moet ook nog `display: block;` worden toegevoegd. Verder werkt de knop Fullscreen precies hetzelfde als op de eerste pagina, zoals beschreven is bij .fullscreen[data-fullscreen="on"]::after.

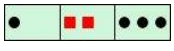
```
#videobox-3 .fullscreen:focus {background-color: white;}
```

De knop Fullscreen in de derde videospeler, als deze focus heeft.
Bij Focus is eerder het herkennen van knoppen met focus geregeld. Omdat de knop Fullscreen echter bij Verborgen bedieningselementen is verborgen, deed die daar niet mee. Daarom wordt hier alsnog opgegeven dat de knop Fullscreen in de derde videospeler bij focus een witte achtergrond moet krijgen.

```
#videobox-3 .sound-slider {display: none;}
```

In de derde videospeler wordt de sleepbalk voor geluid verborgen.

```
#image-3 .speed {display: block; width: 146px; height: 30px;
    position: absolute; bottom: 0; left: -219px;}
```



De elementen met class="speed" binnen het element met id="image-3". De waar de snelheidsregeling van de derde videospeler in zit.
Bij Verborgen bedieningselementen is de snelheidsregeling op deze pagina verborgen met behulp van `display: none;`. Voor de derde videospeler wordt deze weer zichtbaar gemaakt met behulp van `display: block;`.
De rest van de css is om `div.speed` op de juiste plaats te zetten e.d.

```
#speed-3 input, #speed-5 input {position: absolute; left:
    -20000px;}
```

Verberg in de derde en vijfde videospeler de <input>'s die de snelheid regelen. Radioknoppen zijn niet echt heel mooi, daarom verberg ik ze. Verbergen met

`display: none;` of `visibility: hidden;` is geen goed idee, want dan worden ze door schermlezers genegeerd. Daarom worden ze links buiten het scherm geparkeerd. Ze werken nog steeds, maar je ziet ze niet meer.
De bijbehorende <label>'s staan gewoon op het scherm, dus aanraken en klikken blijven gewoon werken. Niet op de knoppen – het is vrij zinloos om 'n meter links van het scherm in de lucht te gaan lopen graaien –, maar wel op de via het `for`-attribuut aan de knop gekoppelde <label>.

`#speed-3 .speed-double, #speed-3 .speed-double-label {display: none;}`

De radioknop en het bijbehorende label voor dubbele snelheid in de derde videospeler. Deze videospeler heeft maar drie snelheden, de dubbele snelheid wordt verborgen.

`#speed-3 label`

De <label>'s in het element met id="speed-3". De <label>'s die horen bij de knoppen voor de snelheidsregeling in de derde videospeler.
De meeste css voor deze <label>'s is hetzelfde, die kan hier in één keer worden opgegeven. Waar nodig wordt die dan later aangepast voor bepaalde <label>'s.

`background: url(../103-images/buttons-background-page-3-klein.png) -1360px no-repeat #d1f7db;`

De zwarte en rode stippen en vierkantjes zijn afkomstig van een achtergrond-afbeelding. Op deze achtergrond-afbeelding is een hele reeks verschillende kleinere afbeeldingen gemonteerd (een 'sprite'), zodat de server maar één keer hoeft te worden aangeroepen voor 'n hele reeks afbeeldingen.
Met behulp van background-position wordt het juiste deel van de achtergrond-afbeelding op de juiste plaats gezet. Deze waarde hoort bij de zwarte enkele stip die bij halve snelheid hoort. Voor andere <label>'s (en als de bij de <label> horende <button> focus heeft) hoeft nu alleen de background-position gewijzigd te worden, om een andere achtergrond-afbeelding te krijgen.

`color: transparent;`

Het script zet tekst op de <label>'s voor snelheidsregeling. Standaard is die tekst 0,5x, 1x, 1,5x en 2x (Hoe je die tekst eventueel kunt wijzigen, staat bij Text). Die tekst is fantastisch en wonderschoon en intelligent en getuigt van 'n diepe rekenkundige wijsheid en zo. Maar nu even niet, omdat ik stippen en vierkantjes gebruik om de snelheid aan te geven.
Dat script doet gewoon blind haar werk, of dat nou nodig is of niet, maar door simpel de voorgrondkleur doorzichtig te maken, zie je de tekst niet meer.
Een andere manier is om de hele tekst bovenin het script te verwijderen, maar dat verwijdert de tekst dan op de hele pagina, in alle videospelers. Hoe je dat kunt doen, staat ook bij Text.

`display: inline-block;`

<label> is een inline-element. Dat betekent o.a. dat je er geen breedte en hoogte aan kunt geven. Bij een blok-element kan dat we. Maar een blok-element komt op 'n nieuwe regel te staan, en ik wil de <label>'s naast elkaar hebben. `inline-block` is het beste uit twee werelden: de <label>'s blijven naast elkaar staan, maar eigenschappen als breedte en hoogte zijn gewoon te gebruiken.

`width: 48px; height: 30px;`

Breedte en hoogte.

`border-right: black solid 1px;`

Randje rechts.

```
#speed-3 .speed-default-label {background-position: -1410px;
    width: 47px;}
#speed-3 .speed-one-half-label {background-position: -1460px;}
```

Voor deze elementen geldt ook de gelijk hierboven bij `#speed-3 label` opgegeven css, voor zover die hier niet wordt veranderd.
De <label>'s bij de knop voor normale en voor anderhalve snelheid in de derde videospeler. Door de background-position te veranderen, verschijnt een andere afbeelding in de <label>. Bij de <label> voor de normale snelheid wordt de breedte 1 px minder gemaakt, omdat de drie <label>'s anders net niet naast elkaar in `div#speed-3` passen. Zo'n miniem verschil is niet te zien.

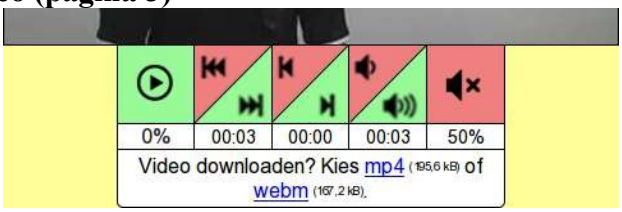
```
#speed-3 input:focus + label {background-color: white;}
```

De <label>'s die gelijk op een <input> volgen binnen het element met id="speed-3", maar alleen als de <input> focus heeft.
Het script zet achter elke <input> een bijbehorend <label>. Als de <input> focus heeft, maak dan de achtergrond van de daarop volgende <label> wit. Meer over wat focus is en waarom het belangrijk is dat aan te geven, kun je vinden bij Focus.

```
#speed-3 .speed-half:checked + .speed-half-label {background-
    position: -1510px;}
#speed-3 .speed-default:checked + .speed-default-label
    {background-position: -1560px;}
#speed-3 .speed-one-half:checked + .speed-one-half-label
    {background-position: -1610px;}
```

Drie regels die grotendeels hetzelfde zijn. De eerste regel is voor de <label> bij de eerste radioknop voor halve snelheid, de tweede voor de <label> bij de knop voor normale snelheid, en de derde voor de <label> bij de knop voor anderhalve snelheid.
`#speed-3`: de elementen moeten binnen de met de snelheidsregeling voor de derde videospeler zitten.
`.speed-half:checked`: de elementen met class="speed-half", maar alleen als deze aangevinkt zijn. Binnen `#speed-3` zit maar één zo'n element: de radioknop waarmee je voor halve snelheid kiest.
`.speed-half-label`: de elementen met class="speed-half-label" die gelijk volgen op de elementen met class="speed-half". Binnen `#speed-3` is dit er maar één: de <label> die bij de radioknop hoort, waarmee je voor halve snelheid kiest.
De eerste regel samengevat: verander de background-position van de <label> bij de radioknop voor halve snelheid in de derde videospeler, maar alleen als die radioknop is aangevinkt.
Iets hierboven bij `#speed-3 label` is een achtergrond-afbeelding opgegeven, die zelf weer bestaat uit een serie kleine afbeeldingen. Door simpelweg de background-position te veranderen, verschijnt er een andere achtergrond-afbeelding in de <label>. De zwarte cirkel verandert in een rood vierkant.
Niet alleen de kleur van de achtergrond-afbeelding verandert, ook de vorm. Mensen die moeite hebben met kleuren, kunnen dan de andere vorm herkennen.
De tweede en derde regel zijn precies hetzelfde, maar dan voor de <label>'s bij de radioknoppen voor normale en anderhalve snelheid. De background-position is dus ook iets anders, omdat er een andere achtergrond-afbeelding moet verschijnen dan bij de <label> voor halve snelheid.

Speciaal voor vierde video (pagina 3)



De knoppen Vijf procent verder en Vijf procent terug ontbreken, net als de sleepbalken voor afspelen en geluidssterkte. Linksonder wordt de verstreken speelduur in procenten weergegeven. De knoppen Naar begin, Naar einde, Tien seconden verder, Tien seconden terug, Harder en Zachter vormen twee aan twee een diagonaal in tweeën verdeeld vierkant. Bij deze videospeler wijkt de volgorde van de elementen op het scherm af van de voor deze pagina opgegeven volgorde van de Tabindex. Daarom wordt onder aan de html de tabindex voor deze videospeler aangepast. Meer daarover bij Het JavaScript onderaan de html-bestanden.

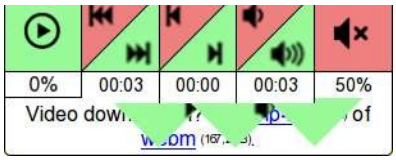
Bij #duration-2... zijn de sleepbalken voor afspelen en geluidssterkte en de knoppen Vijf procent verder en Vijf procent terug voor deze videospeler verborgen met behulp van

```
#sound-slider-4, #five-back-4, #five-forward-4, #image-slider-4 {display: none;}
#videobox-4 .groot {max-width: 296px;}
```

Het element met class=”groot” binnen het element met id=”videobox-4”. De <p> waarbinnen de download-links staan.
De css hiervoor is precies hetzelfde als op de eerste pagina, alleen is de breedte minder, zodat deze <p> met de erin zittende tekst even breed is als de bedieningselementen erboven.

```
#controls-4 {width: 302px; height: 80px; overflow: hidden; position: relative;}
```

Voor dit element geldt ook de eerder bij Bedieningselementen algemeen opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”controls-4”. De waarbinnen de bedieningselementen voor de vierde videospeler staan.
Hoogte en breedte worden aangepast aan de maten van de in de zittende knoppen e.d.



De driehoekige knoppen zijn in werkelijkheid vierkanten, die deels worden gedraaid en over elkaar heen worden gezet. Standaard staat `overflow` op `visible`, waardoor je ook ziet wat niet binnen een element past. Normaal genomen is dat ook, wat je wilt. Mogelijk wordt de lay-out verstoord, maar in ieder geval zie je alles.
Hier wil ik niet zien wat niet binnen de past, want anders knallen aan de onderkant de nep-driehoeken eruit, zoals op de afbeelding is te zien. Dat lijkt me te veel op slecht gepoetste haaientanden. Met behulp van `overflow: hidden;` toveren we de haaientanden aan de onderkant van `div#controls-4` weg.
Nakomelingen van een element kunnen alleen ten opzichte van dat element worden gepositioneerd, als het element een relatieve, fixed of absolute positie heeft. Daarom krijgt de een relatieve positie. Omdat er verder geen `left` of zo bij wordt opgegeven, heeft dit verder geen invloed op deze zelf.

```
#videobox-4 .play {background: url(../103-images/buttons-
    background-page-3-klein.png) -10px 13px no-repeat #f08080;
    width: 60px; height: 60px; border-width: 0 1px 0 0;}
```

Voor dit element geldt ook de eerder bij Speel-pauzeerknop opgegeven css, voor zover die hier niet wordt veranderd.
De Speel-pauzeerknop van de vierde speler. Er zijn alleen wat kleine aanpassingen. Omdat er wat minder ruimte is in de knop, wordt een kleinere achtergrond-afbeelding gebruikt. Breedte en hoogte worden aangepast, en de kanten waar een border wordt neergezet. De achtergrondkleur is iets donkerder.

```
#videobox-4 .play.not-playing {background: url(../103-
    images/buttons-background-page-3-klein.png) -91px 13px no-
    repeat #98fb98;}
```

Zelfde verhaal als gelijk hierboven. Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Hierdoor kan, als de video niet speelt, een ander symbool worden getoond, dan wanneer de video wel speelt.

```
#videobox-4 .play.not-playing:focus {background-color: white;}
```

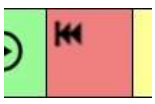
De focus voor de <button>’s is grotendeels al geregeld bij #controls-2 button:focus...
(Meer over wat focus is, staat bij Focus voor de eerste pagina.)
Bij Focus is voor deze pagina geregeld, dat het bedieningselement met focus een witte achtergrond krijgt. Daar is de selector `.controls button:focus` gebruikt. In principe geldt die selector ook voor de Speel-pauzeerknop van de videospelers.
Maar gelijk hierboven is met de selector `#videobox-4 .play.not-playing` een groene achtergrondkleur opgegeven. Deze selector heeft meer specificiteit, meer ’gewicht’, dan de eerdere selector, o.a. omdat er een id in wordt gebruikt. Als de video niet speelt en de knop de class ”not-playing” heeft, zou daardoor de achtergrond altijd groen blijven, ook als de knop focus heeft.
Hier wordt een selector met meer specificiteit dan die bij `.controls button:focus` gebruikt voor de witte achtergrond, waardoor dit ook bij deze knop werkt.

```
#videobox-4 .image {height: 80px;}
```

Voor dit element geldt ook de eerder bij .image opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”image” binnen het element met id=”videobox-4”. De waar de bedieningselementen voor het afspelen in staan voor de vierde videospeler.

Alleen de hoogte wordt hier aangepast. Dat is feitelijk alleen nodig voor Opera op Linux, omdat daar anders de knoppen om een of andere reden niet worden afgekapt en er aan de onderkant uitsteken, zoals op de afbeelding is te zien. Opera op Linux werkt nog met Presto,
maar stapt binnenkort ook over op weergave machine Blink, waarmee dit zou zijn opgelost. In principe maak ik daarom geen aanpassingen meer voor Opera op Linux, maar dit is een heel kleine, dus vooruit.

```
#videobox-4 .to-begin
```



Voor dit element geldt ook de eerder bij .image button en .image .to-begin opgegeven css, voor zover die hier niet wordt veranderd.
Het element met class=”to-begin” binnen het element met id=”videobox-4”. Dit is de knop Naar begin van de vierde videospeler.

In werkelijkheid is deze knop gewoon vierkant. Op de afbeelding zijn de andere knoppen (behalve de Speel-pauzeerknop) even weggelaten en zie je de volledige knop.

`background-color: #f08080; background-position: -506px -8px;`

Bij .image .to-begin is een achtergrondkleur opgegeven, die hier wordt veranderd naar iets roodachtigs. Van de daar al opgegeven achtergrond-afbeelding moet alleen de positie worden aangepast aan de andere afmetingen van deze knop.

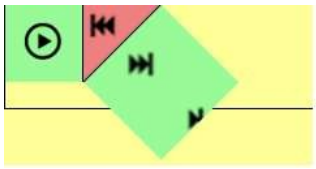
`width: 60px; height: 60px;`

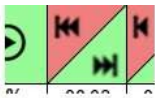
Breedte en hoogte van de knop.

`border-width: 0 1px 0 0;`

Alleen rechts een border.

`#videobox-4 .to-end`





Voor dit element geldt ook de eerder bij .image button en .image to-end opgegeven css, voor zover die hier niet wordt veranderd.
Het element met class=”to-end” binnen het element met id=”videobox-4”. Dit is de knop Naar eind van de vierde videospeler.
Rechts op de afbeelding staat de knop, zoals die uiteindelijk is te zien. Links staat de ’echte’ knop: gewoon een volledig vierkant. Met behulp van `transform` wordt die knop gedraaid.
De rode driehoek links is in werkelijkheid het rode vierkant van de knop Naar begin, zoals die gelijk hierboven is opgegeven. Omdat deze groene knop daaroverheen komt te staan, blijft er van het rode vierkant slechts ’n driehoek zichtbaar.
Aan de onderkant van het groene vierkant komen speelduur e.d. over de knop te staan, waardoor het groene vierkant aan de onderkant horizontaal wordt afgekapt, zoals op de rechterafbeelding is te zien. Aan de rechterkant van het groene vierkant knop Tien seconden terug te staan, die het groene vierkant aan de rechterkant verticaal afkapt. Door het onderaan en rechts afkappen van het vierkant, is het uiteindelijke resultaat een groene driehoek.
Samen met de rode knop Naar begin vult deze knop nu een vierkant, dat diagonaal tussen beide knoppen is verdeeld.

`background-color: #98fb98; background-position: -1388px 18px;`

Bij .image to-end is een achtergrondkleur opgegeven, die hier wordt veranderd naar iets groenachtigs. Van de daar al opgegeven achtergrond-afbeelding moet alleen de positie worden aangepast aan de andere afmetingen van deze knop.

`width: 85px; height: 84px;`

De knop Naar begin heeft een breedte en hoogte van 60 px. Omdat deze knop wordt gedraaid, moeten breedte en hoogte groter zijn dan die van de rode knop Naar begin. Het zichtbare deel dat uiteindelijk overblijft, de groene driehoek, sluit dan goed aan op de knop Naar begin.

`border-width: 0 0 0 1px;`

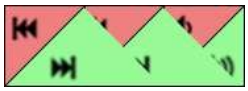
Alleen links een randje. Dit is de zwarte diagonaal die beide knoppen in het vierkant scheidt.

`top: 18px; left: 17px;`

Hieronder wordt de knop gedraaid. Dat draaien gebeurt normaal genomen rondom het middelpunt van het element, hier het centrum van het groene vierkant. Zonder correctie komt dat niet helemaal goed uit, zoals op de afbeelding is te zien.
Met behulp van een al eerder opgegeven absolute positie wordt de knop op de goede plaats gezet. (Je kunt hiervoor ook `transform-origin` gebruiken, maar dit vind ik hier makkelijker.)

```
-ms-transform: rotate(45deg); -webkit-transform:
    rotate(45deg); transform: rotate(45deg);
```
Hier staat eigenlijk drie keer hetzelfde: `transform: rotate(45deg);`.
Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Draai het vierkant 45 graden. Dat is precies de hoek tussen horizontaal en verticaal in. Hierdoor komt de linkerzijkant precies in de goede richting te staan.

```
#videobox-4 .ten-back {background-color: #f08080; background-
    position: -830px -8px; width: 60px; height: 60px; border-
    width: 0 1px; left: 60px; z-index: 10;}
#videobox-4 .ten-forward {background-color: #98fb98; background-
    position: -1448px 18px; width: 85px; height: 84px; border-
    width: 0 0 0 1px; top: 18px; left: 77px; z-index: 20; -ms-
    transform: rotate(45deg); -webkit-transform: rotate(45deg);
    transform: rotate(45deg);}
```

Voor deze elementen geldt ook de eerder bij .image button opgegeven css, voor zover die hier niet wordt veranderd. Voor .ten-back geldt ook de bij .image .ten-back en voor .ten-forward de bij .image .ten-forward opgegeven css, voor zover die hier niet wordt veranderd.
De knoppen Tien seconden terug en Tien seconden verder van de vierde videospeler.
Voor deze twee knoppen geldt precies hetzelfde verhaal als gelijk hierboven bij `#videobox-4 .to-begin` en `#videobox-4 .to-end` is opgegeven. Alleen zijn positie e.d. natuurlijk iets anders.

Verder hebben deze knoppen een z-index nodig. Het script voegt de knoppen in een bepaalde volgorde in. De weergave van de knoppen op het scherm heeft een andere volgorde. Om te voorkomen dat knoppen die later in de html staat over eerdere worden heengezet, krijgen sommige knoppen een z-index. Zonder z-index wordt de volgorde van de html aangehouden, wat er uit zou zien zoals op de afbeelding.

```
#sound-4 {width: 121px; height: 60px; border-left: black solid
    1px; position: absolute; right: 0; z-index: 30;}
```
Het element met id=”sound-4”. De waar de geluidsregeling voor de vierde videospeler in staat.
Alleen wat maten e.d., en een absolute positie om de op de goede plaats te kunnen zetten. Waarom de z-index nodig is, staat gelijk hierboven in de laatste alinea van de vorige css-regel.

```
#videobox-4 .softer {background-color: #f08080; background-
     position: -430px -8px; width: 60px; height: 60px; border-
     width: 0 1px 0 0;}
#videobox-4 .softer::after {display: none;}
#videobox-4 .louder {background-color: #98fb98; background-
     position: -1512px 24px; width: 85px; height: 84px; border-
     width: 0 0 0 1px; top: 17px; left: 18px; -ms-transform:
     rotate(45deg); -webkit-transform: rotate(45deg); transform:
     rotate(45deg);}
#videobox-4 .louder::after {display: none;}
```

Voor `.softer` en `.louder` geldt ook de eerder bij .sound button, .sound span opgegeven css, voor zover die hier niet wordt veranderd. Voor `.softer` geldt ook de bij .sound .softer, voor `.louder` ook de bij .sound .louder opgegeven css, voor zover die hier niet wordt veranderd.
De knoppen Zachter en Harder van de vierde videospeler. Het verhaal is precies hetzelfde als hier gelijk boven voor de knoppen Tien seconden verder en Tien seconden terug.
Bij .videobox[data-sound="no"]... wordt op iOS met behulp van `::after` een pseudo-element met een kruis over de geluidsregeling gezet. Dat kruis wil ik hier weghebben. Door simpelweg dat pseudo-element te verbergen, zijn de knoppen ontkruist.

```
#videobox-4 .mute {background: url(../103-images/buttons-
     background-page-3-klein.png) -172px no-repeat #f08080; width:
     61px; height: 60px; border-width: 0 0 0 1px; top: 0; left:
     60px;}
```

De Geluid aan-uitknop van de vierde videospeler.
Juiste maat e.d. geven voor deze speler. Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen.

```
#videobox-4 .mute::after {display: none;}
```

Bij .videobox[data-sound="no"]... wordt op iOS met behulp van `::after` een pseudo-element met een kruis over de geluidsregeling gezet. Dat kruis wil ik hier weghebben. Door simpelweg dat pseudo-element te verbergen, is de Geluid aan-uitknop ontkruist.

```
#videobox-4 .mute.muted {background-color: #98fb98; background-
     position: -256px;}
```

Zelfde verhaal als iets hierboven bij `#videobox-4 .mute`. Aan de Geluid aan-uitknop wordt door het script een class="muted" toegevoegd, als het geluid uitstaat. Hierdoor kan, als het geluid uitstaat, een ander symbool worden getoond, dan wanneer het geluid aanstaat.

```
#videobox-4 .mute.muted:focus {background-color: white;}
```

De focus voor de <button>'s is grotendeels al geregeld bij #controls-2 button:focus...
Zonder deze extra selector zou de achtergrond van de Geluid aan-uitknop bij focus niet wit worden, als het geluid uitstaat. Verder precies hetzelfde verhaal als bij #videobox-4 .play.not-playing:focus, maar nu voor de Geluid aan-uitknop.

`#videobox-4 .play::after`

Met behulp van `::after` wordt bij de Speel-pauzeerknop van de vierde videospeler een pseudo-element aangemaakt, dat hier wordt gebruikt om de verstreken speelduur in procenten weer te geven. In procenten, want dat wordt namelijk als inhoud van het data-attribuut `data-played` door het script ingevoegd bij `div#videobox-4`. Als je de verstreken speelduur in tijd wilt weergeven, kan dat al, want daar is een speciaal element voor (een met class=”elapsed”).
De eigenlijke inhoud van het pseudo-element wordt met behulp van `content` opgegeven. Dat gebeurt iets verder, omdat die inhoud steeds verandert. Maar alle andere instellingen voor het pseudo-element kunnen hier in één keer worden opgegeven.

```
background: white; width: 59px; line-height: 19px; font-size:
    16px; border: black solid; border-width: 1px 1px 0 0;
    position: absolute; bottom: 0; left: 0;
```

Alleen wat instellingen voor de juiste positie, kleur, e.d.

```
#videobox-4[data-played="percent-00"] .play::after {content:
    "0%";}
#videobox-4[data-played="percent-01"] .play::after {content:
    "1%";}
#videobox-4[data-played="percent-02"] .play::after {content:
    "2%";}
```

(…) nog 95 dezelfde regels met een steeds hoger getal (...)

```
#videobox-4[data-played="percent-98"] .play::after {content:
    "98%";}
#videobox-4[data-played="percent-99"] .play::after {content:
    "99%";}
#videobox-4[data-played="percent-100"] .play::after {content:
    "100%";}
```

Ook dit is alleen bedoeld als illustratie, wat er mogelijk is. 101 regels code alleen om de verstreken speelduur in procenten weer te geven is ietwat veel.

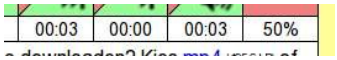
`#videobox-4`: het element met id=”videobox-4”. Dat is de waar de vierde videospeler in staat.

`[data-played="percent-00"]`: Er moet een attribuut ’data-played’ aanwezig zijn dat de waarde ’percent-00’ heeft. (En in latere regels ’percent-01’, ’percent-02’, enz., t/m ’percent-100’.) Dit attribuut wordt door het script ingevoegd. De waarde is het percentage van de verstreken speelduur van de video. Dit percentage wordt door het script bijgehouden.

`.play::after`: het element met class=’play’ dat binnen `div#videobox-4` ligt. Dat is er maar eentje: de Speel-pauzeerknop. De css voor het pseudo-element dat met `::after` wordt gemaakt is al hierboven opgegeven, alleen `content` miste nog.

Als het attribuut `data-played` bij `#videobox-4` een waarde heeft van ’percent-00’ (de video staat helemaal aan het begin), geef dan het met behulp van `::after` bij `.play` gemaakte pseudo-element een `content` van ’0%’. En die ’0%’ verschijnt dan dus op het scherm op de aangegeven plaats.
Als 10% van de video is afgespeeld, is de waarde van `data-played` ’percent-10’ en wordt de inhoud van `content` ’10%’. Enz.

```
#videobox-4 .percentage, #videobox-4 .duration, #videobox-4
    .elapsed, #videobox-4 .remaining {background: white; width:
    60px; line-height: 19px; font-size: 14px; border: black solid
    1px; border-width: 1px 1px 0 0; position: absolute; top: 60px;
    left: 0; z-index: 50;}
```

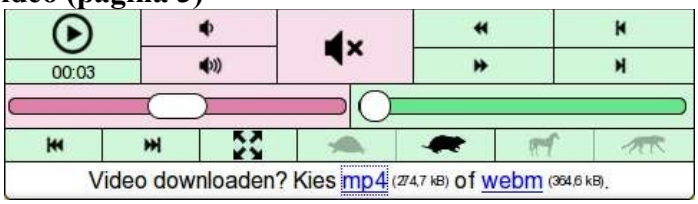


De 's waarin percentage geluidssterkte, totale speelduur, verstreken speelduur en resterende speelduur van de vierde videospeler worden weergegeven. De css is voor alle vier vrijwel hetzelfde, alleen de positie is anders. De meeste css kan hier dus in één keer voor alle vier worden opgegeven.
De z-index is nodig, omdat de volgorde van de elementen op het scherm een andere is dan de volgorde van de elementen in de html. Normaal genomen zouden latere elementen over eerdere heen komen te staan. Door het gebruik van de z-index wordt dat voorkomen.

```
#videobox-4 .duration {left: 0;}
#videobox-4 .elapsed {left: 60px;}
#videobox-4 .remaining {left: 121px;}
```

Totale speelduur, verstreken speelduur en resterende speelduur voor de vierde videospeler. De meeste css is gelijk hierboven al opgegeven. Hier wordt alleen `left` aangepast, omdat ze anders alle drie (en ook nog de geluidssterkte) over elkaar heen zouden komen te staan.

Speciaal voor vijfde video (pagina 3)



Vrijwel hetzelfde als de tweede videospeler, alleen wordt het percentage van de geluidssterkte niet op de knop van de sleepbalk voor geluidssterkte weergegeven, en bij de snelheidsregeling zijn radioknoppen en tekst vervangen door achtergrond-afbeeldingen.
De css voor deze videospeler staat dan ook vrijwel allemaal bij Speciaal voor tweede video. Alleen een kleine aanpassing aan de waar de speler in staat en de snelheidsregeling staan hieronder.
Bij deze videospeler wijkt de volgorde van de elementen op het scherm af van de voor deze pagina opgegeven volgorde van de Tabindex. Daarom wordt onder aan de html de tabindex voor deze videospeler aangepast. Meer daarover bij Het JavaScript onderaan de html-bestanden.

```
#videobox-5 {clear: both;}
```

Het element met id="videobox-5". De waar de vijfde videospeler in staat. De css voor deze is hetzelfde als de algemene voor deze pagina, met de hierboven staande kleine toevoeging.
Omdat de vierde videospeler iets lager is, wordt in sommige browsers deze vijfde speler niet links, maar rechts onder de vierde videospeler gezet. Dat het niet in alle browsers gebeurt, heeft waarschijnlijk met afronden te maken.
Op zich is dit correct gedrag, want zo hoort het te gaan bij gefloate elementen. Maar hier zit het even in de weg. Bovenstaande regel lost het op: de vijfde speler wordt nu altijd links op een nieuwe regel gezet,

`#speed-3 input, #speed-5 input {position: absolute; left: -20000px;}`

Om css uit te sparen is deze regel feitelijk al opgegeven bij #speed-3 input, #speed-5 input. De radioknoppen worden links buiten het scherm geparkeerd, waardoor ze nog wel werken, maar je ze niet meer ziet.

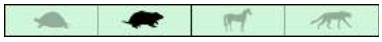
`#speed-5 {background: #d1f7db}`

Voor dit element geldt ook de eerder bij #image-2 .speed, #image-5 .speed opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”speed-5”. De waar de snelheidsregeling voor de vijfde videospeler in staat.
De snelheidsregeling van de tweede videospeler heeft weliswaar een groene achtergrondkleur, maar die is aan de <label>’s van de snelheidsregeling gegeven, niet aan de waar de snelheidsregeling in staat. Hier krijgt de hel een groene achtergrondkleur.

`#speed-5 label {color: transparent; opacity: 0.3;}`

Voor deze elementen geldt ook de eerder bij #speed-2 label, #speed-5 label opgegeven css, voor zover die hier niet wordt veranderd.
De <label>’s binnen het element met id=”speed-5”. De <label>’s bij de radioknoppen voor de snelheidsregeling in de vijfde videospeler.
Het script zet tekst op de <label>’s voor snelheidsregeling. Standaard is die tekst 0,5x, 1x, 1,5x en 2x (Hoe je die tekst eventueel kunt wijzigen, staat bij Text). Hier gebruik ik achtergrond-afbeelding om de snelheid aan te geven. Door simpel de voorgrondkleur doorzichtig te maken, zie je de tekst niet meer. Je kunt de tekst ook helemaal verwijderen bovenin het script, maar dat verwijdert de tekst dan op de hele pagina. Hoe je die tekst kunt verwijderen, staat ook bij Text.
Met `opacity: 0.3;` wordt de hele <label> behoorlijk doorzichtig gemaakt, inclusief achtergrond-afbeelding. Door alleen de <label> van de gekozen snelheid ondoorzichtig te maken, is te zien, welke snelheid is gekozen.

`#speed-5 input:checked + label {opacity: 1;}`



Als een <input> in het element met id=”speed-5” is aangevinkt, doe dan iets met de direct daaropvolgende <label>. In dit geval: maak het label ondoorzichtig, waardoor de erin zittende achtergrond-afbeelding eruit springt. Op de afbeelding is gekozen voor de normale snelheid. De daarbij horende muis is ondoorzichtig (Ik ben geen bioloog en garandeer niet dat de juiste dieren bij de juiste snelheid horen. Volgens mij is een jachtluipaard sneller dan een paard, maar ik doe er geen moord voor. Hmmm, áls het ’n jachtluipaard is... En ís dat wel ’n muis?)

`#speed-5 input:focus + label {background-color: white;}`

Bij Focus is voor de videospelers op deze pagina opgegeven, hoe focus herkend kan worden bij de bedieningselementen. Maar dat is daar niet geregeld voor de snelheidsregeling, omdat die bij de meeste spelers ontbreekt. Daarom wordt dat hier alsnog gedaan voor deze speler.

`#speed-5 input:focus`: als een <input> binnen het element met id=”speed-5” focus heeft. Als een radioknop voor de snelheidsregeling focus heeft.

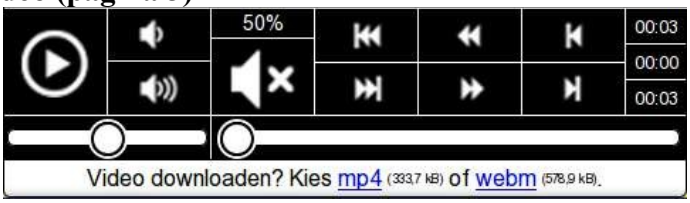
`+ label`: doe dan iets met de gelijk op die <input> volgende <label>. Het script voegt gelijk na de <input> de bijbehorende <label> in, dus dit is altijd de juiste <label>.

Als een <button> focus heeft, maak dan de achtergrond van de gelijk daarop volgende <label> wit.

```
#speed-5 .speed-half-label {background: url(../103-images/buttons-
     background-page-3-klein.png) -1046px no-repeat #d1f7db;}
#speed-5 .speed-default-label {background: url(../103-
     images/buttons-background-page-3-klein.png) -1124px no-repeat
     #d1f7db; left: 69px;}
#speed-5 .speed-one-half-label {background: url(../103-
     images/buttons-background-page-3-klein.png) -1206px no-repeat
     #d1f7db; left: 138px;}
#speed-5 .speed-double-label {background: url(../103-
     images/buttons-background-page-3-klein.png) -1288px no-repeat
     #d1f7db; width: 58px; left: 207px;}
```

Voor deze vier elementen geldt ook de even hierboven bij `#speed-5 label` opgegeven css, voor zover die hier niet wordt veranderd.
Alle vier de elementen liggen binnen het element met id=”speed-5”, de met de snelheidsregeling voor de vijfde videospeler. Het zijn de <label>’s die horen bij de radioknoppen voor halve, normale, anderhalve en dubbele snelheid.
Voor het symbool wordt een achtergrond-afbeelding gebruikt, zoals beschreven bij Symbolen voor bedieningselementen. Hier wordt die achtergrond-afbeelding met de bijbehorende achtergrondpositie opgegeven, en worden de elementen met behulp van `left` op de juiste plaats gezet. Van de laatste <label> wordt de breedte aangepast, zodat de vier <label>’s naast elkaar passen.

Speciaal voor zesde video (pagina 3)



De zesde videospeler is precies hetzelfde als de eerste, alleen zijn de kleuren anders, ook bij focus. Daarom wordt hieronder alleen de css voor de knoppen van de sleepbalken beschreven, want dat zijn de enige elementen die echt ’n beetje anders zijn. Bij de andere bedieningselementen zijn alleen dingen als `color`, `background` (achtergrond-afbeelding, achtergrondpositie, achtergrondkleur) en `border-color` anders dan bij de eerste videospeler. En ’n enkele heel kleine correctie om dat bij omgekeerde kleuren het oog het soms net ’n pixel anders ziet.
De in deze speler gebruikte achtergrond-afbeelding is ’buttons-background-page-3-invert.png’.

```
#sound-slider-button-6, #image-slider-button-6 {background: white;
     border-color: black; border-width: 2px; border-radius: 12px;
     box-shadow: 0 0 0 2px white;}
```

Voor deze elementen geld ook de eerder bij .sound-slider-button of .image-slider-button opgegeven css, voor zover die hier niet wordt veranderd.
De sleepknoppen van de sleepbalken in de zesde videospeler De dubbele rand kan heel simpel worden gemaakt. Achtergrond wit. Daaromheen een gewone zwarte border. En dan daaromheen weer een witte `box-shadow`, die gewoon helemaal rond wordt gemaakt, zodat het lijkt op een tweede border.

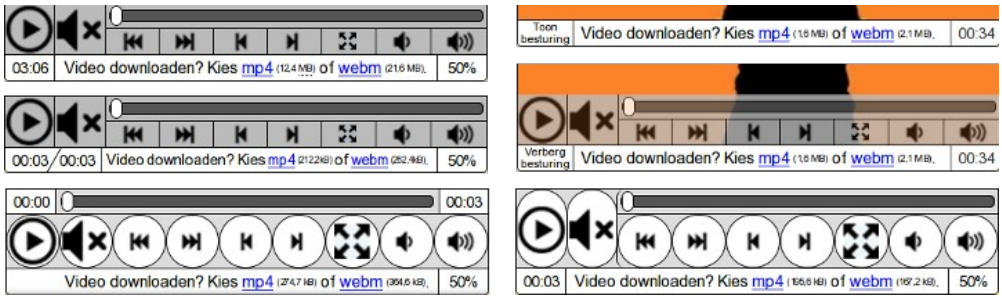
```
#sound-slider-beam-6:focus #sound-slider-button-6, #image-slider-
    beam-6:focus #image-slider-button-6 {border-color: black;
    box-shadow: 0 0 0 2px red;}
```

Uiteraard moeten ook de kleuren bij focus voor de sleepknoppen worden aangepast. (Meer over wat focus is bij Focus.)

Pagina 4

De css die hier wordt beschreven, hoort bij de stylesheet 'afbeelding-103-4-dl.css'. Deze stylesheet hoort bij de vierde pagina met videospelers. Omdat het bij vijf pagina's met grotendeels verschillende videospelers om heel veel css gaat, wordt hier alleen de css besproken die afwijkt van die in 'afbeelding-103-1-dl.css'. De css daarin is te vinden bij Pagina 1.

Omdat het uiterlijk van de zes videospelers op deze pagina nogal verschilt, worden van alle zes hieronder de bedieningselementen neergezet.



Algemeen (pagina 4)

Omdat de volgorde van de elementen in de html anders is dan die op het scherm, heeft deze pagina een aangepaste Tabindex.

Verborgen bedieningselementen (pagina 4)

```
.sound-slider, .image .five-back, .image .five-forward, .speed,
    .duration, .elapsed, .aria-elapsed {display: none;}
```

Op deze pagina worden de sleepbalk voor de geluidssterkte, de knoppen Vijf procent terug en Vijf procent vooruit, de snelheidsregeling, de totale speelduur, de verstreken speelduur en het voorlezen daarvan voor de hele pagina verborgen. Voor aparte spelers kunnen elementen eventueel later weer worden ingeschakeld.

Symbolen voor bedieningselementen (pagina 4)

Op deze pagina worden zogenaamde data-uri's gebruikt voor de symbolen in de videospelers. Een data-uri bestaat uit een hele serie codes, die door de browser weer worden omgezet naar een afbeelding. Het formaat van die codes moet aan een aantal eisen voldoen, zodat de browser weet, wat de bedoeling is.

De background-regel voor de Speel-pauzeerknop ziet er als volgt uit:

```
background:
url(
yCAYAAAAayliMAAAABmJLR0QA/wD/AP+gvaeTAAAACXBIWXMAAAsTAAAL
EwEAmpwYAAAH1UlEQVRo3tWa2W9cZxmHnzGW/xkEi9JXLK2WWiLgRaBa
YEiCmrDIhYhhIRA3CHxr3CPuOQCIbgBVJYIhFRaCCKAgCZpaUlpEpIQO8
k4jh0vmfF4hguej7wczYyXxMh8
```

(…) nog veel meer van dit soort gruwel (...)

```
mXF7y+61O/flGavmUHyb1xITwGc9BMU3r6qFVTZAqhm4vrgr/fst5yYB9
2Uy7sB7rbmTtED5jKpDS/MGA/WL43upfDRqhDqpbwiehT6vEQpg+PFAF4
```

```
mB3WOweMC72yFa7Der0Mjz23Uuy202r0Ot+XhLnUxt9fVW4z6yZoNEjgx
z386CB2hWeUdXaV6xi3zDwK2Gwu+F88i9C7HJhsWHHMAAAAABJRU5ErkJ
ggg==) 0 -1px no-repeat;
```

Je moet even zoeken misschien, maar dit is een doodnormale `background`. Als eerste een `url`, alleen staat tussen de haakjes geen adres, maar data. Na het sluithaakje van de `url` staat een `background-position` (`0 -1px`) en 'n gewone `no-repeat`.
Omdat dit in de css staat, spaart het een aanroep naar de server uit: er hoeft niet 'n aparte afbeelding opgehaald te worden.
Aan het begin, gelijk na `url(` staat ``. Dit vertelt de browsers dat hier data staan, geen eigenschappen, kleuren, waardes, wat dan ook: gewoon rauwe data. (Echt rauw. Lees het maar 'ns voor, er komen alleen maar rauwe geluiden uit je keel.)
`image/png` vertelt dat het om een image, een afbeelding, gaat van het type png. `base64` tenslotte is de gebruikte codering. Deze codering maakt het mogelijk om een afbeelding weer te geven in de vorm van letters, cijfers, enz.
Hier wordt het heel belangrijk dat een `aria-label` wordt gebruikt, omdat een schermlezer anders niet weet, waarvoor een bepaalde knop dient. Aria-labels worden door het script ingevoegd. (Meer over aria-labels bij WAI-ARIA-codes binnen de videospeler.)
Voor elke achtergrond-afbeelding zijn andere data nodig. Nu is de volgende vraag natuurlijk: hoe maak ik zo'n weerzinwekkende reeks data, zonder gillend overspannen te raken? Dat kan online op dopiaza.org, waar de hier gebruikte data-uri's ook zijn gemaakt. Het maken gaat heel simpel: je stopt er 'n mooie afbeelding in en krijgt 'n gruwel terug. Ook de benodigde codes aan het begin `data:image/png;base64` worden door de site aangemaakt. Als je geen png-afbeelding gebruikt, maar bijvoorbeeld jpg, wordt dat mime-type gebruikt.
(Die site doet trouwens wat denken aan het kabinet-Rutte. Wat die site met 'n afbeelding doet, doet Rutte met een heel land doet: je stopt er 'n redelijk fatsoenlijk land in en je krijgt 'n warrige puinhoop terug.)

css voor vensters breder dan 700 px (pagina 4)

```
@media screen and (min-width: 700px) {
	main nav + p {
		-moz-column-count: 2;
		-moz-column-gap: 1em;
		-webkit-column-count: 2;
		-webkit-column-gap: 1em;
		column-count: 2;
		column-gap: 1em;
	}
}
```

In bredere browservensters te lange regels voorkomen door de tekst in kolommen op te delen. Dit is precies hetzelfde als bij css voor vensters breder dan 700 px voor de tweede pagina, waar de beschrijving is te vinden.

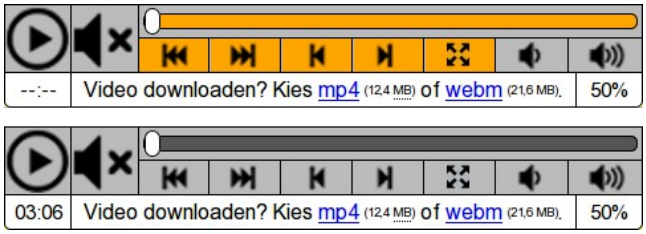
css voor vensters breder dan 1200 px (pagina 4)

```
@media screen and (min-width: 1200px) {
     main nav + p {
          -moz-column-count: 3;
          -webkit-column-count: 3;
          column-count: 3;
     }
}
```

In bredere browservensters te lange regels voorkomen door de tekst in kolommen op te delen. Dit is precies hetzelfde als bij css voor vensters breder dan 1200 px voor de tweede pagina, waar de beschrijving is te vinden.

Bedieningselementen algemeen (pagina 4)

```
.videobox[data-duration="unknown"] .to-begin, .videobox[data-
     duration="unknown"] .ten-back, .videobox[data-
     duration="unknown"] .ten-forward, .videobox[data-
     duration="unknown"] .to-end, .videobox[data-
     duration="unknown"] .fullscreen, .videobox[data-
     duration="unknown"] .image-slider-beam {background-color:
     orange;}
```



Als de speelduur nog onbekend is, wordt de achtergrond van alles dat met afspelen te maken heeft een sfeervol, rustiek, knalhard-oranje. Onder staat dezelfde speler, als de speelduur bekend is.

Als om een of andere reden de speelduur nog niet bekend is, wordt de achtergrond van knoppen en sleepbalk die met afspelen te maken hebben oranje. De werking is precies hetzelfde als bij Bedieningselementen algemeen voor de eerste pagina, alleen worden de knoppen hier niet doorzichtig, maar wordt de achtergrond oranje.

(In sommige spelers wordt een onbekende speelduur op een andere manier aangegeven, hiervoor wordt later aparte css opgegeven.)

```
.controls {background: #bbb; width: 480px; height: 50px; margin: 0
     auto; border: black solid; border-width: 0 1px 1px; position:
     relative;}
```

De elementen met class=”controls”. De ’s waar de bedieningselementen in staan. Het enige verschil met de eerste pagina is de toevoeging van `position: relative;`.Om nakomelingen te kunnen positioneren ten opzichte van een element, moet dat element zelf relatief, fixed of absoluut zijn gepositioneerd. Omdat er verder niets bij wordt ingevuld, heeft dit geen invloed op de zelf.

Speel-pauzeerknop (pagina 4)

```
.play
```

De elementen met class=”play”. De Speel-pauzeerknoppen.

```
     background: url((…)U5ErkJggg==)
          no-repeat 0 -1px;
```

De achtergrond is een data-uri: een in de css opgenomen afbeelding. Deze wordt precies hetzelfde behandeld als een doodgewone achtergrond-afbeelding, alleen staat

er bij `url` geen fatsoenlijk adres, maar hopeloze wartaal. Op de plaats van de haakjes staan nog vele regels abracadabra. Meer uitleg bij Symbolen voor bedieningselementen. Maar feitelijk hoef je alleen maar te weten, dat die wartaal door de browser wordt vertaald naar de afbeelding voor de Speelknop.
Er is 'n kleine correctie van 1 px nodig om de afbeelding precies op de goede plaats te krijgen.

`width: 50px; height: 50px; float: left; border: none; border-right: black solid 1px;`
Gewoon nog wat css om de knop op de juiste plaats te krijgen.

```
.not-playing {background: url ( (…)
U5ErkJggg==) no-repeat 0 -1px;}
```


Behalve dat het hier om een driehoekje gaat, is het verhaal precies hetzelfde als gelijk hierboven bij `.play`. De code die op de plaats van de (…) staat, is ook anders dan bij `.play` hierboven, omdat de afbeelding anders is.
Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Hierdoor kan, als de video niet speelt, een ander symbool worden getoond, dan wanneer de video wel speelt.

Aan-uitknop geluid (pagina 4)

`.sound .mute`


De elementen met class="mute" die binnen een element met class="sound" liggen. De Geluid-aan/uitknop.

`background: url((…)kSuQmCC) no-repeat 1px;`
De achtergrond is een data-uri: een in de css opgenomen afbeelding. Deze wordt precies hetzelfde behandeld als een doodgewone achtergrond-afbeelding, alleen staat er bij `url` geen fatsoenlijk adres, maar hopeloze wartaal. Op de plaats van de haakjes staan nog vele regels abracadabra. Meer uitleg bij Symbolen voor bedieningselementen. Maar feitelijk hoef je alleen maar te weten, dat die wartaal door de browser wordt vertaald naar de afbeelding voor de Speelknop.
Er is een kleine correctie van 1 px nodig om de afbeelding precies op de goede plaats te krijgen.

`width: 50px; height: 50px; float: left; border: none;`
Gewoon nog wat css om de knop op de juiste plaats te krijgen.

```
.mute.muted {background: url((…)
rkJggg==) no-repeat 1px;}
```


Behalve dat hier het kruisje mist, is het verhaal precies hetzelfde als iets hierboven bij `.play`. De code die op de plaats van de (…) staat, is ook anders dan bij `.sound mute` hierboven, omdat de afbeelding anders is.
Aan de Geluid aan-uitknop wordt door het script een class="muted" toegevoegd, als het geluid uitstaat. Hierdoor kan, als het geluid uit staat, een ander symbool worden getoond, dan wanneer het geluid aanstaat.

Knoppen voor afspelen, Harder en Zachter (pagina 4)

```
.image {width: 379px; height: 50px; border-left: black solid 1px;
    position: absolute; top: 0; right: 0;}
```

De elementen met class=”image”. Dit zijn de ’s waarbinnen de knoppen en de sleepbalk voor afspelen staan (behalve de Speel-pauzeerknop). In elke videospeler is er één zo’n element aanwezig.
Gewoon wat css voor de juiste maat, om het goed te positioneren, e.d.

```
.image button, .sound .softer, .sound .louder {width: 53px;
    height: 25px; border: none; border-right: black solid 1px;
    position: absolute; bottom: 0;}
```

`.image button`: de <button>’s binnen de elementen met class=”image”: de knoppen om het afspelen van de video te bedienen.
`.sound .softer, .sound .louder`: de elementen met class=”softer” en class=”louder” binnen een element met class=”sound”. De knoppen Harder en Zachter.
Dit zijn wat basisinstellingen die voor alle knoppen hetzelfde zijn. Waar nodig worden ze later voor aparte spelers weer aangepast.

```
.image .to-begin {background: url(
    (…) VORK5CYII=) no-repeat 50%;}
.image .to-end {background: url( (…)
    RK5CYII=) no-repeat 14px; left: 54px;}
.image .ten-back {background: url( (…)
    K5CYII=) no-repeat 50%; left: 108px;}
.image .ten-forward {background: url(
    (…) TkSuQmCC) no-repeat 16px; left: 162px;}
```

 De knoppen Naar begin, Naar eind, Tien seconden terug en Tien seconden verder. Voor deze knoppen geldt ook de iets hierboven bij `.image button, .sound .softer, .sound .louder` opgegeven css, voor zover die hier niet wordt veranderd.
De achtergrond van alle vier de knoppen is een data-uri: een in de css opgenomen afbeelding. Deze wordt precies hetzelfde behandeld als een doodgewone achtergrond-afbeelding, alleen staat er bij `url` geen fatsoenlijk adres, maar hopeloze wartaal. Op de plaats van de haakjes staan nog vele regels abracadabra. Meer uitleg bij Symbolen voor bedieningselementen. Maar feitelijk hoef je alleen maar te weten, dat die wartaal door de browser wordt vertaald naar de afbeelding voor de Speelknop.
Omdat de afbeeldingen verschillen, staat op de plaats van de (…) bij elke knop andere wartaal, pardon code.
Verder hebben deze knoppen alleen wat eigen css om de afbeeldingen op de goede plaats te krijgen e.d.

```
.image .fullscreen {background: url(
    (...) kSuQmCC) no-repeat 46% -1px; background-size: contain;
    left: 216px;}
```

 De knop Fullscreen. Voor deze knop geldt precies hetzelfde als voor de knoppen hier gelijk boven, met één klein verschil: `background-size: contain;`.
Deze data-uri is gemaakt van een afbeelding, die te groot was voor de beschikbare ruimte. De afbeelding verkleinen leverde geen mooi resultaat op (ook niet lang genoeg

geprobeerd...). Maar omdat de afbeelding te groot was, is de afbeelding die met behulp van de data-uri wordt gemaakt dat ook. `background-size: contain;` zorgt ervoor dat de achtergrond-afbeelding altijd binnen de beschikbare ruimte past.

```
.sound .softer {background: url((…)
    TkSuQmCC) no-repeat 14px; right: 55px; z-index: 100;}
.sound .louder {background: url((…)
    BJRU5ErkJggg==) no-repeat 50%; border: none; right: 0; z-
    index: 100;}
```

 De knoppen voor Zachter en Harder. Ook voor deze knoppen geldt precies hetzelfde als iets hierboven bij `.image .to-begin`.
De enige toevoeging hier is `z-index: 100;` Het script voegt de verschillende elementen in een bepaalde volgorde in. De twee knoppen voor Harder en Zachter worden in de html ingevoegd vóór de vijf hierboven staande knoppen, die met het afspelen te maken hebben. Deze vijf knoppen voor het afspelen staan in `div.image`.

 Omdat dat voor het positioneren makkelijker was, loopt die `div.image` over de volle breedte van de bediening. Dus ook over de ruimte, waar de knoppen Harder en Zachter staan. En omdat `div.image` in de html na de knoppen Harder en Zachter staat, komt de boven de knoppen te staan. Waardoor deze niet meer reageren op klikken of aanraken.
Op de afbeelding is de achtergrond van `div.image` even doorzichtig rood gemaakt. De knoppen en de sleepbalk die binnen `div.image` staan, zijn duidelijk zichtbaar. Maar de knoppen Zachter en Harder, die in de html vóór `div.image` staan en dus onder die verdwijnen, zijn slecht zichtbaar. Als de achtergrond van `div.image` ondoorzichtig rood was gemaakt, zou je ze helemaal niet zien.
Door `z-index: 100;` te gebruiken, komen de knoppen Zachter en Harder toch boven `div.image` te staan, ook al komt `div.image` in de html na beide knoppen.

Percentage volume, verstreken, resterende en totale speelduur (pagina 4)

```
.percentage, .remaining, .duration, .elapsed {width: 43px; font-
    size: 15px; line-height: 25px; text-align: center; padding: 0
    3px; position: absolute; bottom: -26px;}
```

De elementen met class=”percentage”, class=”remaining”, class=”duration” en class=”elapsed”. De ’s waarin de geluidssterkte, de resterende speelduur, de totale speelduur en de verstreken speelduur worden weergegeven.
In de meeste videospelers zien deze ’s er hetzelfde uit, daarom wordt hier in één keer de benodigde css opgegeven. Waar nodig wordt dit dan later aangepast.

```
.remaining {border-right: black solid 1px; left: -101px;}
.percentage {border-left: black solid 1px; right: 0;}
```

De meeste css voor deze ’s voor resterende speelduur en geluidssterkte is gelijk hierboven al opgegeven. Met deze aanpassingen komen ze links en rechts van de download-links te staan.

Focus (pagina 4)

`.controls button:focus`

De <button>'s binnen de elementen met class="controls", maar alleen als ze focus hebben. Dit zijn de knoppen van de videospelers. (Meer over wat focus is, staat bij Focus.)

`background-color: white;`

Achtergrondkleur wit maken.

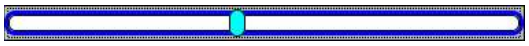
`outline: blue solid 3px;`

Blauwe outline. Je zou ook 'n border kunnen gebruiken, maar bij outline kun je de rand binnen de knop zetten met behulp van het hieronder gebruikte `outline-offset`, en dat kan met een border niet.

`outline-offset: -3px;`

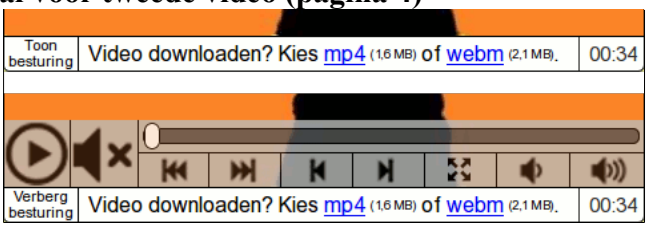
Een outline staat normaal genomen buiten het element, waar de outline bij hoort. Met `outline-offset` kun je de outline naar binnen of naar buiten verplaatsen. De negatieve offset van 3 px zet de outline net binnen het element, waar hij bij hoort. Hierdoor komt de outline aan de bovenkant van knop of sleepbalk niet 'n stukje over de video te staan, maar staat er precies onder.
(Internet Explorer kent `outline-offset` niet, dus daar komt de outline 'n heel klein stukje over de video te staan, maar alleen als de knop of sleepbalk focus heeft.)

```
.controls .image-slider-beam:focus {background-color: white; box-
    shadow: 0 0 0 3px blue;}
.image-slider-beam:focus .image-slider-button {background: #0ff;}
```



Hier gelijk boven is focus voor de <button>'s geregeld, maar de sleepbalken zijn geen <button>'s, maar 's, dus dat moet nog worden gedaan.
De bovenste regel is voor de balk van de sleepbalk. Bij focus witte achtergrond en blauwe box-shadow. Er wordt hier een box-shadow gebruikt en geen outline, omdat een outline geen ronde hoeken krijgt, en een box-shadow wel.
De tweede regel is voor de sleepknop van de sleepbalk: bij focus een lichtblauwe achtergrondkleur geven.

Speciaal voor tweede video (pagina 4)



De bediening van de tweede videospeler ziet er precies hetzelfde uit als de die van de eerste speler. Maar normaal genomen is deze bediening verborgen, zoals op de afbeelding hiernaast bovenin is te zien. Pas na klikken op of aanraken van 'Toon besturing' wordt de bediening zichtbaar. Deze wordt, enigszins doorzichtig, boven de video geplaatst. Bij klikken op of aanraken van 'Verberg besturing' verdwijnt de bediening weer. Sommige mensen gebruiken de Tab-toets om tussen links, knoppen, e.d. te bewegen, omdat ze de muis niet kunnen of willen gebruiken (meer hierover bij Tabindex). Ook met de Tab-toets is de bediening te openen en te sluiten: nadat 'Toon besturing' focus heeft gekregen, is de bediening te openen of te sluiten met de spatiebalk.
Ook is de bediening te openen en te sluiten met sneltoets 7 (afhankelijk van de browser is dat Alt+7, Alt+Shift+7 of Alt+Control+7). Dit werkt hier beter dan Control+→/← of (afhankelijk van de browser) Control+Alt+→/← om naar de vorige of volgende video te gaan, want daarmee ga je naar de Speel-pauzeerknop. En die is hier (nog) niet zichtbaar, dus dat schiet niet op. Met sneltoets 7 ga je in deze speler rechtstreeks naar de knop, waarmee de bediening is te openen of te sluiten.

Om de bediening (on)zichtbaar te maken, is ook in de html een aanpassing nodig. Deze aanpassing staat helemaal los van wat het script doet, hij wordt gewoon in de normale html aangebracht. De vette code hieronder komt alleen bij deze videospeler voor:

```
<div class="videobox">
    <h2 id="titel-2" lang="en">Elliot B senior project
        with after effects</h2>
    <p class="origineel"> (…) Link naar originele video (…) </p>
    <input id="toon-controls" type="checkbox" aria-
        hidden="true" tabindex="191" accesskey="7">
    <div id="wrapper-toon-controls" aria-hidden="true">
        <label id="toon-controls-label" for="toon-
            controls" title="Toon of verberg
            besturing"></label>
    </div>
    <video>
        (…) Alles wat binnen het <video>-element staat (…)
    </video>
    <p class="groot"> (…) Download-links (…) </p>
</div>
```

Alle overbodige code is hierboven weggelaten, alleen wat nodig is voor het tonen en verbergen van de bediening is volledig aanwezig.
Er zijn een extra <input> en een extra aangebracht. Deze staan binnen `div.videobox`. Het maakt niet uit waar ze staan, als het maar vóór `div.controls` met de bediening is.
(`div.controls` is hierboven niet te zien, omdat die door het script wordt ingevoegd. `div.controls` staat gelijk na </video>. In de gegenereerde code is alle code te zien, ook wat het script invoegt.)

`<input id="toon-controls" type="checkbox" aria-hidden="true" tabindex="191" accesskey="7">`

Een gewone checkbox. Alleen als deze checkbox is aangevinkt, is de bediening zichtbaar. Als de checkbox niet is aangevinkt, staat de bediening links buiten het venster van de browser geparkeerd en is daardoor niet zichtbaar. Maar de bediening is er dus wel degelijk, alleen niet zichtbaar. Het vinkje bij de checkbox wordt, wat nou niet echt heel mooi is, wordt verborgen onder de witte achtergrond van de <p> met de download-links.

Omdat de bediening gewoon aanwezig is, en ook gewoon werkt, heeft een schermlezer niets aan deze constructie. Die leest gewoon de normale bediening voor, want waar die op het scherm wordt neergezet, is onbelangrijk voor een schermlezer. Daarom wordt deze hele knop met `aria-hidden="true"` verborgen voor schermlezers. Een overbodige knop zou alleen maar verwarrend werken.

Op de vierde pagina zijn de tabindexen aangepast. Bij gebruik van de Tab-toets moet de checkbox die de bediening toont of verbergt in de juiste volgorde worden aangedaan, vandaar de `tabindex="191"`. De checkbox staat nu, bij gebruik van de Tab-toets, tussen de link naar het origineel boven <video> en de bedieningselementen onder <video>. Als de tabindex niet zou zijn aangepast, zou je

gewoon kunnen volstaan met `tabindex="0"`, wat eigenlijk beter is voor de toegankelijkheid. Meer over het hoe en waarom van tabindex bij Tabindex.

`<div id="wrapper-toon-controls" aria-hidden="true">`

Deze is eigenlijk alleen nodig om de <label> met de tekst 'Toon besturing' (of 'Verberg besturing') op de goede plaats neer te kunnen zetten. Door deze horizontaal halverwege `div.videobox` te zetten, even breed te maken als de video en dan weer de halve breedte naar links terug te zetten, staat de linkerkant van deze altijd gelijk met de linkerkant van de video. En kan de hierin staande <label> dus ook op de goede plaats worden gezet.
Ook hier geldt weer dat een schermlezer niets aan deze constructie heeft, dus wordt de (en daarmee alles wat erin staat ook) verborgen voor schermlezers met `aria-hidden="true"`.
Het zou makkelijker zijn als de <label> gewoon binnen de <p> met de download-links kon worden neergezet, maar dat werkt niet zo goed, omdat er dan problemen met de outline bij focus optreden. Dit lijkt de makkelijkste oplossing te zijn.

```
<label id="toon-controls-label" for="toon-controls"
    title="Toon of verberg besturing"></label>
```

Een gewoon <label> die bij `button#toon-controls` hoort.

`#videobox-2 {position: relative;}`

Het element met id="videobox-2". Dit is de , waar alles van de tweede videospeler in staat. Om nakomelingen van een element te kunnen positioneren ten opzichte van dat element, moet het element een relatieve, absolute of fixed positie hebben. Dat is hierbij geregeld. Omdat er verder niets bij de positie wordt opgegeven, heeft dit verder geen enkele invloed op `#videobox-2` zelf.

`#percentage-2 {display: none;}`

Het element met id="percentage-2". De waarin de geluidssterkte wordt weergegeven. Die wordt in deze videospeler niet gebruikt.

`#toon-controls`

Het element met id="toon-controls". Dit is de <input> type "checkbox" die zorgt voor het tonen of verbergen van de bediening. Alleen als deze checkbox is aangevinkt, wordt de bediening getoond.
Deze checkbox is normaal genomen niet te zien, omdat de witte achtergrond van de download-links hem verbergt. Bovendien staat de bij de checkbox horende <label> er ook nog 'ns overheen. Als de witte achtergrond van de download-links en de <label> even onzichtbaar worden gemaakt, is de checkbox gewoon te zien, zoals op de afbeelding is te zien.
De checkbox wordt verborgen omdat, tja, eh, hoe zeg je dat politiek correct, zonder op uiterlijk te discrimineren en zo: de checkbox ziet er fantastisch mooi uit, maar komt in deze omgeving niet helemaal tot z'n recht. Daarom wordt de lelijkerd weggemoffeld achter de witte achtergrond en de <label>. (Ja zeg, ál te politiek correct is ook saai. En medelijden hoef je niet te hebben: het is de Kai van der Linden van de checkboxen: je ziet hem niet, maar hij regelt stiekem achter de schermen het verschijnen en verdwijnen van de hele bediening.)

`margin-left: -230px; position: absolute; bottom: 0; left: 50%;`

Absoluut positioneren om op de juiste plaats neer te kunnen zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een relatieve, absolute of fixed positie heeft. Dat is hier `#videobox-2`, de waar de tweede videospeler met bijbehorende titels, links, e.d. in staat.
Met `left: 50%;` wordt de checkbox halverwege die gezet. En met `margin-left: -230px;` wordt weer 230 px terug naar links verplaatst. Omdat de video 480 px breed is, staat de checkbox nu redelijk links in de . Op de plaats waar later de <label> overheen wordt gezet.
Met `bottom: 0;` wordt de checkbox onderaan neergezet.

`#videobox-2 #wrapper-toon-controls`

Het element met id=”wrapper-toon-controls” binnen het element met id=”videobox-2”. De binnen de tweede videospeler die nodig is om de <label> met de tekst ’Toon/Verberg besturing’ op de juiste plaats te zetten.

`width: 480px;`

Even breed als <video> maken. Nu hoeft de alleen nog maar aan de linkerkant precies gelijk met <video> (en dus de video zelf) te worden neergezet, om de in de zittende <label> altijd op de juiste plaats neer te kunnen zetten.

`text-align: center;`

Tekst horizontaal centreren.

`margin-left: -240px;`

Hieronder wordt `div#wrapper-toon-controls` halverwege `#videobox-2` neergezet. De is 480 px breed. Als hij 240 px terug naar links wordt gezet, staat de linkerkant dus altijd precies 240 px links van het midden van `#videobox-2`. Dat is ook, waar de linkerkant van de video staat, waardoor de dus altijd links is uitgelijnd met de video, ongeacht de breedte van het venster van de browser. Daarmee is ook de in de zittende <label> altijd op de juiste plaats neer te zetten.

`position: absolute;`

Om de op de juiste plaats neer te kunnen zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `#videobox-2`, de waar de tweede videospeler, met alle bijbehorende titels, links, enz., in zit.

`bottom: 1px;`

Aan de onderkant neerzetten, op gelijke hoogte als de download-links.

`left: 50%;`

Halverwege `#videobox-2` neerzetten. In combinatie met de `margin-left: -240px;` hierboven staat de nu altijd op de juiste plaats. En dus de in de zittende <label> ook.

`#videobox-2 #toon-controls-label`

Het element met id=”toon-controls-label” binnen het element met id=”videobox-2”. De <label> bij de hierboven staande <input> type ”checkbox”. Binnen deze <label> staat de tekst ’Toon besturing’ of ’Verberg besturing’, afhankelijk van of de checkbox is aangevinkt of niet.

`line-height: 12px; height: 25px;`

Met behulp van `::after` wordt hieronder een pseudo-element bij de <label> gemaakt. Daarin komt de tekst te staan, die de <label> moet tonen. Die tekst is ’Toon

\Abesturing' of 'Verberg \Abesturing'. Op het scherm zie je de '\A' niet. Dit is een zogenaamde 'escape code', die aangeeft dat een nieuwe regel begint. De twee woorden komen dus niet naast elkaar, maar onder elkaar te staan.
De totale hoogte is 25 px, en bij een regelhoogte van 12 px staan de twee woorden netjes op hun plaats.

`float: left;`

De <label> staat binnen de hierboven staande `div#wrapper-toon-controls`. De linkerkant van die staat gelijk aan de linkerkant van de video. Door de <label> naar links te floaten, komt die ook op de goede plaats aan de linkerkant te staan.
Een <label> is een inline-element. Door een element te floaten, verandert het in een soort blok-element, waardoor eigenschappen als hoogte en padding zijn te gebruiken.

`font-size: 0.7em; border-right: black solid 1px; padding: 0 3px;`

Hele handel laten passen en opleuken.

`position: relative; z-index: 10;`

De <label> moet links van de download-links komen te staan. In de html komt de <label> vóór de download-links. Daardoor zouden deze links boven de <label> komen te staan, en door de witte achtergrond van de download-links zou je de tekst van de <label> niet meer zien. Bovendien werkt klikken op of aanraken van de <label> niet, omdat de <label> door de download-links wordt afgedekt.
Daarom wordt een z-index aan de <label> gegeven: nu staat de <label> toch boven de download-links en is dus zichtbaar, en aanraken en klikken werken ook. Een z-index werkt alleen bij een element met een absolute, relatieve of fixed positie, vandaar de `position: relative;`. Verder heeft dit geen invloed, omdat voor de positie verder geen waarden worden opgegeven.

`#toon-controls:focus + div #toon-controls-label`

Bij Focus is het uiterlijk van <button>'s met focus geregeld. Maar dit is een <input> type "checkbox", dus dat moet hier alsnog geregeld worden, zodat ook hier dezelfde blauwe outline verschijnt bij focus, als bij alle andere knoppen e.d.

`#toon-controls:focus`: het element met id="toon-controls", als dit focus heeft. Dit is de checkbox die tonen en verbergen van de bediening regelt.

`+ div`: de die gelijk op deze checkbox volgt.

`#toon-controls-label`: het element met id="toon-controls-label" dat binnen deze ligt. Dit is de bij de checkbox horende <label>.

Samengevat: doe iets met de bij de checkbox horende <label>, als de checkbox focus heeft.

`outline: blue solid 3px;`

Drie px brede blauwe outline.

`#toon-controls-label::after`

Met behulp van `::after` wordt bij het element met id="toon-controls-label" een pseudo-element gemaakt, met behulp waarvan tekst kan worden weergegeven. `#toon-controls-label` is de <label> bij de checkbox, waarmee de bediening kan worden getoond of verborgen.

`content: "Toon\A besturing";`

De tekst die moet worden weergegeven. De `\A` is een code voor een nieuwe regel, op het scherm zie je die niet. 'Toon' en 'besturing' komen dus onder elkaar te staan en niet naast elkaar. Hierdoor passen ze binnen de beschikbare ruimte.

`white-space: pre;`

De `\A` voor een nieuwe regel werkt alleen, als dit ook is opgegeven.

`#toon-controls:checked + div #toon-controls-label::after`

`#toon-controls:checked`: als het element met id=”toon-controls” is aangevinkt. Dit is de checkbox die het tonen en verbergen van de bediening regelt.

`+ div`: de die gelijk op deze checkbox volgt.

`#toon-controls-label::after`: maak met behulp van `::after` een pseudo-element, waarin tekst kan worden weergegeven.

Anders dan gelijk hierboven bij `#toon-controls-label::after` gaat het hier dus om tekst, die alleen zichtbaar is als de checkbox is aangevinkt.

`content: "Verberg\A besturing";`

De tekst die moet worden weergegeven. De `\A` is een code voor een nieuwe regel, op het scherm zie je die niet. ’Verberg’ en ’besturing’ komen dus onder elkaar te staan en niet naast elkaar. Hierdoor passen ze binnen de beschikbare ruimte.

`white-space: pre;`

De `\A` voor een nieuwe regel werkt alleen, als dit ook is opgegeven.

`#video-2 ~ .groot`

Het element met class=”groot” dat in de html volgt op het element met id=”video-2”. Anders dan bij een +, hoeft `.groot` niet gelijk op `#video-2` te volgen bij een ~, als het in de html maar volgt op `#video-2`. Bovendien moeten `.groot` en `#video-2` dezelfde ouder hebben (dat is hier `div#videobox-2`).

Dit is de <p>, waarbinnen de download-links staan.

`overflow: hidden;`

Hier gelijk onder bij `#video-2[controls="controls"] ~ .groot::before` wordt met behulp van `::before` een wit blokje over de <label> met ’Toon/Verberg besturing’ gezet. Dat is nodig, als is gekozen voor de ingebouwde videospelers. De tekst ’Toon/Verberg besturing’ moet dan worden verborgen. Dat blokje krijgt een hoogte van 26 px. Als het lager wordt, dekt het de <label> niet goed af. Maar bij 26 px steekt het in sommige browsers net boven `div.groot` uit, zoals op de afbeelding is te zien. De combinatie van de hieronder opgegeven hoogte van 26 px met de hier opgegeven `overflow: hidden;` werkt in alle browsers.

`position: relative;`

Om nakomelingen van een element te kunnen positioneren ten opzichte van een element, moet dat element zelf een relatieve, absolute of fixed positie hebben. Omdat er verder niets wordt opgegeven, heeft dit verder geen enkele invloed op het element zelf.

`#video-2[controls="controls"] ~ .groot::before`

`#video-2[controls="controls"]`: het element met id=”video-2”, maar alleen als dit het attribuut `controls="controls"` heeft. Dit is het <video>-element van de tweede videospeler.

`~ .groot`: de elementen met class=”groot”, die in de html na `#video-2` komen. Anders dan bij een +, hoeft `.groot` niet gelijk op `#video-2` te volgen bij een ~, als het in de html maar volgt op `#video-2`. Bovendien moeten `.groot` en `#video-2` dezelfde ouder hebben (dat is hier `div#videobox-2`).

`::before`: met behulp van `::before` wordt bij `.groot` een pseudo-element gemaakt, waarmee – in dit geval – een rechthoekig wit blokje wordt weergegeven om de tekst 'Toon/Verberg besturing' onder te verbergen.

Als bovenaan de pagina wordt gekozen voor de ingebouwde standaardspeler, voegt het script helemaal geen bedieningselementen in. Maar de tekst 'Toon besturing' wordt door css ingevoegd en is dus nog steeds zichtbaar. Wat in dit geval wat eigenaardig is, want er ís helemaal geen besturing om te tonen. Daarom moet die tekst in dit geval worden verborgen. De tekst 'Toon besturing' wordt met behulp van een door `::after` bij de <label> aangebracht pseudo-element getoond. De tekst is dus met behulp van css gemaakt, en moet ook met behulp van css weer worden verwijderd. Het zou vrij makkelijk zijn om het script deze tekst te laten verwijderen, als wordt gekozen voor de ingebouwde spelers, maar dan ga je het script aanpassen voor één videospeler. En de bedoeling was nou juist dat het hele uiterlijk met behulp van css kon worden aangepast, helemaal los van het script.

Als de ingebouwde standaardvideospeler moet worden gebruikt, geef je dat aan door het attribuut `controls` aan <video> toe te voegen: `<video controls>`. In dit geval wordt dit door het script ingevoegd, als wordt gekozen voor de standaardvideospeler. Wat wordt ingevoegd is niet alleen `controls`, maar het iets langere `controls="controls"`. Dat is alleen een verschil in woorden, het werkt hetzelfde. (Er zit 'n heel verhaal achter over html en xhtml, grote ruzies, multinationals en Tante Truus en Ome Piet driehoog achter die haast geen sites meer konden maken, maar dat sla ik allemaal maar even over.)
Goed. Als bij <video> het attribuut `controls="controls"` aanwezig is, dan is dus gekozen voor de ingebouwde standaardspeler. En moet de tekst 'Toon besturing' worden afgedekt met 'n wit vlakje.
Dat afdekken zou het makkelijkst kunnen met een met behulp van `::after` of `::before` bij <video> aangemaakt pseudo-element, maar dat kan niet. <video> is een zogenaamd 'replaced element' (het wordt door 'n video vervangen), en daarbij werken `::after` en `::before` niet. Vandaar de constructie om het via `.groot` te doen. Daar werkt `::before` wel, want `.groot` is 'n doodgewone <p>.
En dan nu eindelijk het inmiddels fameuze witte blokje:

`background: white;`
: Witte achtergrond.

`content: "";`
: Geen inhoud. In dat geval geef je als inhoud `""`: dubbele aanhalingstekens met niets ertussen.

`width: 55px; height: 26px;`
: Met deze breedte en hoogte wordt de tekst in de <label> volledig afgedekt.

`position: absolute;`
: Om op de goede plaats neer te kunnen zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder, die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `.groot`, het element waar deze `::before` bij hoort.

`bottom: 0; left: 3px;`
: Op deze positie wordt de tekst van de <label> volledig afgedekt. Niet helemaal links neerzetten, omdat dan een stukje van de ronde hoek linksonder van `.groot` zou wegvallen onder het witte vlakje.

`z-index: 50;`
: Dit pseudo-element moet de tekst in <label> verbergen. De <label> heeft bij #videobox-2 #toon-controls-label zelf een `z-index=10;` gekregen. Daarom moet

dit pseudo-element een hogere z-index krijgen, anders zou het witte vlakje onder de tekst van de <label> staan en dus volmaakt nutteloos zijn.

`#controls-2`

Voor dit element geldt ook de bij .controls opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”controls-2”. De waarin de bedieningselementen voor de tweede videospeler zitten.

`-webkit-animation: bugfix infinite 1s;`

In oudere versies van Android browser zit een bug, waardoor de bedieningselementen nooit worden geopend. Met behulp van deze regel worden ze toch geopend. Meer hierover bij @-webkit-keyframes bugfix.

`margin-left: -20000px;`

De reden van deze grote negatieve marge staat iets hieronder bij `left: 50%;`.

`opacity: 0.8;`

Klein beetje doorzichtig maken, zodat de video er iets doorheen schijnt als de besturing zichtbaar is.

`position: absolute;`

Om de op de goede plaats te kunnen zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `div#videobox-2`, de waar de hele tweede videospeler met bijbehorende titels, links, e.d. in staat.

`left: 50%;`

Halverwege `div#videobox-2` neerzetten. De reden hiervan is ’n wat ingewikkeld verhaal, vrees ik.
De video staat altijd in het midden van `div#videbox-2`. Als ik de linkerkant van `#controls-2` dus halverwege `div#videobox-2` zet, staat de linkerkant altijd exact in het midden van de video (en de daarbij horende links, titels, e.d.). Ongeacht de breedte van het browservenster.
Bij .controls heeft `#controls-2` een breedte van 480 px gekregen. Oftewel: 240 px van deze staat links van het midden van de video. Hierboven wordt met `margin-left: -20000px;` de hele nog 20000 px naar links verplaatst. Het linkerpunt van de staat nu dus 20000 + 240 = 20240 px links van het midden van de video.
Oftewel: ik heb links buiten het scherm een punt gecreëerd met een bekende afstand tot het midden van de video. Hierdoor weet ik, op welke afstand van links ik andere elementen binnen `div#controls-2` neer moet zetten, om die op de juiste plaats onder de video te krijgen, ook als `#controls-2` zelf niet zichtbaar is, omdat deze links buiten het scherm is geparkeerd.
Dit is van belang, omdat rechts van de download-links de resterende tijd altijd zichtbaar moet zijn, ook als de bediening is verborgen. Omdat ik nu ’n aanknopingspunt heb, kan ik later bepalen hoever de met de resterende tijd naar rechts moet worden verplaatste, zodat deze op de juiste plaats onder de video komt te staan.
Dit is ook ’n reden waarom `display: none;` of `visibility: hidden;` hier niet bruikbaar zijn: ook de met de resterende tijd zou dan onzichtbaar zijn. Bovendien zouden `display: none;` en `visibility: hidden;` de bediening ook verbergen voor schermlezers. Nu is de bediening voor schermlezers gewoon aanwezig, want dat die links buiten het scherm staat, maakt voor een schermlezer niets uit.

`z-index: 10;`

De met de resterende tijd moet altijd zichtbaar zijn. Deze wordt rechts van de download-links gezet, die in `.groot` zitten. In de html komt `.groot` na `#controls-2`, waardoor `.groot` met z'n witte achtergrond over de met de resterende tijd zou komen te staan. Deze zou daardoor niet meer zichtbaar zijn. Door de hele `div#controls-2` een iets hogere z-index te geven, komt deze toch boven `.groot` te staan, en dus de in `#controls-2` zittende met de resterende tijd ook.

`#toon-controls:checked ~ #controls-2`

Voor dit element geldt ook de gelijk hierboven bij #controls-2 opgegeven css, voor zover die hier niet wordt veranderd.

`#toon-controls:checked`: het element met id="toon-controls" als dit is aangevinkt.

Dit is de checkbox die het tonen en verbergen van de bedieningselementen regelt.

`~ #controls-2`: het element met id="controls-2", dat in de html na `#video-2` komt.

Anders dan bij een +, hoeft `#controls-2` niet gelijk op `#toon-controls` te volgen bij een ~, als het in de html maar volgt op `#toon-controls`. Bovendien moeten `#controls-2` en `#toon-controls` dezelfde ouder hebben (dat is hier `div#videobox-2`).

`position: relative;`

Hier gelijk boven bij #controls-2 is een absolute positie opgegeven. Die wordt hier veranderd in een relatieve. Hierdoor kan de met de bedieningselementen makkelijk horizontaal onder de video worden gecentreerd door het gebruik van een margin. Dat kan niet als er absoluut is gepositioneerd.

`margin: -51px auto 0;`

Omdat voor links geen waarde is ingevuld, krijgt links dezelfde waarde als rechts. Hier staat dus eigenlijk `margin: -51px auto 0 auto;` in de volgorde boven – rechts – onder – links.

Door de 51 px naar boven te verplaatsen, komt deze net boven de video te staan.

`auto` betekent hier: evenveel. Oftewel: de marge links en rechts is even groot. De staat dus altijd in het midden van `div#videobox-2`, en daarmee ook midden onder de video. De gelijk hierboven bij #controls-2 opgegeven onwijs grote negatieve linkermarge heeft hier geen nut meer.

Aan de onderkant geen marge.

`left: 0;`

Hier gelijk boven bij #controls-2 is `left: 50%;` opgegeven. Dat werkt hier storend op het horizontaal centreren van de , dus weg ermee.

`#toon-controls ~ #controls-2 .remaining`

Voor dit element geldt ook de bij .percentage, .remaining, .duration, .elapsed en .remaining opgegeven css, voor zover die hier niet wordt veranderd.

De elementen met class="remaining" binnen het element met id="controls-2".

`#controls-2` moet in de html na `#toon-controls` komen. Anders dan bij een +, hoeft `#controls-2` niet gelijk op `#toon-controls` te volgen bij een ~, als het in de html maar volgt op `#toon-controls`. Bovendien moeten `#controls-2` en `#toon-controls` dezelfde ouder hebben (dat is hier `div#videobox-2`).

`#toon-controls` is de checkbox die het tonen en verbergen van de bedieningselementen regelt, `#controls-2` is de waarin de bedieningselementen van de tweede videospeler staan.
Dit is de waarin de resterende tijd staat.

`height: 25px;`

Hoogte.

`margin-left: 20190px;`

Pardon? Is dit tijdreizen of zo?
Bij #controls-2 is `div#controls-2` links buiten het browservenster geparkeerd met een grote negatieve linkermarge: `margin-left: -20000px;`. Hiermee zijn de bedieningselementen verborgen. Maar deze met de resterende tijd moet altijd zichtbaar zijn, ook als de bediening is verborgen.
Door deze weer 'n onwijs grote linkermarge te geven, komt de weer binnen het scherm te staan, terwijl de rest van de bediening nog links buiten het scherm staat te hangjongeren.

`border: none; border-left: black solid 1px;`

Lijntjes op de juiste manier aanbrengen.

`top: 0;`

En op de juiste plaats zetten.

`#toon-controls:checked ~ #controls-2 .remaining`

Voor dit element geldt ook de gelijk hierboven bij `#toon-controls ~ #controls-2 .remaining` opgegeven css, voor zover die hier niet wordt veranderd.
De uitleg van de selector staat gelijk hierboven, alleen moet de checkbox hier zijn aangevinkt.

`margin-left: 0;`

Hier gelijk boven is een grote linkermarge opgegeven, zodat de met de resterende tijd altijd te zien is. Dat is nodig als `div#controls-2` links buiten het scherm is geplaatst. Maar als de checkbox is aangevinkt, is `div#controls-2` met de bedieningselementen gewoon zichtbaar op het scherm, dus is deze marge niet meer nodig.

`top: 51px; right: -1px;`

Op de juiste plaats neerzetten.

`left: auto;`

Bij .remaining is `left: -101px;` opgegeven. Dat wordt hier veranderd naar de standaardinstelling `auto`, omdat anders de `right` gelijk hierboven niet werkt.

Speciaal voor derde video (pagina 4)

Deze videospeler is precies hetzelfde als de eerste, alleen staan resterende en totale speelduur links van de download-links, gescheiden door een schuin streepje.

`#videobox-3`

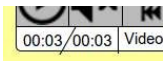
Het element met id="videobox-3". De waar de derde videospeler met bijbehorende titels, links, e.d. in staat.

`clear: both;`

Omdat de tweede video rechtsboven lager is dan de video linksboven, zou deze videospeler niet onder de eerste, maar onder de tweede videospeler worden neergezet. Dat komt doordat alle element met class="videobox" naar links worden

gefloat. Deze regel zorgt ervoor dat ook de derde videospeler op een nieuw regel wordt neergezet, onder de eerste, zoals de bedoeling is.

`overflow: hidden;`



Afhankelijk van de browser steekt het schuine lijntje tussen resterende en totale speelduur iets meer of minder onder de <p> met de download-links uit, zoals op de afbeelding is te zien. De onderkant van die <p> staat precies gelijk met de onderkant van `div#videobox-3`. Door daar de overflow te verbergen, verdwijnt het stukje lijn dat aan de onderkant uitsteekt.

`#videobox-3 .remaining`

Voor dit element geldt ook de eerder bij .percentage, .remaining, .duration, .elapsed en .remaining opgegeven css, voor zover die hier niet wordt veranderd
De elementen met class=”remaining” binnen het element met id=”videobox-3”. Dat is er maar eentje: de met de resterende speelduur.

`border: none;`

De eerder opgegeven border rechts moet weg, want daar komt het schuine lijntje te staan.

`#videobox-3 .remaining::after`

Met behulp van `::after` wordt bij de met de resterende speelduur een pseudo-element gemaakt, waarin het schuine streepje wordt gezet.

`content: "";`

Echte inhoud is er niet, dat geef je aan met twee aanhalingstekens met niets ertussen.

`height: 30px;`

Hoogte. Dit is iets hoger dan de zelf. Dat is nodig omdat het schuine streepje (wat een gewone border is) iets langer moet zijn dan een verticaal streepje zou moeten zijn.

`border-left: black solid 1px;`

Joechei, eindelijk het inmiddels beroemde schuine streepje. Dit is een gewone border, die hieronder iets wordt gedraaid. Een gewone ’/’ werd in elke browser wanstaltig vet, maar een border blijft aanminnig dun.

`position: absolute;`

Om op de goede plaats neer te kunnen zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een relatieve, absolute of fixed positie heeft. Dat is hier `span#remaining-3`.

`top: -2px; right: 0;`

Op de goede plaats neerzetten.

`-ms-transform: rotate(30deg); -webkit-transform: rotate(30deg); transform: rotate(30deg);`

Hier staat in feite drie keer hetzelfde: `transform: rotate(30deg);`.Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Door het hele pseudo-element 30 graden te draaien, draait ook de border links. Wat als resultaat het schuine streepje oplevert.

`#videobox-3 .duration`

Voor dit element geldt ook de eerder bij .percentage, .remaining, .duration, .elapsed opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met class=”duration” binnen het element met id=”videobox-3”. Dat is er maar eentje: de met de totale speelduur.

`display: block;`

Bij Verborgen bedieningselementen is deze verborgen met `display: none;`. Voor deze videospeler wordt de weer zichtbaar gemaakt.

`border-right: black solid 1px;`

Lijntje rechts.

`left: -54px;`

En op de goede plaats zetten.

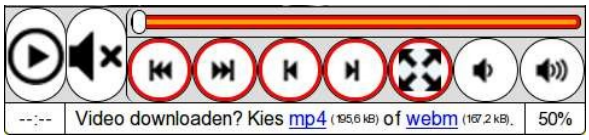
`#videobox-3[data-sound="no"] .duration {left: 0;}`

Op iOS kan de geluidssterkte alleen met de knoppen op het apparaat worden veranderd. Daarom ontbreken daar wat knoppen en moet het uiterlijk worden aangepast, op de manier zoals is beschreven bij .videobox[data-sound="no"] .play.

`#videobox-3 .groot {font-size: 0.9em; word-spacing: -0.1em; max-width: 431px; padding-left: 46px;}`

De elementen met class="groot" binnen het element met id="videobox-3". Dat is er maar eentje: de <p> met de download-links.
Aan de linkerkant moet extra ruimte worden gemaakt voor de met de resterende speelduur. Daar is de padding-left voor. Om dat te compenseren moet de maximale breedte worden verminderd. Met een iets kleinere letter die iets dichter op elkaar staat, kan de tekst voor de download-links dan net in de beschikbare ruimte worden gepropt.

Speciaal voor vierde video (pagina 4)



De vierde videospeler heeft ronde knoppen. Zolang de speelduur onbekend is, krijgen de nog niet bruikbare knoppen een rode outline, zoals op de afbeelding het geval is.
Omdat het uiterlijk van de vierde en vijfde videospeler deels hetzelfde is, staat hier ook al wat css voor de vijfde speler.

`#controls-4, #image-4, #controls-5, #image-5 {height: 78px;}`

Voor deze elementen geldt ook de eerder bij .controls en .image opgegeven css, voor zover die hier niet wordt veranderd
De elementen met id="controls-4", id="image-4", id="controls-5" en id="image-5".
`#controls-4` en `#controls-5` zijn de 's, waarbinnen de bedieningselementen voor de vierde en vijfde videospeler staan. `#image-4` en `#image-5` zijn de 's, waarbinnen de knoppen en de sleepbalk voor het afspelen van de vierde en vijfde videospeler staan (met uitzondering van de Speel-pauzeerknop).
Hoger maken, zodat er ruimte voor de hogere knoppen ontstaat.

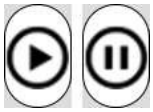
`#controls-4, #controls-5`

Voor deze elementen geldt ook de eerder bij .controls en de gelijk hierboven bij `#controls-4, #image-4, #controls-5, #image-5` opgegeven css, voor zover die hier niet wordt veranderd.
De elementen met id="controls-4" en id="controls-5". De 's waar de bedieningselementen voor de vierde en vijfde videospeler in staan.

`background: #ddd;`

Iets lichtere achtergrond dan eerder is opgegeven.

`#play-4`



Links het afspelen-, rechts het pauze-symbool.

Voor dit element geldt ook de eerder bij .play opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”play-4”. De Speel-pauzeerknop van de vierde videospeler.

`background-color: white;`
Witte achtergrond. Normaal genomen gebruik ik `background` voor alle achtergrond-eigenschappen, maar dat kan hier niet. Eerder is `background` gebruikt om bij .play achtergrond-afbeelding e.d. op te geven. Als ik nu weer background zou gebruiken, maar in dit geval alleen voor de achtergrondkleur met `background: white;`, worden alle andere waarden bij `background` teruggezet naar de standaardwaarden. En verdwijnen dus achtergrond-afbeelding e.d.

`background-position: 0 50%;`
Achtergrond-afbeelding links neerzetten en verticaal in het midden.

`width: 51px; height: 78px;`
Breedte en hoogte van de knop.

`border: black solid 1px; border-radius: 25px;`
Zwarte rand met ronde hoeken.

`#mute-4`

Voor dit element geldt ook de eerder bij .sound .mute opgegeven css, voor zover die hier niet wordt veranderd.
Het element met class=”mute-4”. De Geluid aan-uitknop van de vierde videospeler.

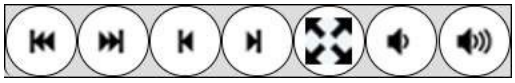
`background-color: white; background-position: 1px 50%; width: 51px; height: 78px; margin-left: -1px; border: black solid 1px; border-radius: 25px;`
De css voor deze knop is vrijwel hetzelfde als die gelijk hierboven voor de Speel-pauzeerknop. Er zijn alleen wat kleine aanpassingen om alles op de goede plaats te krijgen.

`#image-4`

Voor dit element geldt ook de eerder bij .image opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”image-4”. De waar de knoppen en de sleepbalk voor afspelen van de vierde videospeler in staan.

`border-radius: 8px;`
Ronde hoekjes geven. Deze zijn alleen aan de linkerkant zichtbaar, omdat eerder alleen links een border is opgegeven.

`#image-4 button, #softer-4, #louder-4, #play-5, #mute-5, #image-5 button, #softer-5, #louder-5`



Voor deze elementen geldt ook de eerder bij .image button, .sound .softer, .sound .louder opgegeven css, voor zover die hier niet wordt veranderd.
`#image-4 button`: de <button>’s binnen het element met id=”image-4”. Dit zijn de knoppen Naar begin, Naar eind, Tien seconden terug, Tien seconden vooruit en Fullscreen voor de vierde videospeler.

De elementen met id=”softer-4” en id=”louder-4”: de knoppen Zachter en Harder voor de vierde videospeler.

De elementen met id=”play-5” en id=”mute-5”: de Speel-pauzeerknop en Geluid aan-uitknop voor de vijfde videospeler.

`#image-5 button, #softer-5, #louder-5`: zelfde knoppen als hierboven genoemd voor de vierde speler, maar dan voor de vijfde.

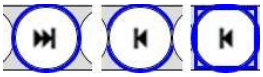
```
background-color: white; width: 54px; height: 53px; border: black solid 1px; border-radius: 26px; position: absolute; bottom: 0;
```

Weinig spannend. Deze knoppen zien er allemaal hetzelfde uit, en krijgen dus allemaal dezelfde css.

```
#videobox-4[data-sound="no"] .to-begin {left: 25px;}
#videobox-4[data-sound="no"] .to-end {left: 107px;}
#videobox-4[data-sound="no"] .ten-back {left: 190px;}
#videobox-4[data-sound="no"] .ten-forward {left: 274px;}
#videobox-4[data-sound="no"] .fullscreen {left: 355px;}
```

Op iOS kan de geluidssterkte alleen met de knoppen op het apparaat worden veranderd. Daarom ontbreken daar wat knoppen en moet het uiterlijk worden aangepast, op de manier zoals is beschreven bij .videobox[data-sound=”no”] .play.

`#controls-4 button:focus, #controls-5 button:focus`



Links zoals het hoort, midden zonder z-index en rechts met outline én box-shadow.

De <button>’s binnen het element met id=”controls-4” of id=”controls-5”, maar alleen als ze focus hebben.

Bij Focus is eerder voor deze pagina opgeven, dat knoppen met focus een blauwe outline moeten krijgen. Maar de knoppen in de vierde en vijfde speler zijn rond, en een outline krijgt geen ronde hoeken. Daarom moet dat hier worden aangepast.

`box-shadow: 0 0 0 3px blue;`

Dit heeft hetzelfde effect als een outline, maar box-shadow krijgt wel ronde hoeken. Omdat alleen de waarde voor de dikte is opgegeven, ziet de ’schaduw’ er in dit geval uit als een blauwe cirkel van 3 px dik.

`outline: none;`

De eerder bij Focus opgegeven outline moet weg, anders hebben we ’n cirkel én ’n vierkant, zoals rechts op de afbeelding is te zien.

`z-index: 50;`

De outline komt buiten de knop te staan. Daardoor komt de outline gedeeltelijk over elementen te staan, die later in de html staan. Daardoor zou de outline verdwijnen onder die elementen, zoals in het midden van de afbeelding is te zien. Een hogere z-index voorkomt dat.

`#play-4:focus, #mute-4:focus`

De Speel-pauzeerknop en de Geluid aan-uitknop van de vierde videospeler, maar alleen als deze focus hebben.

`position: relative;`

Hier gelijk boven is het grootste deel van de focus al afgehandeld. Maar een z-index werkt alleen, als het element relatief, absoluut of fixed is gepositioneerd. En dat zijn deze knoppen niet, dus dat wordt hier geregeld. (De andere knopen hebben eerder bij .image button, .sound .softer, .sound .louder al een absolute positie gekregen.)

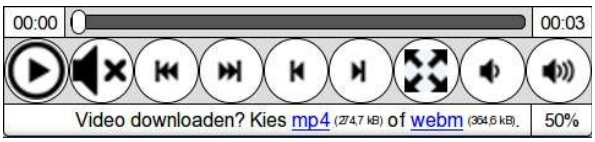
```
#videobox-4[data-duration="unknown"] .to-begin, #videobox-4[data-
    duration="unknown"] .ten-back, #videobox-4[data-
    duration="unknown"] .ten-forward, #videobox-4[data-
    duration="unknown"] .to-end, #videobox-4[data-
    duration="unknown"] .fullscreen, #videobox-4[data-
    duration="unknown"] .image-slider-beam, #videobox-5[data-
    duration="unknown"] .to-begin, #videobox-5[data-
    duration="unknown"] .ten-back, #videobox-5[data-
    duration="unknown"] .ten-forward, #videobox-5[data-
    duration="unknown"] .to-end, #videobox-5[data-
    duration="unknown"] .fullscreen, #videobox-5[data-
    duration="unknown"] .image-slider-beam
```

`#videobox-4` en `#videobox-5` zijn de 's, waarin de vierde en vijfde videospeler staan. `[data-duration="unknown"]` wordt door het script geregeld. Zolang de speelduur onbekend is, is de waarde 'unknown', zodra de speelduur bekend is, verandert dit in 'known'. Hoe dit precies werkt, staat verder bij Bedieningselementen algemeen. Door te testen op de waarde van `data-duration` kan de css worden aangepast, als de speelduur nog onbekend is. In dit geval wordt een rode lijn rondom de zonder bekende speelduur nog niet werkende knoppen en sleepbalk gezet.
Deze selectors bestrijken alle knoppen en de sleepbalk voor afspelen.

```
box-shadow: 0 0 0 3px red inset;
```

Omdat alleen de waarde voor de dikte is opgegeven, ziet de 'schaduw' er in dit geval uit als een rode cirkel van 3 px dik. `inset` zorgt ervoor, dat de cirkel binnen de knop komt te staan, zodat de cirkels van naast elkaar staande knoppen elkaar niet overlappen.

Speciaal voor vijfde video (pagina 4)



De vijfde videospeler is grotendeels hetzelfde als de vierde. De Speel-pauzeerknop en de Geluid aan-uitknop zijn wat lager, en de sleepbalk voor afspelen is wat ingekort. Hierdoor is er naast de sleepbalk ruimte gekomen voor de verstreken en resterende speelduur.
Omdat deze speler zo op de vierde lijkt, staat veel van de css bij Speciaal voor vierde video. Waar dat van toepassing is in de css hieronder, staat steeds een verwijzing naar de css bij de vierde speler. Los van die verwijzingen hieronder is css voor de vijfde speler te vinden bij #controls-4, #image-4, #controls-5, #image-5 en #controls-4, #controls-5.
Bovenstaande geldt voor `#play-5`, `#mute-5`, `#image-5 button`, `#softer-5`, `#louder-5`, `#controls-5 button:focus`, `#play-5:focus`, `#mute-5-focus` en selectors die beginnen met `#videobox-5[data-duration="unknown"]`, gevolgd door een knop of sleepbalk.

```
#play-5, #mute-5, #image-5 button, #softer-5, #louder-5
```

Voor deze elementen geldt ook de eerder bij #image-4 button, #softer-4, ... opgegeven css, voor zover die hier niet wordt veranderd.
De Speel-pauzeerknop, Geluid aan-uitknop, knoppen voor afspelen, Zachter en Harder in de vijfde videospeler.

`width: 53px;`

1 px breder maken dan eerder is opgegeven, zodat het goed past.

`#play-5 {background-position: 1px -1px;}`

Voor dit element geldt ook de gelijk hierboven opgegeven css, voor zover die hier niet wordt veranderd.
De Speel-pauzeerknop van de vijfde videospeler. Kleine correctie van de positie van de achtergrond-afbeelding.

`#mute-5`

Voor dit element geldt ook de iets hierboven bij `#play-5, #mute-5, ...` opgegeven css, voor zover die hier niet wordt veranderd.
De Geluid aan-uitknop van de vijfde videospeler.
`background-position: 1px -1px;`

Kleine correctie van de positie van de achtergrond-afbeelding.
`left: 53px;`

Op de juiste plaats neerzetten.

`#image-5 {border: none;}`

Voor dit element geldt ook de eerder bij #controls-4, #image-4, #controls-5, #image-5 opgegeven css, voor zover die hier niet wordt veranderd.
De waarbinnen de knoppen en de sleepbalk voor afspelen van de vijfde videospeler staan. De eerder opgeven border staat hier als ’n soort eenzame vogelverschrikker midden tussen de knoppen. Weg ermee.

`#image-slider-5`

Het element met id=”image-slider-5”. De waarbinnen de sleepbalk voor afspelen van de vijfde videospeler staat.
`width: 480px;`

Even breed maken als de video. In de algemene css voor deze pagina zijn deze sleepbalken naar rechts gefloat. Door de sleepbalk even breed te maken als de video, vult de sleepbalk nu altijd precies de ruimte onder de video.
Deze is niet de balk van de sleepbalk, maar de waar de hele sleepbalk in staat. Hieronder wordt de eigenlijke sleepbalk een veel kleinere breedte gegeven, waardoor links en rechts van de balk ruimte voor de verstreken en resterende speelduur ontstaat.
`position: relative;`

Om nakomelingen van een element te kunnen positioneren ten opzichte van dat element, moet het element een relatieve, absolute of fixed positie hebben. Omdat er verder niets wordt opgegeven, heeft dit geen enkele invloed op het element zelf.

`#image-slider-beam-5`

Het element met id=”image-slider-beam-5”. De balk van de sleepbalk voor afspelen van de vijfde videospeler.
`width: 370px;`

Deze balk staat in `div#image-slider-5`, die gelijk hierboven 480 px breed is gemaakt. Door deze balk korter te maken, komt er links en rechts van de balk ruimte om de verstreken en resterende speelduur neer te zetten.

`left: 54px;`

In de algemene css voor deze pagina zijn deze balken absoluut gepositioneerd. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een relatieve, absolute of fixed positie heeft. Dat is hier `div#image-slider-5`. Nu is er links van de balk ruimte voor de verstreken speelduur, en rechts voor de resterende speelduur.

`#videobox-5[data-duration="unknown"] .image-slider-beam`

Zolang de speelduur onbekend is, is de waarde van `data-duration` in `div.videobox` 'unknown'. Zodra de speelduur bekend is, verandert het script dit in 'known'. Dit geeft de mogelijkheid met behulp van css het uiterlijk aan te passen, als de speelduur nog onbekend is. Meer over hoe dit precies werkt bij Bedieningselementen algemeen.

`background-color: white;`

Eerder is bij Bedieningselementen algemeen voor deze pagina een oranje achtergrond aan de sleepbalk gegeven, als de speelduur nog onbekend is. Hier wil ik 'n witte achtergrond.

```
#to-begin-5 {left: 5px;}
#to-end-5 {left: 58px;}
#ten-back-5 {left: 111px;}
#ten-forward-5 {left: 164px;}
#fullscreen-5 {left: 217px;}
```

Voor deze elementen geldt ook de eerder bij #image-4 button, #softer-4, ... opgegeven css, voor zover die hier niet wordt veranderd. Afhankelijk van de knop geldt ook nog de eerder bij .image .to-begin, .image .to-end, .image .ten-back, .image .ten-forward en .image .fullscreen opgegeven css, voor zover die hier niet wordt veranderd.
De knoppen Naar begin, Naar eind, Tien seconden vooruit, Tien seconden terug en Fullscreen van de vijfde videospeler. Op de juiste plaats neerzetten.

`#image-5 .fullscreen, #louder-5, #softer-5 {width: 54px;}`

Voor `#image-5 .fullscreen` geldt ook de gelijk hierboven bij `#fullscreen-5` opgegeven css, voor zover die hier niet wordt veranderd. Voor `#louder-5` en `#softer-5` geldt ook de bij #image-4 button, #softer-4, ... en .sound .louder respectievelijk .sound .softer opgegeven css, voor zover die hier niet wordt veranderd.
De knoppen Fullscreen, Harden en Zachter van de vijfde videospeler worden 1 px breder gemaakt om alles passend te krijgen.

`#softer-5 {right: 54px;}`

Voor dit element geldt ook de gelijk hierboven bij `#image-5 .fullscreen, #louder-5, #softer-5` opgegeven css, voor zover die hier niet wordt veranderd.
De knop Zachter voor de vijfde videospeler. Nog 'n kleine correctie om alles passend te krijgen.

`#remaining-5`

De sleepbalk met links de verstreken, rechts de resterende speelduur.

Voor dit element geldt ook de eerder bij .percentage, .remaining, .duration, .elapsed en .remaining opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”remaining-5”. De met de resterende speelduur voor de vijfde videospeler. Op de afbeelding staat deze rechts van de sleepbalk.

```
background: white; width: 42px; height: 24px; border: none;
    border-left: black solid 1px; top: 0; right: 0; left:
    auto;
```

Alleen wat css voor het uiterlijk en om op de goede plaats te zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `div#image-5`.
Eerder is bij .remaining `left: -101px;` opgegeven. Dat wordt hier veranderd in de standaardinstelling `auto`, omdat `right` anders wordt genegeerd.

`#elapsed-5`

Voor dit element geldt ook de eerder bij .percentage, .remaining, .duration, .elapsed opgegeven css, voor zover die hier niet wordt veranderd.
Het element met id=”elapsed-5”. De met de verstreken speelduur voor de vijfde videospeler. Op de afbeelding iets hierboven staat deze links van de sleepbalk.

```
background: white; display: block; width: 42px; height: 24px;
    border-right: black solid 1px; position: absolute; top:
    0; left: -100px;
```

Min of meer hetzelfde verhaal als gelijk hierboven voor `#remaining-5`, maar deze komt links van de sleepbalk te staan.

Pagina 5

De css die hier wordt beschreven, hoort bij de stylesheet ’afbeelding-103-5-dl.css’. Deze stylesheet hoort bij de vijfde pagina met videospelers. Omdat het bij vijf pagina’s met grotendeels verschillende videospelers om heel veel css gaat, wordt hier alleen de css besproken, die afwijkt van die in ’afbeelding-103-1.css’. De css daarin is te vinden bij Pagina 1.

Het uiterlijk van deze zes spelers verschilt nogal. Een aantal is absoluut meer geschikt voor een rariteitenkabinet, dan voor een fatsoenlijke pagina op een site. Maar het was lollig het te maken, en het geeft weer wat er mogelijk is.

Algemeen (pagina 5)

Omdat de volgorde van de elementen in de html anders is dan die op het scherm, heeft deze pagina een aangepaste Tabindex.

css voor vensters breder dan 700 px (pagina 5)

```
@media screen and (min-width: 700px) {
     main nav + p {
          -moz-column-count: 2;
          -moz-column-gap: 1em;
          -webkit-column-count: 2;
          -webkit-column-gap: 1em;
          column-count: 2;
          column-gap: 1em;
     }
}
```

In bredere browservensters te lange regels voorkomen door de tekst in kolommen op te delen. Dit is precies hetzelfde als bij css voor vensters breder dan 700 px voor de tweede pagina, waar de beschrijving is te vinden.

css voor vensters breder dan 1200 px (pagina 5)

```
@media screen and (min-width: 1200px) {
     main nav + p {
          -moz-column-count: 3;
          -webkit-column-count: 3;
          column-count: 3;
     }
}
```

In bredere browservensters te lange regels voorkomen door de tekst in kolommen op te delen. Dit is precies hetzelfde als bij css voor vensters breder dan 1200 px voor de tweede pagina, waar de beschrijving is te vinden.

Speciaal voor eerste video (pagina 5)

Onder de bediening verschijnt een tekst. De inhoud van die tekst is afhankelijk van de geluidssterkte: bij elke 10% vermindering of vermeerdering van de geluidssterkte, verandert de tekst. Bij 100% verdwijnt de video onder een tekst. Op de afbeelding is de geluidssterkte bovenin 20%, onderin 81%.

```
#sound-1, #sound-3, #sound-4 {width: 144px;}
```

De elementen met id=”sound-1”, id=”sound-3” en id=”sound-4”. De ’s waar de geluidsregeling voor de eerste, derde en vierde videospeler in staat.
In de algemene css voor deze pagina is een breedte van 108 px opgegeven. Omdat de knoppen voor Harder en Zachter in deze drie spelers iets breder zijn, moet ook de waar de geluidsregeling in staat iets breder worden.

```
#softer-1, #louder-1, #softer-3, #louder-3, #softer-4, #louder-4
     {width: 43px;}
```

De elementen met id=”softer-1”, id=”louder-1”, id=”softer-3”, id=”louder-3”, id=”softer-4” en id=”louder-4”. De knoppen Zachter en Harder van de eerste, derde en vierde videospeler.

In de algemene css voor deze pagina is een breedte van 25 px opgegeven. Deze knoppen worden hier iets breder gemaakt.

```
#sound-slider-beam-1, #sound-slider-beam-3, #sound-slider-beam-4
    {width: 140px;}
```

De elementen met id=”sound-slider-beam-1”, id=”sound-slider-beam-3” en id=”sound-slider-beam-4”. De ’s die de balk van de sleepbalk voor geluidssterkte vormen in de eerste, derde en vierde videospeler.
In de algemene css voor deze pagina hebben deze elementen een breedte van 100 px gekregen. In deze drie spelers worden ze 140 px breed gemaakt.

```
#image-1, #image-3, #image-4 {width: 294px;}
```

De elementen met id=”image-1”, id=”image-3” en id=”image-4”. De ’s waar de bedieningselementen voor het afspelen van de eerste, derde en vierde videospeler in staan.
In de algemene css voor deze pagina hebben deze elementen een breedte van 330 px gekregen. In deze drie spelers worden ze 294 px breed gemaakt.

```
#videobox-1[data-sound="no"] .image, #videobox-3[data-
    sound="no"] .image, #videobox-4[data-sound="no"] .image
    {width: 436px;}
```

Op iOS kan de geluidssterkte alleen met de knoppen op het apparaat worden veranderd. Daarom ontbreken daar wat knoppen en moet het uiterlijk worden aangepast, op de manier zoals is beschreven bij .videobox[data-sound=”no”] .play.

```
#speed-1 {display: none;}
```

Het element met id=”speed-1”. De met de snelheidsregeling in de eerste videospeler. Verbergen.

```
#duration-1, #remaining-1, #duration-3, #remaining-3, #duration-4,
    #remaining-4 {right: 0;}
```

De elementen met id=”duration-1”, id=”remaining-1”, id=”duration-3”, id=”remaining-3”, id=”duration-4” en id=”remaining-4”. De ’s voor weergave van totale en resterende speelduur in de eerste, derde en vierde videospeler.
Helemaal rechts neerzetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier respectievelijk `#image-1`, `#image-3` en `#image-4`.

```
#videobox-1[data-sound="no"] button, #videobox-3[data-sound="no"]
     button, #videobox-4[data-sound="no"] button {width: 57px;}
#videobox-1[data-sound="no"] .play, #videobox-3[data-
     sound="no"] .play, #videobox-4[data-sound="no"] .play {width:
     42px;}
#videobox-1[data-sound="no"] button.to-begin, #videobox-3[data-
     sound="no"] button.to-begin, #videobox-4[data-sound="no"]
     button.to-begin {width: 48px;}
#videobox-1[data-sound="no"] .fullscreen::before, #videobox-
     3[data-sound="no"] .fullscreen::before, #videobox-4[data-
     sound="no"] .fullscreen::before {line-height: 18px;}
#videobox-1[data-sound="no"] .image-slider-beam, #videobox-3[data-
     sound="no"] .image-slider-beam, #videobox-4[data-
     sound="no"] .image-slider-beam {width: 335px;}
#videobox-1[data-sound="no"] .remaining, #videobox-3[data-
     sound="no"] .remaining, #videobox-4[data-
     sound="no"] .remaining {width: 46px;}
```

Op iOS kan de geluidssterkte alleen met de knoppen op het apparaat worden veranderd. Daarom ontbreken daar wat knoppen en moet het uiterlijk worden aangepast, op de manier zoals is beschreven bij .videobox[data-sound=”no”] .play.

```
#video-1 + .controls .sound::after
```

Met behulp van `::after` wordt bij de elementen met class=”sound” die binnen een element met class=”controls” liggen een pseudo-element gemaakt. `.controls` moet gelijk op het element met id=”video-1” volgen.
In normale taal: bij de waarin de bediening voor de geluidssterkte zit voor de eerste videospeler een pseudo-element. Dit pseudo-element wordt hier gebruikt om een tekst in weer te geven.
De tekst die wordt weergegeven, is afhankelijk van de geluidssterkte. Die tekst zelf wordt verderop geregeld. Maar het uiterlijk van de tekst is steeds hetzelfde (behalve als de geluidssterkte 100% is), dat kan hier in één keer worden geregeld.

```
background: pink; width: 480px; text-align: center; border:
     black solid 1px; border-radius: 5px;
```

Rose achtergrond, even breed als de video, tekst horizontaal centreren, zwart randje, ronde hoeken.

```
position: absolute; top: 68px; left: -41px;
```

Op de goede plaats zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een relatieve, absolute of fixed positie heeft. Dat is hier `div#controls-1`. Met deze waarden komt de tekst precies onder de download-links te staan.

```
#sound-1 {-webkit-animation: bugfix infinite 1s;}
```

In oudere versies van Android browser zit een bug, waardoor de tekst nooit wordt veranderd. Met behulp van deze regel wordt de tekst toch veranderd. Meer hierover bij @-webkit-keyframes bugfix.

```
#videobox-1[data-volume^="percent-0"] .sound::after {content:
     "Lief dat je om de baby denkt, maar íéts harder mag wel.";}
```

De meeste css hiervoor is al iets hierboven opgegeven bij `#video-1 + .controls .sound::after`. Hier wordt alleen de inhoud van de tekst nog opgegeven.

Bij elke `div#videobox`, de waar de video met bijbehorende titel, links, e.d. in staat, wordt door het script een attribuut `data-volume` ingevoegd. Hierin staat de geluidssterkte in procenten. In bovenstaande regel geeft 'percent-0' aan, dat de geluidssterkte 0 procent is. Het geluid staat dus uit. De geluidssterkte wordt automatisch door het script ingevoegd, zoals beschreven bij data-volume.

`#videobox-1`: het element met id="videobox-1", de waar de eerste video met bijbehorende titel, links, e.d. in staat.

`[data-volume^="percent-0"]`: die moet een attribuut `data-volume` hebben. De waarde van `data-volume` moet beginnen met 'percent-0'. Als hierna alleen het teken = zou staan, moet wat daarna tussen aanhalingstekens staat exact de waarde van `data-volume` zijn. Maar omdat hier `^=` staat, is het in dit geval voldoende als de waarde hiermee begint. In dit geval moet de waarde dus met 'percent-0' beginnen. 'percent-01', 'percent-02', (…), 'percent-09' voldoen ook allemaal aan deze voorwaarde. Deze selector bestrijkt dus de geluidssterkte van 0 t/m 9%.

`[data-volume^="percent-1"]`, zoals even hieronder wordt gebruikt, bestrijkt 'percent-10', 'percent-11', (…) 'percent-19', dus 10 t/m 19%. 'percent-2' bestrijkt 20 t/m 29%, enz.

`.sound ::after`: doe iets met het pseudo-element dat met behulp van `::after` bij `.sound` is gemaakt.

`{content: "Lief dat je om de baby denkt, maar íéts harder mag wel.";}`

Deze tekst wordt de inhoud van het pseudo-element en wordt dus zichtbaar op het scherm.

```
#videobox-1[data-volume^="percent-1"] .sound::after {content:
     "Meid, ben je ziek? Ik hoor je herrie haast niet.";}
#videobox-1[data-volume^="percent-2"] .sound::after {content: "Ja
     hoor, mag best iets harder. Is geen bibliotheek hier.";}
#videobox-1[data-volume^="percent-3"] .sound::after {content:
     "Iets harder graag, want nu hoor ik je meezingen.";}
#videobox-1[data-volume^="percent-4"] .sound::after {content:
     "Iets harder mag, maar alleen als het geen house is.";}
#videobox-1[data-volume^="percent-5"] .sound::after {content:
     "Goed hoor, keurig beschaafd volume.";}
#videobox-1[data-volume^="percent-6"] .sound::after {content:
     "Voor jouw doen staat het niet eens zo hard.";}
#videobox-1[data-volume^="percent-7"] .sound::after {content: "Kan
     het iets zachter? De vissen stoppen hun vinnen in hun oren.";}
#videobox-1[data-volume^="percent-8"] .sound::after {content:
     "Hallo! Ik kan mezelf niet horen denken.";}
#videobox-1[data-volume^="percent-9"] .sound::after {content:
      "Misschien 'ns zoeken naar 'tinnitus'?";}
```

Precies hetzelfde als hierboven bij `#videobox-1[data-volume^="percent-0"] .sound::after`, maar dan voor 10 t/m 19%, 20 t/m 29%, (…) 90 t/m 99%.

`#videobox-1[data-volume="percent-100"] .sound::after`

Voor dit element geldt ook de iets hierboven bij `#videobox-1[data-volume^="percent-0"] .sound::after` opgegeven css, voor zover die hier niet wordt veranderd.
Iets hierboven is een tekst opgeven die moet worden getoond, als het percentage van de geluidssterkte met ’1’ begint: ’Meid, ben je ziek? Ik hoor je herrie haast niet.’.
Bij 100% doet zich nu een probleem voor, want ook dat percentage begint met een ’1’. Daardoor zou je, als je van 99 naar 100% gaat, de tekst te zien krijgen die bij het veel zachtere 10 t/m 19% hoort. Daarom wordt voor 100% een aparte regel gemaakt.
Deze regel is precies hetzelfde als die voor 10 t/m 19%, alleen staat er in plaats van `[data-volume^="percent-1"]` (waarde van `data-volume` moet beginnen met ’1’) `[data-volume="percent-100"]` (waarde van `data-volume` moet exact ’percent-100’ zijn).
Deze selector heeft evenveel specificiteit (’gewicht’) als die hierboven voor 10 t/m 19%. Daardoor overrulet hij de eerdere selector, omdat hij later in de css staat,

`content: "ASO!!! De buren trillen hun bed uit!!! Zet onmiddellijk zachter!!!";`
De tekst die wordt getoond.

`background: red; color: white;`
Rode achtergrond. Witte voorgrondkleur, dus witte tekst.

`height: 360px;`
Even hoog maken als de video met erboven staande titel e.d., zodat de hele video wordt afgedekt.

`font-size: 3.5em;`
Grote letter.

`top: -362px;`
Met deze positie wordt de video volledig afgedekt, maar blijft de bediening vrij.

Speciaal voor tweede video (pagina 5)

De tweede video is speciaal voor mensen die conectar -yoga beoefenen. Beoefenaren van deze vorm van yoga zweven in de lucht, waarbij ze verticaal om hun as draaien. Omdat de video tijdens het afspelen ronddraait, kan tijdens deze oefening ongestoord video worden gekeken. Verder zou ik geen enkele nuttige toepassing hiervoor weten te bedenken, behalve dat het grappig was om uit te proberen. O ja, en voor ruimtevaarders die gewichtloos rondzweven!
Voor het draaien van de video wordt gebruik gemaakt van het door het script bij `div.videobox` ingevoegde data-played. In dat data-attribuut staat de verstreken speeltijd in procenten. Door aan elk percentage een bepaalde draaiing te koppelen, draait de video tijdens het afdraaien rond.
Dit betekent dat je voor elk procent 4 regels code nodig hebt. Een idioterie die alleen wordt begrepen door mensen die denken dat je echt zwevend om je as kunt draaien. Oftewel: dit is dus echt niet bedoeld om serieus toe te passen.

Omdat de tweede video maar 34 seconden lang is, zijn er in totaal maar 35 standen gebruikt van de honderd die er mogelijk zijn. Mede hierdoor draait de video nogal schokkerig. Maar zelfs bij honderd standen zal het nooit echt vloeiend worden.
Met behulp van `@keyframes` zou je de draaiing wel redelijk vloeiend kunnen krijgen, maar Safari op OS X crasht volledig als je de video met behulp van `@keyframes` laat draaien. Dat gebeurt ook als je de hele `div.videobox` met behulp van `@keyframes` laat draaien. En met elke methode die ik kon bedenken, waarbij je de video laat draaien met behulp van `@keyframes`.
Daarom wordt `transform: rotate` gebruikt, waarbij de overgang van de ene stand naar de andere schoksgewijs gebeurt.

`#video-2`

Het element met id=”video-2”. Het <video>-element van de tweede videospeler. Deze id wordt automatisch door het script ingevoegd, zoals beschreven bij Overzicht van door het script ingevoegde bedieningselementen, classes en id’s.
De meeste css is al opgegeven bij de algemene css voor deze pagina.
`position: relative; z-index: 100;`

Omdat deze video tijdens het afspelen ronddraait, komt de video over andere elementen te staan. Als die in de html na <video> komen, kan de video ónder deze elementen komen te staan. Door een z-index aan <video> te geven, komt <video> en dus de daarin zittende video altijd bovenaan te staan.
Een z-index werkt alleen, als het element absoluut, relatief of fixed is gepositioneerd. Vandaar de relatieve positie. Omdat er verder niets bij wordt opgegeven, heeft dit verder geen invloed op <video>.

```
#videobox-2[data-played="percent-01"] video {-ms-transform:
     rotate(3.6deg); -webkit-transform: rotate(3.6deg); transform:
     rotate(3.6deg);}
#videobox-2[data-played="percent-02"] video {-ms-transform:
     rotate(7.2deg); -webkit-transform: rotate(7.2deg); transform:
     rotate(7.2deg);}
#videobox-2[data-played="percent-03"] video {-ms-transform:
     rotate(10.8deg); -webkit-transform: rotate(10.8deg);
     transform: rotate(10.8deg);}
```

(…) nog 94 keer dezelfde vier regels met een steeds hoger percentage en aantal graden (…)

```
#videobox-2[data-played="percent-98"] video {-ms-transform:
     rotate(352.8deg); -webkit-transform: rotate(352.8deg);
     transform: rotate(352.8deg);}
#videobox-2[data-played="percent-99"] video {-ms-transform:
     rotate(356.4deg); -webkit-transform: rotate(356.4deg);
     transform: rotate(356.4deg);}
#videobox-2[data-played="percent-100"] video {-ms-transform:
     rotate(0deg); -webkit-transform: rotate(0deg); transform:
     rotate(0deg);}
```

Hier staat in feite drie keer hetzelfde: `transform: rotate(...deg);` Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

`#videobox-2`: de waarin de tweede videospeler met bijbehorende titel, links, e.d. staat.

`[data-played="percent-..."]`: op de plaats van de puntjes staat een percentage.
De selector is alleen geldig, als het bij `div#videobox-2` door het script ingevoegde data-attribuut `data-played` exact het in de selector zittende percentage heeft.

`video`: de selector is bedoeld voor het in `#videobox-2` zittende <video>-element.

```
transform: rotate(... deg);
```

Op de plaats van de puntjes staat het aantal graden dat er gedraaid moet worden.
In een cirkel zitten 360 graden. Als één procent van de speeltijd is verstreken, moet de draaiing dus 3,6 graden zijn. Dat wordt aangegeven met `(3.6deg)`, want er wordt een punt en geen komma gebruikt voor de decimalen. Bij 2% is het 7.2 deg, enz. (Ik kan een rekenmachientje aanbevelen.)
Het script past het percentage afgespeelde tijd voortdurend aan, daar hoeft verder niet voor gedaan te worden.
Er is in dit geval geen regel voor 0% nodig, omdat dat gewoon de standaardinstelling is: normaal horizontaal. Pas als er 1% is afgespeeld, verandert het aantal graden.

Speciaal voor derde video (pagina 5)

De derde videospeler heeft de snelheidsregeling onder de andere bedieningselementen staan. De bedienings-elementen staan niet onder, maar boven de video.
Bij deze videospeler wijkt de volgorde van de elementen op het scherm af van de voor deze pagina opgegeven volgorde van de Tabindex. Daarom wordt onder aan de html de tabindex voor deze videospeler aangepast. Meer daarover bij Het JavaScript onderaan de html-bestanden.
Het uiterlijk van deze videospeler is grotendeels hetzelfde als dat van de eerste speler. Een aantal aanpassingen voor de breedte is daar al opgegeven:

`#sound-3` krijgt bij #sound-1, #sound-3, #sound-4 een breedte van 144 px.

`#softer-3` en `#louder-3` krijgen bij #softer-1, #louder-1, ... een breedte van 43 px.

`#sound-slider-beam-3` krijgt bij #sound-slider-beam-1, #sound-slider-beam-3, ... een breedte van 140 px.

`#image-3` krijgt bij #image-1, #image-3, #image-4 een breedte van 294 px.

`#duration-3` en `#remaining-3` worden bij #duration-1, #remaining-1, ... met

```
right: 0;
```

rechts neergezet.

```
#video-3:not([controls]) {margin-top: 70px;}
```

Omdat de bediening bij deze videospeler boven de video staat, moet hiervoor aan de bovenkant ruimte worden vrijgemaakt. Door de video aan de bovenkant een marge van 70 px te geven, komt er aan de bovenkant 70 px vrij.

Kiezen voor de standaardspeler zonder `:not([controls])` in de selector levert bovenstaande kier op.

Maar dit mag alleen gebeuren, als is gekozen voor de aangepaste speler. Als is gekozen voor de in de browser ingebouwde standaardspeler, ontstaat anders een nutteloos gat van 70 px tussen video en titel boven de video.
`#video-3`: het element met id=”video-3”. De derde <video>.

`[controls]`: alleen als het attribuut ’controls’ aanwezig is bij `#video-3`. Dit attribuut wordt door het script toegevoegd aan <video>, als wordt gekozen voor de ingebouwde videospeler.

`:not`: maar omdat er in dit geval `:not` voor `[controls]` staat, geldt deze selector juist alleen als er géén attribuut 'controls' aanwezig is. Oftewel: als niet is gekozen voor de ingebouwde standaardspeler.

`#controls-3`

Het element met id="controls-3". De waar de bedieningselementen voor de derde videospeler in staan.
Omdat de bedieningselementen hier boven de video staan, zijn een paar kleine aanpassingen van de algemene css voor deze pagina nodig.

`height: 70px;`

De hoogte wordt verhoogd van 42 px naar 70 px, omdat de snelheidsregeling er ook in moet passen.

`margin-top: -392px;`

392 px naar boven zetten. Nu staan de bedieningselementen precies in de lege ruimte tussen de titel boven de video en de iets hierboven omlaag gezette video.

`border-top: black solid 1px;`

Randje aan de bovenkant, anders is er geen afscheiding tussen de bedienings-elementen en de titel erboven.

`#speed-3, #speed-4`

De elementen met id="speed-3" en id="speed-4". De 's waar de snelheidsregeling voor de derde en vierde videospeler in zit. Omdat deze voor de vierde speler vrijwel hetzelfde is als voor de derde, wordt veel css voor de snelheidsregeling hier voor beide spelers opgegeven.
Op de afbeelding is de normale snelheid gekozen, wat wordt aangegeven door het rode vierkantje.

`width: 480px; height: 28px;`

Breedte en hoogte aanpassen aan de grotere knoppen.

`position: absolute; bottom: -28px;`

Op de goede plaats zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `div#image-3` respectievelijk `div#image-4`.

`#speed-3 label, #speed-4 label`

De <label>'s binnen de elementen met id="speed-3" en id="speed-4". De <label>'s bij de knoppen voor de snelheidsregeling in de derde en vierde videospeler.
Van de algemene css voor deze pagina blijft alleen `display: block;` hetzelfde.
<label>'s zijn inline-elementen. Door er een blok-element van te maken, kun je eigenschappen als hoogte en breedte gebruiken.

`box-sizing: border-box;`

Normaal genomen worden marge, border en padding bij de breedte opgeteld. Hier wordt opgegeven dat de breedte inclusief marge, border en padding is.
Er zijn vier knoppen. Door ze alle vier 25% breed te maken, inclusief marge, border en padding, passen er vier naast elkaar. Zo ontloop je het risico dat in een of andere browser 'n pixel net verkeerd wordt afgerond waardoor de vierde knop niet naast de derde, maar daaronder komt te staan.

`width: 25%;`

Er zijn vier knoppen. Met deze breedte krijgen ze alle vier 'n kwart van de breedte van hun ouder: `div#speed-3`.

`height: 28px;`

Hoogte.

`float: left;`

In de algemene css voor deze pagina zijn de <label>'s met behulp van `display: block;` veranderd van inline-elementen in blok-elementen. Blok-elementen komen op een nieuwe regel te staan. Door ze naar links te floaten gebeurt dat niet, maar komen ze naast elkaar te staan.

`font-size: 20px;`

Tamelijk grote lettergrootte. Deze wordt gebruikt voor de teksten binnen de <label>'s: '0,5x', '1x', '1,5x' en '2x'. (Hoe je deze teksten eventueel in het script kunt wijzigen, staat bij Text).

`line-height: 28px;`

Door de regelhoogte even hoog te maken als de hoogte, staat de tekst verticaal gecentreerd.

`text-align: left;`

Tekst in de <label> links uitlijnen. Een aantal <label>'s is in de algemene css voor deze pagina rechts uitgelijnd, omdat er weinig ruimte is. Hier is genoeg ruimte.

`border: black solid; border-width: 1px 1px 0 0;`

Zwarte rand. De kleur en de stijl is aan alle vier de kanten hetzelfde. Bij de breedte wordt opgegeven, dat onder en rechts geen border moet komen. Dit kost minder css dan het opgeven van afzonderlijke borders.

`padding: 0; padding-left: 26px;`

De in de algemene css voor deze pagina opgegeven padding verwijderen en links een nieuwe padding van 26 px opgeven.

`#speed-3 label {border-width: 1px 1px 1px 0;}`

De <label>'s in het element met id="speed-3". De <label>'s bij de knoppen voor de snelheidsregeling in de derde videospeler.
Hier gelijk boven is een `border-width: 1px 1px 0 0;` opgegeven. Deze <label>'s moeten ook een border aan de onderkant krijgen, omdat ze boven de video staan.

`#speed-3 label:last-of-type, #speed-4 label:last-of-type {border-right: none;}`

Het laatste <label>-element binnen de elementen met id="speed-3" en id="speed-4". De <label>'s bij de laatste knoppen voor de snelheidsregeling.
Hierboven bij `#speed-3 label, #speed-4 label` hebben deze <label>'s een border aan de rechterkant gekregen. De laatste <label> moet die niet krijgen, anders komt die border tegen de rechterborder van `div#controls-3` te staan, wat een dubbele border op zou leveren.

`#speed-3 label::after, #speed-4 label::after`

Met behulp van `::after` wordt bij de <label>'s binnen de elementen met id="speed-3" en id="speed-4" een pseudo-element gemaakt. Met behulp van dat pseudo-element wordt aangegeven, welke snelheid is gekozen.
De css voor deze vier pseudo-elementen (eentje bij elk <label>) is grotendeels hetzelfde, die wordt hier in één keer opgegeven. Afwijkende css voor aparte elementen volgt dan hieronder.

`content: "";`

In de algemene css voor deze pagina is een vierkantje of een rondje opgeven als indicatie voor de gekozen snelheid. Dat vierkantje of rondje is een gewoon karakter. Hier wordt de gekozen snelheid op een andere manier aangegeven, dus wordt de content leeggemaakt.

`background: black; width: 14px; height: 14px;`

Zwarte achtergrond. 14 px breed, 14 px hoog. Dit levert dus een zwart vierkantje op van 14 x 14 px.

`border-radius: 7px;`

Omdat maar één waarde is opgegeven, worden alle hoeken even rond en precies cirkelvormig. Hoogte en breedte zijn 14 px, dus deze border-radius levert een volledig ronde cirkel op.

`top: 8px;`

Het geheel verticaal in het midden van de <label> neerzetten.

`#speed-3 input:checked + label::after, #speed-4 input:checked + label::after`

Voor deze elementen geldt ook de hierboven bij `#speed-3 label::after, #speed-4 label::after` opgegeven css, voor zover die hier niet wordt veranderd. Als een <input> in het element met id=”speed-3” of id=”speed-4” is aangevinkt, doe dan iets met het pseudo-element dat met behulp van `::after` bij de direct op de <input> volgende <label> is aangemaakt.

`background: red; border-radius: 0;`

Rode achtergrond, gewone rechte hoeken. De hierboven opgegeven zwarte cirkel verandert in een rood vierkant. Door niet alleen de kleur, maar ook de vorm te veranderen, is ook voor kleurenblinden duidelijk te zien, welke snelheid is gekozen.

```
#speed-3 .speed-half-label::after, #speed-4 .speed-half-label::after {left: 80px;}
#speed-3 .speed-default-label::after, #speed-4 .speed-default-label::after {left: 194px;}
#speed-3 .speed-one-half-label::after, #speed-4 .speed-one-half-label::after {left: 318px;}
#speed-3 .speed-double-label::after, #speed-4 .speed-double-label::after {left: 434px;}
```

In de algemene css voor deze pagina zijn de <label>’s al absoluut gepositioneerd. Hier worden de verschillende <label> voor de snelheidsregeling van de derde en vierde videospeler op de juiste plaats gezet.

`#video-3:not([controls]) ~ .groot {-webkit-animation: bugfix infinite 1s; margin-top: 320px;}`

Als je kiest voor de ingebouwde standaardspeler en dan weer kiest voor de aangepaste speler, staat in oudere versies van Android de video vervolgens 320 px te laag. Met behulp van de regel met `bugfix` wordt dit gecorrigeerd. Meer hierover bij @-webkit-keyframes bugfix. In dit geval is ook een marge boven de download-links nodig. Vraag me niet waarom, maar het werkt.

Speciaal voor vierde video (pagina 5)

De vierde videospeler is vrijwel hetzelfde als de derde, alleen staat hier de hele bediening onder de download-links. Heel veel css is al opgegeven bij de eerste en derde speler, omdat deze speler daar nogal veel mee gemeenschappelijk heeft.
Bij deze videospeler wijkt de volgorde van de elementen op het scherm af van de voor deze pagina opgegeven volgorde van de Tabindex. Daarom wordt onderaan de html de tabindex voor deze videospeler aangepast. Meer daarover bij Het JavaScript onderaan de html-bestanden.

`#sound-4` krijgt bij #sound-1, #sound-3, #sound-4 een breedte van 144 px.
`#softer-4` en `#louder-4` krijgen bij #softer-1, #louder-1, ... een breedte van 43 px.
`#sound-slider-beam-4` krijgt bij #sound-slider-beam-1, #sound-slider-beam-3, ... een breedte van 140 px.
`#image-4` krijgt bij #image-1, #image-3, #image-4 een breedte van 294 px.
`#duration-4` en `#remaining-4` worden bij #duration-1, #remaining-1, ... `met right: 0;` rechts neergezet.
`#speed-4` wordt bij #speed-3, #speed-4 op de goede plaats gezet en krijgt daar ook de juiste maten.

Het uiterlijk van de <label>'s van de vierde videospeler wordt bij #speed-3 label, #speed-4 label en verder geregeld. Hieronder staat slechts één kleine aanpassing.

```
#video-4:not([controls]) ~ .groot
```

`[controls]`: alleen als het attribuut 'controls' aanwezig is. Dit attribuut wordt door het script toegevoegd, als wordt gekozen voor de ingebouwde videospeler.
`:not`: maar omdat er in dit geval `:not` voor `[controls]` staat, geldt deze selector juist alleen als er géén attribuut 'controls' aanwezig is. Oftewel: als niet is gekozen voor de ingebouwde standaardspeler.
`#video-4:not([controls])`: als de vierde <video> geen attribuut 'controls' heeft.
`.groot`: de elementen met class="groot". Dit zijn de <p>'s waar de download-links in staan, eentje bij elke video.
`~`: `.groot` moet in de html moet volgen op `#video-4`, maar anders dan bij + hoeft `.groot` er niet gelijk op te volgen. Ook moeten `.groot` en `#video-4` dezelfde ouder hebben. Dat is hier `div#videobox-4`.

Omdat de download-links bij de vierde speler boven de bediening staan, moet er nogal wat van de algemene css voor deze pagina worden aangepast.

`height: 19px;`
De download-links moeten precies tussen de bedieningselementen en de video passen. Zonder hoogte ontstaat er in sommige browsers een hele kleine kier tussen de <p> met de download-links en de bedieningselementen.

`border-width: 0 1px 1px;`
Omdat voor links geen waarde is ingevuld, krijgt links automatisch dezelfde waarde als rechts. Hier staat dus eigenlijk `border-width: 0 1px 1px 1px;` in de volgorde boven – rechts – onder – links. Alleen aan de bovenkant geen border.

`border-radius: 0;`
De ronde hoekjes aan de onderkant moeten gewoon rechthoekig worden.

`position: relative; top: -97px;`

Ten opzichte van de normale positie 97 px omhoog plaats. Hiermee komt de <p> met de download-links precies boven de bedieningselementen te staan.

`#controls-4`

Het element met id=”controls-4”. De waar de bedieningselementen van de vierde videospeler in staan. Ook hier zijn wat aanpassingen nodig, omdat deze hier onder de download-links komt te staan.

`height: 70px;`

Omdat de snelheidsregeling hier onder de rest van de bediening staat, moet de hoog genoeg worden gemaakt om de achtergrondkleur e.d. ook onder de snelheidsregeling door te laten lopen.

`margin-top: 26px;`

Stukje omlaag zetten, zodat hij onder de download-links komt te staan.

`border-radius: 0 0 7px 7px;`

Rechts- en linksonder ronde hoekjes.

`#speed-4 {border-left: none;}`

Het element met id=”speed-4”. De met de snelheidsregeling. Links geen border, die is hier overbodig, want de border van `div#controls-4` is voldoende. Bovendien is de border van deze recht en zou daardoor onder ronde hoek linksonder uitsteken.

Speciaal voor vijfde video (pagina 5)

Bij de vijfde videospeler zijn de bedieningselementen rondom de video geplaatst. De zesde speler is, als de video speelt, grotendeels hetzelfde als deze speler. Daarom staat een groot deel van de css voor de zesde speler hier bij deze videospeler.
Bij deze videospeler wijkt de volgorde van de elementen op het scherm af van de voor deze pagina opgegeven volgorde van de Tabindex. Daarom wordt onder aan de html de tabindex voor deze videospeler aangepast. Meer daarover bij Het JavaScript onderaan de html-bestanden.

`#videobox-5, #videobox-6 {padding-top: 50px;}`

De elementen met id=”videobox-5” en id=”videobox-6”. De ’s waar de vijfde en zesde videospeler met bijbehorende titel, links, e.d. in staan.
Door de andere opbouw staan deze videospelers te dicht op de erboven staande. Vooral in smallere browservensters valt dat op. De makkelijkste manier om daar wat aan te doen, is het geven van een extra padding aan de bovenkant.

`#videobox-5 h2, #videobox-6 h2`

De <h2>’s binnen de elementen met id=”videobox-5” en id=”videobox-6”. Dit zijn de <h2>’s waarin de titel boven de vijfde en zesde video staat.

`max-width: 568px;`

In de algemene css voor deze pagina is een maximumbreedte van 474 px opgegeven. Omdat hier links en rechts van de video ook knoppen staan, wordt die breedte hier vergroot.

`height: 20px;`

Omdat het er wat benauwd uitzag, wordt wat extra hoogte aan de titel gegeven.

`position: relative; top: -50px;`

Ten opzichte van de normale positie wordt de <h2> 50 px naar boven verplaatst, precies boven de knoppen.

```
#videobox-5 .origineel, #videobox-6 .origineel {max-width: 568px;
    height: 14px; border-bottom: black solid 1px; position:
    relative; top: -50px;}
```

De elementen met class=”origineel” binnen de elementen met id=”videobox-5” en id=”videobox-6”. De <p>’s waarin de links naar de originele video’s van de vijfde en zesde video staan staan.

Voor de css geldt precies hetzelfde als gelijk hierboven bij `#videobox-5 h2, #videobox-6 h2` staat. Alleen is hier ook nog aan de onderkant een zwart randje toegevoegd.

`#video-5, #video-6`

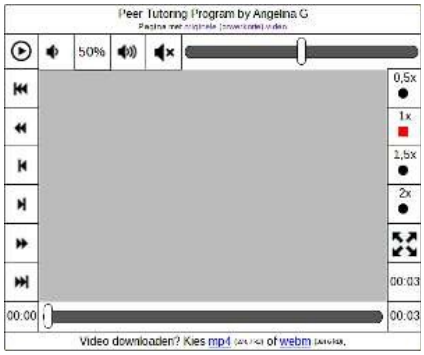
De elementen met id=”video-5” en id=”video-6”. Het vijfde en zesde <video>-element.

`position: relative; top: -3px;`

De relatieve positie maakt een kleine correctie naar boven mogelijk. De knoppen boven de video waren nogal lastig in elke browser exact even groot te krijgen. En omdat luiheid een gezonde eigenschap is, heb ik dat ook niet lang geprobeerd. Hieronder wordt gewoon de <video> 3 px omhoog gezet, waardoor alles boven de <video> op dezelfde hoogte eindigt. Althans: dat lijkt zo, want in werkelijkheid kapt de <video> de iets uitstekende knoppen e.d. dus gewoon af.

Op de afbeelding hieronder is, met enige moeite vanwege de verkleining, de ongelijke lijn aan de bovenkant te zien, omdat <video> in de afbeelding onzichtbaar is (de reden daarvan staat gelijk hieronder).

`z-index: 10;`



Zonder z-index zie je geen video, maar de achtergrond van `div.controls`: alleen bruikbaar in een afkickkliniek voor YouTube-verslaafden.

De bedieningselementen staan in een `div.controls`. Die staat normaal genomen op een andere plaats dan <video>, dus ook op een andere plaats dan de in <video> zittende video. Hier echter is `div.controls` nogal opgerekt om de knoppen aan alle kanten van de video neer te kunnen zetten. Daardoor staat `div.controls` nu ook over <video> heen, en dus ook over de daarin zittende video. En hoewel de grijze achtergrond van `div.controls` ongelooflijk rustgevend is, is het nut van een videospeler zonder zichtbare video mogelijk toch ietwat twijfelachtig. Daarom wordt de <video> met behulp van een z-index boven `div.controls` gezet.

Een z-index werkt alleen als het element relatief, positief of fixed is gepositioneerd. Dat is hier het geval, want even hierboven is een relatieve positie opgegeven.

`#video-5[controls], #video-6[controls]`

De elementen met id=”video-5” en id=”video-6”, de vijfde en zesde <video>, maar alleen als het attribuut ’controls’ aanwezig is. Als is gekozen voor de ingebouwde standaardspeler, voegt het script het attribuut ’controls’ in bij het <video>-element. Deze selector geldt dus alleen als is gekozen voor de ingebouwde standaardspeler.
Zonder de hier opgegeven css zou het er, als wordt gekozen voor de ingebouwde videospeler, uitzien zoals hiernaast op de afbeelding.

`margin: -48px auto -46px;`

Bij de aangepaste speler staan er boven de video bedieningsknoppen. Om daar ruimte voor te maken, zijn iets hierboven bij `#videobox-5 h2, #videobox-6 h2` en `#videobox-5 .origineel, #videobox-6 .origineel` de titel en link naar de originele video iets omhoog gezet.
Als is gekozen voor de ingebouwde standaardspeler, ontstaat hierdoor een kier tussen de video en de erboven staande titel en link. Het makkelijkste werk je die kier weg, door <video> 48 px omhoog te zetten met een negatieve marge.
Aan de onderkant wordt een negatieve marge van 46 px gegeven. Die heeft geen invloed op het element <video> zelf, maar wel op wat daaronder staat: dat wordt 46 px omhoog getrokken.
Als is gekozen voor de ingebouwde speler, staat onder <video> de <p> met de download-links. Ook tussen <video> en die <p> zit een kier, omdat de sleepbalk mist. Deze kier wordt weggewerkt door de negatieve marge aan de onderkant.

`padding: 47px;`

Aan alle kanten een padding van 47 px. Ook dit is feitelijk weer een onderdeel van het belazeren van de kluit. Het levert een mooie witte rand rondom de video op, maar in feite is het een rare sprong van een kat in het nauw.
Bij `#videobox-5 h2, #videobox-6 h2` en `#videobox-5 .origineel, #videobox-6 .origineel` iets hierboven zijn de titel en de link naar het origineel breder gemaakt, vanwege de knoppen links en rechts van de video. Iets meer hieronder gebeurt bij `#videobox-5 .groot, #videobox-6 .groot` hetzelfde voor de onder de video staande download-links.
Als is gekozen voor de in de browser ingebouwde speler, zijn deze drie elementen plotsklaps veel te breed, want er zijn dan helemaal geen knoppen links en rechts van de video.
Je zou `p.groot` met de download-links nog smaller kunnen maken, door te controleren op iets als `video[controls] ~ .groot` (doe iets met het op `video` volgende `.groot` als <video> een ’controls’-attribuut heeft). Maar titel en link naar het origineel staan vóór <video> in de html, dus daar werkt deze truc niet. Als het al kan, zou het gigantisch ingewikkeld worden om titel en link naar het origineel smaller te maken.
Daarom wordt `.origineel` niet smaller gemaakt, maar wordt <video> breder gemaakt met behulp van deze padding.

#controls-5, #controls-6

De elementen met id=”controls-5” en id=”controls-6”. De ’s waarin de bedienings-elementen van de vijfde en zesde speler staan.

width: 480px; height: 368px;

Breedte en hoogte.

margin-top: -372px;

372 px naar boven zetten. Zonder deze correctie zou de gewoon onder <video> komen te staan. Nu staat de onderkant van de gelijk met de onderkant van <video>

border: none;

Geen border.

position: relative;

Om nakomelingen van een element te kunnen positioneren ten opzichte van dat element, moet het element zelf een absolute, relatieve of fixed positie hebben. Omdat verder niets wordt opgegeven, heeft dit geen enkele invloed op het element zelf.

#play-5

Het element met id=”play-5”. De Speel-pauzeerknop van de vijfde videospeler.

background: url(../103-images/buttons-background-page-3-klein.png) -16px 7px no-repeat white;

Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen.

width: 48px; height: 48px; border: black solid; border-width: 0 1px 0;

Grootte en borders opgeven.

position: absolute; left: -48px;

Op de juiste plaats zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier div#controls-5.

#play-5.not-playing {background: url(../103-images/buttons-background-page-3-klein.png) -97px 7px no-repeat white;}

Het element met id=”play-5” en class=”not-playing”. De Speel-pauzeerknop van de vijfde videospeler als de video niet speelt. Alleen de achtergrond-afbeelding is anders dan gelijk hierboven, wanneer de video speelt.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Hierdoor kan, als de video niet speelt, een ander symbool worden getoond, dan wanneer de video wel speelt.

#play-5::after, #softer-5::after, #louder-5::after, #mute-5::after, #to-begin-5::after, #five-back-5::after, #ten-back-5::after, #ten-forward-5::after, #five-forward-5::after, #to-end-5::after, #fullscreen-5::before, #play-6::after, #mute-6::after, #to-begin-6::after, #five-back-6::after, #ten-back-6::after, #ten-forward-6::after, #five-forward-6::after, #to-end-6::after, #fullscreen-6::before {display: none;}

In de algemene css voor deze pagina is opgegeven dat voor de symbolen op de knoppen gewone karakters worden gebruikt. Deze worden met behulp van een met ::after of

`::before` gemaakt pseudo-element weergegeven. In de vijfde en zesde videospeler wordt gebruik gemaakt van een achtergrond-afbeelding voor de symbolen. Daarom worden die pseudo-elementen hier verborgen.

```
#controls-5 button:focus, #speed-5 input:focus + label,
    #controls-6 button:focus, #speed-6 input:focus + label
```

`#controls-5 button:focus`: als een <button> binnen het element met id=”controls-5” focus heeft, doe dan iets met die <button> (en idem voor de <button>’s in `#controls-6`). Samengevat: doe iets met een <button> als die focus heeft.

`#speed-5 input:focus + label`: als een <input> in het element met id=”speed-5” focus heeft, doe dan iets met de direct daarop volgende <label> (en idem voor de <input>’s in `#speed-6`). Dit zijn de <label>’s die bij de snelheidsregeling horen.

In de algemene css voor deze pagina is een witte achtergrond gegeven aan knoppen die focus hebben. De knoppen e.d. in de vijfde en zesde videospeler hebben al een witte achtergrond, dus er moet iets anders worden gebruikt om aan te geven welk element focus heeft. (Meer over focus bij Focus.) In deze spelers gebeurt dat door een blauwe outline rondom de knop te zetten.

`outline: blue solid 3px;`

Blauwe outline van 3 px dik.

`outline-offset: -3px;`

De outline 3 px, de dikte van de outline, naar binnen zetten. Hierdoor staat de outline binnen de knop, zodat hij niet net over de video heen komt te staan.
(Internet Explorer kent geen outline-offset. Daarin staat de outline dus soms iets over de video heen.)

```
#sound-5, #sound-6 {background: white; width: 527px; height: 49px;
    border-right: black solid 1px;}
```

De elementen met id=”sound-5” en id=”sound-6”. De ’s waarin de geluidsbesturing voor de vijfde en zesde videospeler staat.
Alleen wat simpele aanpassingen aan de algemene css voor de hele pagina voor wat betreft grootte en uiterlijk.

```
#sound-5 button, #sound-5 span {width: 49px; height: 49px; border:
    black solid; border-width: 0 1px 1px 0;}
```

De <button>’s en ’s binnen het element met id=”sound-5”. De Knoppen Harder, Zachter, Geluid aan/uit en de voor weergave van de geluidssterkte in de vijfde videospeler. Een aantal instellingen is voor alle knoppen hetzelfde, dat kan hier in één keer worden opgegeven.
De knoppen moeten groter worden dan is opgegeven in de css voor de hele pagina, en de border wordt ook iets aangepast.

```
#sound-5 .softer, #sound-6 .softer {background: url(../103-
    images/buttons-background-page-3.png) -424px no-repeat white;}
```

De elementen met class=”softer” binnen het element met id=”sound-5” en id=”sound-6”. De knoppen Zachter in de vijfde en zesde videospeler. Voor de knop in de vijfde speler geldt ook de gelijk hierboven bij `#sound-5 button, #sound-5 span` opgegeven css.

Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen.

```
#sound-5 .percentage, #sound-6 .percentage {background: white;
    font-size: 18px; line-height: 46px;}
```

De elementen met class=”percentage” binnen de elementen met id=”sound-5” en id=”sound-6”. De ’s waarin de geluidssterkte bij de vijfde en zesde videospeler wordt weergegeven. Voor de in de vijfde speler geldt ook de iets hierboven bij `#sound-5 button, #sound-5 span` opgegeven css.
De wordt groter weergegeven dan in de algemene css voor de pagina is opgegeven, met een witte achtergrond.

```
#sound-5 .louder, #sound-6 .louder {background: url(../103-
    images/buttons-background-page-3.png) -338px no-repeat white;}
```

De elementen met class=”louder” binnen het element met id=”sound-5” en id=”sound-6”. De knoppen Harder in de vijfde en zesde videospeler. Voor de knop in de vijfde speler geldt ook de iets hierboven bij `#sound-5 button, #sound-5 span` opgegeven css.
Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen.

```
#sound-5 .mute, #sound-6 .mute {background: url(../103-
    images/buttons-background-page-3-klein.png) -175px no-repeat
    white;}
```

De elementen met class=”mute” binnen het element met id=”sound-5” en id=”sound-6”. De knoppen Geluid aan/uit in de vijfde en zesde videospeler. Voor de knop in de vijfde speler geldt ook de iets hierboven bij `#sound-5 button, #sound-5 span` opgegeven css.
Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen.

```
#videobox-5[data-sound="no"] .sound, #videobox-6[data-
    sound="no"] .sound {display: block;}
```

Op iOS kan de geluidssterkte alleen worden veranderd met de knoppen op het apparaat, niet met de knoppen in de videospeler. Daarom is in de algemene css voor deze pagina de geluidsbesturing op iOS verborgen door aan het attribuut `data-sound` van `div.videobox` de waarde ’no’ te geven. Meer daarover bij .videobox[data-sound=”no”] .play.
Voor de vijfde en zesde speler wordt de geluidsbesturing weer zichtbaar gemaakt.

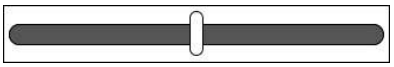
```
#videobox-5[data-sound="no"] .softer::after, #videobox-5[data-
    sound="no"] .louder::after, #videobox-5[data-
    sound="no"] .mute::after, #videobox-5[data-sound="no"] .sound-
    slider-button::after, #videobox-6[data-
    sound="no"] .softer::after, #videobox-6[data-
    sound="no"] .louder::after, #videobox-6[data-
    sound="no"] .mute::after, #videobox-6[data-sound="no"] .sound-
    slider-button::after {content: "x"; color: rgba(255, 0, 0,
    0.6); display: block; height: 37px; font-size: 50px; line-
    height: 28px; text-align: center; margin-top: -1px; speak:
    none;}
```

Op iOS kan de geluidssterkte alleen worden veranderd met de knoppen op het apparaat, niet met de knoppen in de videospeler. Daarom wordt op iOS een rood kruis door de bedieningselementen voor het geluid gezet.
Als het geluid niet kan worden aangepast binnen de videospeler, krijgt het attribuut `data-sound` van `div.videobox` de waarde 'no'. Meer daarover bij .videobox[data-sound="no"] .play. Door dit attribuut in de selector op te nemen, kan het uiterlijk worden aangepast.
Met behulp van `::after` wordt een pseudo-element aangemaakt, met behulp waarvan boven de knoppen een rode, grote, enigszins doorzichtige, 'x' wordt neergezet. Met wat extra css wordt de 'x' op de goede plaats gezet.

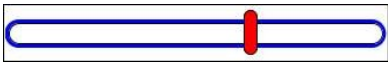
```
#sound-5 .mute.muted, #sound-6 .mute.muted {background:
    url(../103-images/buttons-background-page-3-klein.png) -259px
    no-repeat white;}
```

De Aan-uitknop voor het geluid heeft altijd een class="mute". Maar zodra het geluid wordt uitgezet, wordt aan de Aan-uitknop voor het geluid door het script een extra class toegevoegd: `.muted`. Deze extra class wordt weer verwijderd, zodra het geluid weer wordt aangezet.
Door in de selector deze extra class op te nemen, kan de achtergrond-afbeelding van de Aan-uit knop worden gewijzigd, als het geluid wordt uitgezet.

```
#sound-5 .sound-slider, #sound-6 .sound-slider {height: 48px;
    width: 331px; position: absolute; top: 0; right: 0;}
#sound-5 .sound-slider-beam, #sound-6 .sound-slider-beam {width:
    320px; height: 16px; border-radius: 10px; top: 15px;}
#sound-slider-beam-5:focus, #image-slider-beam-5:focus, #sound-
    slider-beam-6:focus, #image-slider-beam-6:focus {box-shadow: 0
    0 0 3px blue; outline: none;}
#sound-slider-button-5, #sound-slider-button-6 {width: 10px;
    height: 36px; border-radius: 6px; top: -12px;}
```



Ook het uiterlijk van de sleepbalk voor het geluid in de vijfde en zesde speler wordt aangepast. De eerste regel met `.sound-slider` is voor de waar de sleepbalk in staat, de tweede regel met `.sound-slider-beam` is voor de met de balk van de sleepbalk, de vierde regel met `#sound-slider-button-5` is voor de met de knop van de sleepbalk.



De derde regel met `:focus` regelt de focus. Niet alleen voor de sleepbalk voor het geluid, maar ook voor de sleepbalk voor weergave. (Meer over focus bij Focus.) Bij focus krijgt de sleepbalk een blauwe box-shadow. Omdat maar één waarde is opgegeven bij box-shadow, resulteert dit in een simpele blauwe lijn. Een gewone outline kan niet worden gebruikt, want die volgt de ronde hoeken niet. Een box-shadow volgt wel de ronde hoeken van de balk.

```
#videobox-5[data-sound="no"] .sound-slider-button::after,
    #videobox-6[data-sound="no"] .sound-slider-button::after
    {margin-left: -7px;}
```

Op iOS moet de plaats van het bij #videobox-5[data-sound=”no”] .softer::after opgegeven rode kruis iets worden aangepast.

```
#image-5, #image-6 {width: 574px; height: 362px; margin: -2px 0 0
    -48px; border: black solid; border-width: 0 1px 1px;}
```

De elementen met id=”image-5” en id=”image-6”. De ’s waarin de bedieningselementen voor de weergave voor de vijfde en zesde videospeler staan.
Alleen wat simpele aanpassingen aan grootte en uiterlijk van de algemene css voor de hele pagina.

```
#image-5 button, #image-6 button {width: 46px; height: 54px;
    border-width: 1px 0 0; position: absolute;}
```

De <button>’s binnen de elementen met id=”image-5” en id=”image-6”. De bedienings-knoppen voor weergave voor de vijfde en zesde videospeler. Een aantal instellingen is voor alle knoppen hetzelfde, dat kan hier in één keer worden opgegeven.
De knoppen moeten groter worden dan is opgegeven in de css voor de hele pagina, en de border wordt ook iets aangepast.
De absolute positie is nodig om de knoppen op de juiste plaats te zetten. Waar precies wordt later opgegeven, want dat verschilt per knop. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `#image-5` respectievelijk `#image-6`.

```
#videobox-5 .image .to-begin, #videobox-6 .image .to-begin
    {background: url(../103-images/buttons-background-page-3.png)
    -499px no-repeat white;}
```

De elementen met class=”to-begin” binnen de elementen met class=”image”, die weer binnen een element met id=”videobox-5” of id=”videobox-6” moeten liggen. De knoppen Naar begin van de vijfde en zesde videospeler. Voor deze knoppen geldt ook de gelijk hierboven bij `#image-5 button, #image-6 button` opgegeven css.
Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen.

```
#image-5 .five-back, #image-6 .five-back {background: url(../103-
    images/buttons-background-page-3.png) -658px no-repeat white;
    top: 54px;}
```

De elementen met class=”five-back” binnen het element met id=”image-5” en id=”image-6”. De knoppen Vijf procent terug van de vijfde en zesde videospeler. Voor deze knoppen geldt ook de iets hierboven bij `#image-5 button, #image-6 button` opgegeven css.

Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen. Met `top: 54px;` wordt de knop op de juiste plaats gezet.

```
#image-5 .ten-back, #image-6 .ten-back {background: url(../103-images/buttons-background-page-3.png) -816px no-repeat white; top: 106px;}
```

De elementen met class=”ten-back” binnen het element met id=”image-5” en id=”image-6”. De knoppen Tien seconden terug van de vijfde en zesde videospeler. Voor deze knoppen geldt ook de iets hierboven bij `#image-5 button, #image-6 button` opgegeven css.
Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen. Met `top: 106px;` wordt de knop op de juiste plaats gezet.

```
#image-5 .ten-forward, #image-6 .ten-forward {background: url(../103-images/buttons-background-page-3.png) -896px no-repeat white; top: 159px;}
```

De elementen met class=”ten-forward” binnen het element met id=”image-5” en id=”image-6”. De knoppen Tien seconden vooruit van de vijfde en zesde videospeler. Voor deze knoppen geldt ook de iets hierboven bij `#image-5 button, #image-6 button` opgegeven css.
Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen. Met `top: 159px;` wordt de knop op de juiste plaats gezet.

```
#image-5 .five-forward, #image-6 .five-forward {background: url(../103-images/buttons-background-page-3.png) -738px no-repeat white; top: 213px;}
```

De elementen met class=”five-forward” binnen het element met id=”image-5” en id=”image-6”. De knoppen Vijf procent vooruit van de vijfde en zesde videospeler. Voor deze knoppen geldt ook de iets hierboven bij `#image-5 button, #image-6 button` opgegeven css.
Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen. Met `top: 159px;` wordt de knop op de juiste plaats gezet.

```
#image-5 .to-end, #image-6 .to-end {background: url(../103-images/buttons-background-page-3.png) -578px no-repeat white; top: 266px;}
```

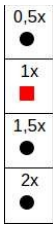
De elementen met class=”to-end” binnen het element met id=”image-5” en id=”image-6”. De knoppen Naar einde van de vijfde en zesde videospeler. Voor deze knoppen geldt ook de iets hierboven bij `#image-5 button, #image-6 button` opgegeven css.
Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen. Met `top: 266px;` wordt de knop op de juiste plaats gezet.

```
#elapsed-5, #duration-5, #remaining-5, #elapsed-6, #duration-6,
    #remaining-6 {background: white; width: 46px; height: 41px;
    font-size: 16px; line-height: 39px; border: black solid 1px;
    border-width: 1px 1px 0 0;}
```

De elementen met id=”elapsed-5”, id=”duration-5”, enz. De ’s van de vijfde en zesde videospeler waarin de verstreken, totale en resterende speelduur wordt weergegeven. Deze drie ’s zien er vrijwel het zelfde uit. Voor `#duration` en `#remaining` worden later nog wat kleine aanpassingen gegeven.

Een groot deel van de css is hetzelfde als die voor de hele pagina. Hier worden wat maten en borders e.d. aangepast.

```
#image-5 .speed, #image-6 .speed {width: 46px; height: 212px;
    9order-top: black solid 1px; padding-top: 0; line-height:
    22px;}
```



De elementen met class=”speed” binnen het element met id=”image-5” en id=”image-6”. De ’s waarbinnen de snelheidsregeling van de vijfde en zesde videospeler zit. De snelheidsregeling wordt in deze spelers verticaal weergegeven. Daarvoor zijn wat aanpassingen van een aantal maten nodig.

```
#speed-5 label, #speed-6 label
```

De <label>’s binnen het element met id=”speed-5” en id=”speed-6”. De <label>’s die horen bij de knoppen voor de snelheidsregeling in de vijfde en zesde videospeler.

```
background: white; width: 45px; height: 51px; font-size: 16px;
    text-align: center; border-bottom: black solid 1px;
```

Wat aanpassingen aan kleur, grootte, e.d.

```
padding-left: 1px;
```

1 px minder dan in de algemene css voor de pagina is opgegeven, anders worden de <label>’s net 1 px te breed en valt de border rechts weg.

```
position: absolute;
```

Om de <label>’s op de juiste plaats te kunnen zetten.

Nakomelingen van een element kunnen alleen gepositioneerd worden ten opzichte van dat element, als het element zelf absoluut, relatief of fixed is gepositioneerd. Dat is een bijkomend voordeel van deze absolute positie: de gelijk hieronder met behulp van `::after` gemaakte pseudo-elementen kunnen nu ten opzichte van de bijbehorende <label> worden gepositioneerd.

```
#speed-5 label::after, #speed-6 label::after
```

Met behulp van `::after` wordt bij de <label>’s binnen de elementen met id=”speed-5” en id=”speed-6” een pseudo-element gemaakt. Met behulp van dat pseudo-element wordt aangegeven, welke snelheid is gekozen.

De css voor deze vier pseudo-elementen (eentje bij elk <label>) is grotendeels hetzelfde, die wordt hier in één keer opgegeven. Afwijkende css voor aparte elementen volgt dan later.

```
content: "";
```

In de algemene css voor deze pagina is een vierkantje of een rondje opgeven als indicatie voor de gekozen snelheid. Dat vierkantje of rondje is een gewoon karakter. Hier wordt de gekozen snelheid op een andere manier aangegeven, dus wordt de content leeggemaakt.

`background: black; width: 14px; height: 14px;`

Zwarte achtergrond. 14 px breed, 14 px hoog. Dit levert dus een zwart vierkantje op van 14 x 14 px.

`border-radius: 7px;`

Omdat maar één waarde is opgegeven, worden alle hoeken even rond en precies cirkelvormig. Hoogte en breedte zijn 14 px, dus deze border-radius levert een volledig ronde cirkel op.

`position: absolute; top: 26px; left: 14px;`

Met behulp van een absolute positie worden de <label>'s op de juiste plaats gezet. Deze `top` en `left` zijn voor de <label> bij de halve snelheid. De andere <label>'s krijgen verderop een andere `top`, omdat anders alle <label>'s op dezelfde plaats en dus over elkaar heen zouden komen te staan.

Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `#speed-5` respectievelijk `#speed-6`.

`#speed-5 input:checked + label::after, #speed-6 input:checked + label::after`

Voor deze elementen geldt ook de hierboven bij `#speed-5 label::after, #speed-6 label::after` opgegeven css, voor zover die hier niet wordt veranderd. Als een <input> in het element met id="speed-5" of id="speed-6" is aangevinkt, doe dan iets met het pseudo-element dat met behulp van `::after` bij de direct op de <input> volgende <label> is aangemaakt.

`background: red; border-radius: 0;`

Rode achtergrond, gewone rechte hoeken. De hierboven opgegeven zwarte cirkel verandert in een rood vierkant. Door niet alleen de kleur, maar ook de vorm te veranderen, is ook voor kleurenblinden duidelijk te zien, welke snelheid is gekozen.

```
#speed-5 .speed-default-label, #speed-6 .speed-default-label {top: 53px;}
#speed-5 .speed-one-half-label, #speed-6 .speed-one-half-label {top: 106px;}
#speed-5 .speed-double-label, #speed-6 .speed-double-label {height: 52px; top: 159px;}
```

Voor deze elementen geldt ook de iets hierboven bij `#speed-5 label, #speed-6 label` opgegeven css, voor zover die hier niet wordt veranderd. Daar zijn deze <label>'s o.a. absoluut gepositioneerd. Voor de tweede, derde en vierde <label> wordt `top` aangepast, zodat de <label>'s niet over elkaar heen komen te staan.

De laatste <label> wordt 1 px hoger gemaakt dan de andere, zodat de hele handel goed past.

```
#videobox-5 .fullscreen, #videobox-6 .fullscreen {background: url(../103-images/buttons-background-page-3.png) -978px no-repeat white; top: 213px; right: 0;}
```

De elementen met class="fullscreen" binnen het element met id="videobox-5" en id="videobox-6". De knoppen Fullscreen van de vijfde en zesde videospeler. Voor deze knoppen geldt ook de iets hierboven bij `#image-5 button, #image-6 button` opgegeven css.

Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen. Met behulp van `top` en `right` wordt de knop op de juiste plaats gezet.

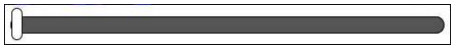
```
#duration-5, #duration-6 {height: 52px; line-height: 50px; border:
    none; border-top: black solid 1px; top: 267px; right: 0;}
```

De elementen met id=”duration-5” en id=”duration-6”. De ’s waarin de totale speelduur in de vijfde en zesde videospeler wordt weergegeven. Voor deze elementen geldt ook de eerder bij #elapsed-5, #duration-5, ... opgegeven css, voor zover die hier niet wordt veranderd.
Grootte en uiterlijk worden iets aangepast en de wordt op de juiste plaats gezet.

```
#image-5 .image-slider, #image-6 .image-slider {background: white;
    width: 480px; height: 41px; border-right: black solid 1px;
    position: absolute; bottom: 0; left: 47px; z-index: 10;}
```

De elementen met class=”image-slider” binnen de elementen met id=”image-5” en id=”image-6”. De ’s waarbinnen de sleepbalk voor weergave staat in de vijfde en zesde videospeler.
Voor deze ’s is in de algemene css voor deze pagina nauwelijks iets geregeld, dat gebeurt hier alsnog. Uiterlijk en positie worden aan de spelers aangepast. De z-index is nodig, omdat ook <video> eerder een z-index heeft gekregen. Zonder z-index zou het bovenste stukje van de anders onder <video> verdwijnen.

```
#image-5 .image-slider-beam, #image-6 .image-slider-beam {width:
    466px; height: 16px; margin: 0; border-radius: 10px; top:
    12px;}
```



De elementen met class= ”image-slider-beam” binnen de elementen met id=”image-5” en id=”image-6”. De ’s met de balk van de sleepbalk voor weergave van de vijfde en zesde videospeler.
Het uiterlijk en de positie worden wat aangepast.

```
#image-5 .image-slider-button, #image-6 .image-slider-button
    {width: 10px; height: 32px; border-radius: 6px; top: -10px;}
```

De elementen met class=”image-slider-button” binnen de elementen met id=”image-5” en id=”image-6”. De ’s met de knop van de sleepbalk voor weergave van de vijfde en zesde videospeler.
Het uiterlijk wordt aangepast aan de grotere sleepbalk.

```
#remaining-5, #remaining-6 {border-width: 1px 0 0; right: 0;}
```

De elementen met id=”remaining-5” en id=”remaining-6”. De ’s waarin de resterende speelduur van de vijfde en zesde videospeler wordt weergegeven. Voor deze elementen geldt ook de eerder bij #elapsed-5, #duration-5, ... opgegeven css, voor zover die hier niet wordt veranderd.
Border en positie worden aangepast.

```
#videobox-5 .groot, #videobox-6 .groot
```

De elementen met class=”groot” binnen de elementen met id=”videobox-5” en id=”videobox-6”. De <p>’s met de download-links van de vijfde en zesde videospeler.

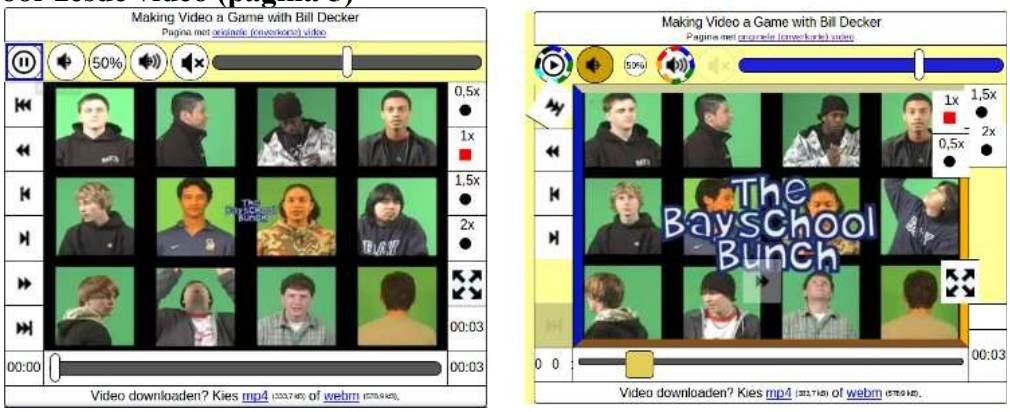
`max-width: 568px;`

In de algemene css voor deze pagina is een maximumbreedte van 474 px opgegeven. Omdat hier links en rechts van de video ook knoppen staan, wordt die breedte hier vergroot.

`margin-top: 42px;`

Deze <p>'s staan gelijk onder `div.controls`, de met de bedienings-elementen. Maar onder `div.controls` staat ook nog een rij bedieningselementen (de sleepbalk en de tijden links en rechts daarvan).
Om te voorkomen dat de <p> met de download-links gedeeltelijk verdwijnt onder de sleepbalk, krijgt de <p> een extra hoge marge aan de bovenkant.

Speciaal voor zesde video (pagina 5)



Als de video speelt, zoals links op de afbeelding, ziet alles er normaal uit. Als de video niet speelt, rechts op de afbeelding, is de bediening één grote, op hol geslagen, bewegende chaos.

Als de video speelt, ziet het er min of meer normaal uit, zoals links op de afbeelding. Maar als de video niet draait, slaan de knoppen en sleepbalken volledig op hol. Ze verschuiven, verkleuren, draaien, noem maar op.
Als de video speelt, ziet de speler er vrijwel hetzelfde uit als de vijfde videospeler. Daarom staat het overgrote deel van de css daar. Hier staan, wat betreft het uiterlijk, alleen wat kleine aanpassingen, die voornamelijk met de ronde knoppen linksboven hebben te maken.
Het overgrote deel van de css voor deze speler heeft te maken met het bewegen, verkleuren, draaien, e.d.
Niet alle animaties werken volledig in elke browser. Vooral in Android browser 4.0.3 werkt een en ander niet, en Opera op Linux is een compleet drama. (Opera op Linux werkt nog niet met Blink, de nieuwe weergave-machine van Opera. Op Windows, OS X en Android is er geen probleem met Opera.) Internet Explorer 9 kent `@keyframes` niet, dus daarin beweegt en verandert er niets. Een volledig overzicht van wat er niet werkt staat bij Bekende problemen (en oplossingen).
Bij deze videospeler wijkt de volgorde van de elementen op het scherm af van de voor deze pagina opgegeven volgorde van de Tabindex. Daarom wordt onder aan de html de tabindex voor deze videospeler aangepast. Meer daarover bij Het JavaScript onderaan de html-bestanden.
In verband met de beknoptheid (wat heet bij zo'n verhaal als dit …) is bij de css hieronder de -webkit-variant bij `@keyframes` weggelaten. Deze is wel aanwezig in het bijgesloten afbeelding-103-5-dl.css (en in de op de site gebruikte stylesheet).

`#controls-6 {background: transparent;}`

Het element met id="controls-6". De waar de bedieningselementen van de zesde videospeler in staan. Voor dit element geldt ook de eerder bij #controls-5, #controls-6 opgegeven css, voor zover die hier niet wordt veranderd.

De eerder voor de met de bedieningselementen opgegeven grijze achtergrond wordt doorzichtig gemaakt.

```
@keyframes border-change {
     0%, 100% {border-color: yellow red lime green;}
     50% {border-color: silver gold brown blue;}
}
```

Hier gelijk onder bij `#play-6.not-playing::before` wordt met behulp van een door `::before` aangemaakt pseudo-element de rand rondom de video langzaam verkleurd. Daar wordt ook opgegeven dat de animatie 2,5 seconde duurt en eindeloos doorgaat.
De volgorde van de kleuren is boven – rechts – onder – links. Aan het begin van de animatie (bij 0%) wordt de bovenste serie kleuren gebruikt. Halverwege de animatie (bij 50%, na 1,25 seconde) wordt de onderste serie kleuren gebruikt. Waarna de kleuren weer overgaan in de eerste serie kleuren, die aan het eind van de animatie (na 2,5 seconden, bij 100%) wordt bereikt. Daarna wordt dezelfde cyclus weer hervat.
Op de afbeelding staan drie willekeurige opnames. De video is even zwart gemaakt, zodat de border duidelijker is te zien. De geleidelijke verandering van de kleuren is op de afbeelding uiteraard niet te zien.

`#play-6.not-playing::before`

Maak bij het element met id=”play-6” en class=”not-playing” met behulp van `::before` een pseudo-element. Oftewel: maak bij de Speel-pauzeerknop van de zesde videospeler, als de video niet speelt, een pseudo-element. Dit pseudo-element wordt gebruikt om rondom de video een verkleurende rand weer te geven, zoals op de afbeelding gelijk hierboven bij `@keyframes border-change` is te zien.
Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de verkleurende rand alleen te tonen, als de video niet speelt.

`content: "";`

Ook al is er geen inhoud, `content` is toch nodig. Dat er geen inhoud is, wordt aangegeven door niets tussen de aanhalingstekens te zetten.

`-webkit-animation: border-change 2.5s infinite; animation: border-change 2.5s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: border-change 2.5s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam ’border-change’. Deze is gelijk hierboven bij `@keyframes border-change` gedefinieerd. Speel deze animatie volledig af in de tijd van 2,5 seconde. (In css gebruik je geen komma, maar een punt voor decimalen.) `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

`width: 459px; height: 299px;`

Met deze breedte en hoogte staat de rand precies binnen de video.

`border: solid 10px;`

10 px brede border. De kleuren worden bij de eerder gedefinieerde animatie opgegeven.

`position: absolute; top: 46px; left: 46px;`

Om de border op de goede plaats te zetten. Er wordt gepositioneerd ten opzichte van de eerste voorouder die zelf een absolute, relatieve of fixed positie heeft. Dat is hier `#play-6`.

`z-index: 10;`

Omdat <video> een z-index van 10 heeft, moet dit pseudo-element ook een z-index krijgen, anders verdwijnt de border onder <video>. (Wat vermoedelijk geen groot verlies zou zijn, maar dat is een andere discussie.)

```
#play-6 {background: url(../103-images/buttons-background-page-3-
    klein.png) -18px 6px no-repeat white; width: 45px; height:
    45px; border: black solid 1px; border-radius: 22px; position:
    absolute; top: 1px; left: -46px;}
```

 Het element met id=”play-6”. De Speel-pauzeerknop voor de zesde videospeler. Witte knop met ronde hoeken. Voor het symbool wordt, net als op de derde pagina, een achtergrond-afbeelding gebruikt, zoals omschreven bij Symbolen voor bedieningselementen.

```
#play-6.not-playing {background: url(../103-images/buttons-
    background-page-3-klein.png) -99px 6px no-repeat white;}
```

 Het element met id=”play-6” en class=”not-playing”. De Speel-pauzeerknop van de zesde videospeler als de video niet speelt. Alleen de achtergrond-afbeelding is anders dan gelijk hierboven, wanneer de video speelt.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Hierdoor kan, als de video niet speelt, een ander symbool worden getoond, dan wanneer de video wel speelt.

(Alleen de achtergrond-afbeelding is anders????? Andere oogarts? Nee hoor. Die gekleurde rand heeft hier niets mee te maken. Die wordt gelijk hieronder met behulp van een pseudo-element over de knop gezet. De achtergrond-afbeelding zorgt alleen voor de zwarte cirkel en het daarin zittende driehoekje.)

```
@keyframes turn {
    0% {transform: rotate(0deg);}
    100% {transform: rotate(360deg);}
}
```

 Hier gelijk onder bij `#play-6.not-playing::after, #play-6.not-playing + #sound-6 .louder::after` wordt met behulp van door `::after` aangemaakte pseudo-elementen een draaiende rand rondom de Speel-pauzeerknop en de knop Harder gezet. Daar wordt ook opgegeven dat de animatie 1,5 seconde duurt, eindeloos doorgaat en in dezelfde snelheid draait. Elke anderhalve seconde draait de rand dus één keer rond: van 0 naar 360 graden. Omdat van 0 naar 360 graden wordt gedraaid, is de draaiing met de klok mee.

Op de afbeelding staan drie willekeurige opnames. Omdat de rand draait, lijkt deze in werkelijkheid minder blokkerig dan de stilstaande rand op de afbeelding. Maar die draaiing is hier uiteraard niet te zien.

`#play-6.not-playing::after, #play-6.not-playing + #sound-6 .louder::after`

Eerste selector: maak bij het element met id=”play-6” en class=”not-playing” met behulp van `::after` een pseudo-element. Oftewel: maak, als de video niet speelt, een pseudo-element bij de Speel-pauzeerknop van de zesde videospeler.
Tweede selector: maak bij de elementen met class=”louder” die binnen het element met id=”sound-6” dat gelijk volgt op het element met id=”play-6” en class=”not-playing” met behulp van `::after` een pseudo-element. Oftewel: maak, als de video niet speelt, een pseudo-element bij de knop Harder van de zesde videospeler.
Deze tweede selector is iets ingewikkelder. Als je ’n + gebruikt, moeten de elementen voor en na de + dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#sound-6`, maar niet bij `#play-6` en `.louder`. Vandaar dat het noodzakelijk is `#sound-6` als ’n soort koppelaar tussen `#play-6` en `.louder` te zetten.
Deze pseudo-elementen worden gebruikt om een draaiende rand rondom de Speel-pauzeerknop en de knop Harder weer te geven. Op de afbeelding gelijk hierboven staat de Speel-pauzeerknop. Voor de knop Harder worden verderop bij #play-6.not-playing + #sound-6 .louder::after nog enkele kleine aanpassingen gedaan, daar staat ook de afbeelding.
Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de draaiende rand alleen te tonen, als de video niet speelt.

`content: "";`

Ook al is er geen inhoud, `content` is toch nodig. Dat er geen inhoud is, wordt aangegeven door niets tussen de aanhalingstekens te zetten.

`-webkit-animation: turn 1.5s linear infinite; animation: turn 1.5s linear infinite;`

Hier staat in feite twee keer hetzelfde: `animation: turn 1.5s linear infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam ’turn’. Deze is gelijk hierboven bij `@keyframes turn` gedefinieerd. Speel deze animatie volledig af in de tijd van 1,5 seconde. (In css gebruik je geen komma, maar een punt voor decimalen.)
`linear`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de draaiing steeds even snel is. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.
Voor de knop Harder wordt deze animatie later door een andere vervangen.

`display: block;`

Hier gelijk onder wordt het met behulp van `::after` gemaakte pseudo-element verborgen. Dat is ook de bedoeling als de video speelt. Maar het zou ook worden verborgen als de video niet speelt, en dat is niet de bedoeling. Daarom wordt hier expliciet opgegeven dat het pseudo-element moet worden getoond.
De selector om het te verbergen staat weliswaar hieronder en zou dus moeten ’winnen’ van deze regel, maar deze regel heeft meer specificiteit, meer ’gewicht’. Deze selector: `#play-6.not-playing::after`. De selector hieronder: `#play-6::after`. Deze selector heeft één pseudo-element meer dan de hieronder staande en ’wint’ daardoor.

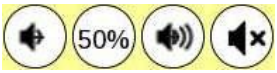
`width: 31px; height: 31px; border: dotted 6px; border-color: blue green cyan black; border-radius: 17px; position: absolute; top: 0; left: 0;`

Nog een serie css om de maten, de kleur van de border, de positie, e.d. op te geven.

```
#sound-6 {background: transparent;}
```

Het element met id=”sound-6”. De waarbinnen de volumeregeling voor de zesde videospeler zit. Voor dit element geldt ook de eerder bij #sound-5, #sound-6 opgegeven css, voor zover die hier niet wordt veranderd.
Eerder is de achtergrond van deze wit gemaakt. Hier wordt de achtergrond doorzichtig gemaakt.

```
#sound-6 button, #sound-6 span
```



De <button>’s en ’s binnen het element met id=”sound-6”. De knoppen en ’s voor de geluidsregeling in de zesde videospeler.

```
width: 45px; height: 45px; margin-left: 4px; border: black
    solid 1px; border-radius: 22px; top: 1px;
```

Een aantal instellingen is voor alle knoppen en ’s van de geluidsregeling hetzelfde, die worden hier in één keer opgegeven.

```
#sound-6 span {width: 43px; height: 43px;}
```

De ’s binnen het element met id=”sound-6”. Er is maar één van belang: de waarin het percentage van de geluidssterkte wordt weergegeven. Voor dit element geldt ook de gelijk hierboven bij `#sound-6 button, #sound-6 span` opgegeven css, voor zover die hier niet wordt veranderd.
Hierboven is 45 px opgegeven voor breedte en hoogte. Iets kleiner ziet er iets beter uit.

```
@keyframes backgroundcolor {
    0%, 80% {background-color: yellow;}
    10%, 100% {background-color: blue;}
    20% {background-color: cyan;}
    30% {background-color: orange;}
    40% {background-color: green;}
    50% {background-color: cyan;}
    60% {background-color: silver;}
    70% {background-color: brown;}
    90% {background-color: lime;}
}
```

Hier gelijk onder bij `#play-6.not-playing + #sound-6 .softer` wordt de hier gedefinieerde animatie gebruikt om de achtergrond van de knop Zachter te verkleuren. Daar wordt ook opgegeven dat de animatie 2 seconden duurt en eindeloos doorgaat.
Bij 0% (aan het begin van de animatie) is de achtergrond geel. Tussen 0% en 10% (van 0 tot 0,2 seconde) verandert de achtergrond van geel naar blauw. Tussen 10% en 20% (van 0,2 tot 0,4 seconde) verandert de achtergrond van blauw naar cyaan. Enz.,Geel en blauw komen twee keer terug.
Op de afbeelding staan drie willekeurige opnames. De geleidelijke verandering (die heel kort duurt) van de ene kleur naar de andere is op de afbeelding uiteraard niet te zien.

```
#play-6.not-playing + #sound-6 .softer
```

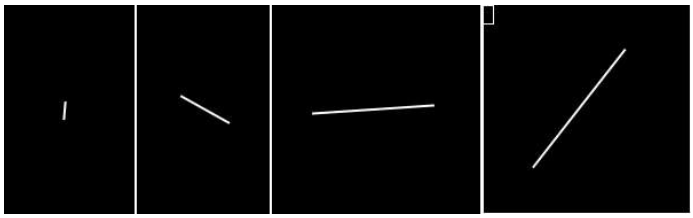
De elementen met class=”softer” die binnen het element met id=”sound-6” dat gelijk op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de knop Zachter van de zesde videospeler, als de video niet speelt.

Als je 'n + gebruikt, moeten de elementen voor en na de + dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#sound-6`, maar niet bij `#play-6` en `.softer`. Vandaar dat het noodzakelijk is `#sound-6` als 'n soort koppelaar tussen `#play-6` en `.softer` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Dit maakt het mogelijk de verkleurende achtergrond alleen te tonen, als de video niet speelt.

`-webkit-animation: backgroundcolor 2s infinite; animation: backgroundcolor 2s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: backgroundcolor 2s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam 'backgroundcolor'. Deze is gelijk hierboven bij `@keyframes backgroundcolor` gedefinieerd. Speel deze animatie volledig af in de tijd van 2 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes spiral {
	0%, 10% {
		border-width: 0;
		left: 238px;
	}
	10% {
		padding-left: 0;
		transform: rotate(0deg);
	}
	100% {
		border-width: 1px;
		padding-left: 150px;
		left: 163px;
		transform: rotate(14400deg);
	}
}
```



Hier gelijk onder bij `#play-6.not-playing + #sound-6::after` wordt de hier gedefinieerde animatie gebruikt om een ronddraaiende, groeiende lijn boven de video te laten verschijnen.
Daar wordt ook opgegeven dat de animatie 10 seconden duurt, eindeloos doorgaat en in een gelijkmatig tempo wordt vertoond.
('spiral' als naam voor 'n draaiende lijn? Nou, ik probeerde eerst eigenlijk een groeiende spiraal te maken, maar vond dit ook wel leuk...)
Op de afbeelding staan vier willekeurige opnames. Het ronddraaien en groeien van de lijn is uiteraard niet te zien. De video is op de afbeelding even zwart gemaakt, zodat de lijn beter zichtbaar is.
Wat je in feite ziet is de border van een verder leeg pseudo-element. Hier gelijk onder wordt het pseudo-element absoluut gepositioneerd en krijgt de border een witte kleur.
De totale animatie duurt 10 seconden, dus 10% van de animatie dus 1 seconde.

Van 0 tot 10% (van 0 tot 1 seconde) heeft de border geen breedte en is dus alles onzichtbaar. Bij 100% (aan het eind van de animatie, na 10 seconden) is de border 1 px.
Het element zelf is leeg, dus je ziet alleen twee borders die tegen elkaar aan staan, samen 2 px breed. Door afrondingen is de border vrij snel te zien, na zo'n 1-2 seconden.
Alleen aan de linkerkant is een padding gegeven. Alleen langs die padding zie je de border. Omdat het pseudo-element dus maar in één richting breedte heeft, zie je de borders maar in één richting, waardoor je in feite een rechte lijn van 2 px ziet.
Bij 10% (na 1 seconde) is de padding links 0 px. Bij 100% (aan het eind van de animatie, na 10 seconden) is dat 150 px. De padding groeit gedurende 9 seconden van 0 naar 150 px. Dat betekent dat ook de border groeit, want de border staat gewoon naast de padding. Aan het eind van de animatie is de border dus ook 150 px lang.
Omdat de border steeds langer wordt, moet `left` worden aangepast om het midden van de border in het midden van de video te houden. Van 0 tot 10% is `left` 238 px, daarna verandert dat geleidelijk naar 163 px bij 100%. `left` verandert dus mee met de padding.
Bij 10% (na 1 seconde) is `transform: rotate` (de draaiing) 0 graden. Bij 100% (aan het eind van de animatie, na 10 seconden) is de draaiing 14400 graden. Eén volledige omwenteling is 360 graden, dus 14400 graden is 40 omwentelingen. Van 10 tot 100%, van 1 tot 10 seconden, draait de border 40 keer in de rondte. Daarna begint de animatie weer van voren af aan begint te draaien.

`#play-6.not-playing + #sound-6::after`

Maak met behulp van `::after` een pseudo-element bij het element met id="sound-6" dat gelijk volgt op het element met id="play-6" en class="not-playing". Oftewel: de met de geluidsregeling, als de video niet speelt.
Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Dit maakt het mogelijk de draaiende lijn alleen te tonen, als de video niet speelt.

`content: "";`

Ook al is er geen inhoud, `content` is toch nodig. Dat er geen inhoud is, wordt aangegeven door niets tussen de aanhalingstekens te zetten.

`-webkit-animation: spiral 10s linear infinite; animation: spiral 10s linear infinite;`

Hier staat in feite twee keer hetzelfde: `animation: spiral 10s linear infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam 'spiral'. Deze is gelijk hierboven bij `@keyframes spiral` gedefinieerd. Speel deze animatie volledig af in de tijd van 10 seconden.
`linear`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de draaiing steeds even snel is. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

`border: white solid; position: absolute; top: 206px; z-index: 10;`

In samenhang met de hierboven bij `@keyframes spiral` opgegeven css zorgt dit voor de draaiende witte lijn. Bij de animatie zijn de eigenschappen die van waarde veranderen opgegeven, hier staan de eigenschappen die steeds hetzelfde blijven. `top` bijvoorbeeld blijft altijd hetzelfde, omdat alleen de padding links verandert.
De z-index is nodig, omdat ook <video> een z-index van 10 heeft. Zonder z-index zou de draaiende lijn onder de video verdwijnen.

```
@keyframes scale {
      0%, 100% {transform: scale(0.1);}
      50%, 75% {transform: scale(1);}
}
```

Hier gelijk onder bij `#play-6.not-playing + #sound-6.percentage` wordt de hier gedefinieerde animatie gebruikt om de waarin de geluidssterkte zit te verkleinen en te vergroten van vrijwel onzichtbaar tot normale grootte. Daar wordt ook opgegeven dat de animatie 1 seconde duurt en eindeloos doorgaat.
Bij 0% (aan het begin van de animatie) en 100% (aan het eind van de animatie, na 1 seconde) wordt de naar 0,1 van de normale grootte geschaald, vrijwel onzichtbaar.
Bij 50% (na 'n halve seconde) heeft de animatie de normale schaal: 1. Elke seconde gaat de knop dus van vrijwel onzichtbaar naar normaal zichtbaar naar vrijwel onzichtbaar.
Op de afbeelding staan drie willekeurige opnames. De geleidelijke verandering van klein naar groot en terug is op de afbeelding uiteraard niet te zien.

`#play-6.not-playing + #sound-6 .percentage`

De elementen met class="percentage" die binnen het element met id="sound-6" dat gelijk volgt op het element met id="play-6" en class="not-playing". Oftewel: de waarin de geluidssterkte van de zesde videospeler wordt weergegeven, als de video niet speelt.
Als je 'n + gebruikt, moeten de elementen voor en na de + dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#sound-6`, maar niet bij `#play-6` en `.percentage`. Vandaar dat het noodzakelijk is `#sound-6` als 'n soort koppelaar tussen `#play-6` en `.percentage` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Dit maakt het mogelijk de verandering van grootte alleen te tonen, als de video niet speelt.

```
-webkit-animation: scale 1s infinite; animation: scale 1s
      infinite;
```

Hier staat in feite twee keer hetzelfde: `animation: scale 1s infinite;`.
Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam 'scale'. Deze is gelijk hierboven bij `@keyframes scale` gedefinieerd. Speel deze animatie volledig af in de tijd van 1 seconde.
`infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes turn-back {
      0% {transform: rotate(360deg);}
      100% {transform: rotate(0deg);}
}
```

Hier gelijk onder bij `#play-6.not-playing + #sound-6 .louder::after` wordt met behulp van een door `::after` aangemaakt pseudo-element een draaiende rand rondom de knop Harder gezet. Daar wordt ook opgegeven dat de animatie 1,5 seconde duurt, eindeloos doorgaat en in dezelfde snelheid draait. Elke anderhalve seconde draait de rand dus één keer rond: van 360 naar 0 graden. Omdat van 360 naar 0 graden wordt gedraaid, is de draairichting tegen de klok in.
Op de afbeelding staan drie willekeurige opnames. Omdat de rand draait, lijkt deze in werkelijkheid minder blokkerig dan de stilstaande rand op de afbeelding, maar dat is op de afbeelding uiteraard niet te zien.

`#play-6.not-playing + #sound-6 .louder::after`

De elementen met class=”louder” die binnen het element met id=”sound-6” dat gelijk op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de knop Harder van de zesde videospeler, als de video niet speelt.

Als je ’n + gebruikt, moeten de elementen voor en na de + dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#sound-6`, maar niet bij `#play-6` en `.louder`. Vandaar dat het noodzakelijk is `#sound-6` als ’n soort koppelaar tussen `#play-6` en `.louder` te zetten.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de draaiing alleen te tonen, als de video niet speelt.

`-webkit-animation: turn-back 1.5s linear infinite; animation: turn-back 1.5s linear infinite;`

Hier staat in feite twee keer hetzelfde: `animation: turn-back 1.5s linear infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam ’turn-back’. Deze is gelijk hierboven bij `@keyframes turn-back` gedefinieerd. Speel deze animatie volledig af in de tijd van 1,5 seconde. (In css gebruik je geen komma, maar een punt voor decimalen.)

`linear`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de draaiing steeds even snel is. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

`border-color: red blue green orange;`

De eerder bij #play-6.not-playing::after, #play-6.not-playing + #sound-6 .louder::after opgegeven kleuren voor de border worden hier veranderd.

```
@keyframes transparency {
    0%, 100% {opacity: 1;}
    50% {opacity: 0;}
}
```

Hier gelijk onder bij `#play-6.not-playing + #sound-6 .mute` wordt de hier gedefinieerde animatie gebruikt om de Geluid aan-uitknop van doorzichtig in ondoorzichtig en terug te veranderen. Daar wordt ook opgegeven dat de animatie 2 seconden duurt en eindeloos doorgaat.

Bij 0% (aan het begin van de animatie) en bij 100% (aan het eind van de animatie, na 2 seconden) is de knop gewoon zichtbaar. Bij 50% (halverwege de animatie, na 1 seconde) is de knop volledig doorzichtig, dus onzichtbaar. Elke twee seconden verdwijnt en verschijnt de knop dus.

Op de afbeelding staan drie willekeurige opnames. De geleidelijke verandering van doorzichtig naar ondoorzichtig en terug is op de afbeelding uiteraard niet te zien.

`#play-6.not-playing + #sound-6 .mute`

De elementen met class=”mute” die binnen het element met id=”sound-6” dat gelijk op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de Geluid aan-uitknop van de zesde videospeler, als de video niet speelt.

Als je ’n + gebruikt, moeten de elementen voor en na de + dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#sound-6`, maar niet bij `#play-6` en `.mute`. Vandaar dat het noodzakelijk is `#sound-6` als ’n soort koppelaar tussen `#play-6` en `.mute` te zetten.

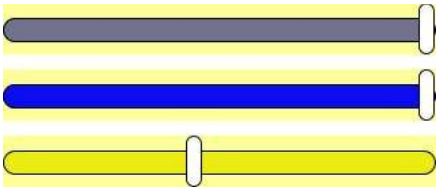
Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de doorzichtigheid alleen te veranderen, als de video niet speelt.

`-webkit-animation: transparency 2s infinite; animation: transparency 2s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: transparency 2s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam ’transparency’. Deze is gelijk hierboven bij `@keyframes transparency` gedefinieerd. Speel deze animatie volledig af in de tijd van 2 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes change-background {
    0%, 100% {background: blue;}
    50% {background: yellow;}
}
```



Hier gelijk onder bij `#play-6.not-playing + #sound-6 .sound-slider-beam` wordt de hier gedefinieerde animatie gebruikt om de achtergrondkleur van de balk van de sleepbalk voor het geluid te veranderen. Daar wordt ook opgegeven dat de animatie 10 seconden duurt, eindeloos doorgaat en regelmatig verloopt.

Bij 0% (aan het begin van de animatie) en bij 100% (aan het eind van de animatie, na 10 seconden) is de achtergrond blauw. Bij 50% (halverwege de animatie, na 5 seconden) is de achtergrond geel. De overgang van blauw naar geel duurt dus 5 seconden, en terug van geel naar blauw duurt ook weer vijf seconden.

Op de afbeelding staan drie willekeurige opnames. De geleidelijke verandering van kleur uiteraard niet te zien.

`#play-6.not-playing + #sound-6 .sound-slider-beam`

De elementen met class=”sound-slider-beam” die binnen het element met id=”sound-6” dat gelijk op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de balk van de sleepbalk voor geluid van de zesde videospeler, als de video niet speelt.

Als je ’n + gebruikt, moeten de elementen voor en na de + dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#sound-6`, maar niet bij `#play-6` en `.sound-slider-beam`. Vandaar dat het noodzakelijk is `#sound-6` als ’n soort koppelaar tussen `#play-6` en `.sound-slider-beam` te zetten.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de achtergrondkleur alleen te veranderen, als de video niet speelt.

`-webkit-animation: change-background 10s linear infinite; animation: change-background 10s linear infinite;`

Hier staat in feite twee keer hetzelfde: `animation: change-background 10s linear infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam ’change-background’. Deze is gelijk hierboven bij `@keyframes change-background` gedefinieerd. Speel deze animatie volledig af in de tijd van 10 seconden.

`linear`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de verkleuring steeds even snel is. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes move-button {
      0% {left: 0;}
      25%, 100% {left: 308px;}
}
```

Hier gelijk onder bij `#play-6.not-playing + #sound-6 .sound-slider-button` wordt de hier gedefinieerde animatie gebruikt om de positie van de knop van de sleepbalk voor het geluid te veranderen. Daar wordt ook opgegeven dat de animatie 5 seconden duurt, eindeloos doorgaat en aan het eind vertraagt.
Bij 0% (aan het begin van de animatie) staat de knop helemaal links op de sleepbalk. Bij 25% (na 1,25 seconde) en bij 100% (aan het eind van de animatie, na 5 seconden) staat de knop helemaal rechts op de sleepbalk (de sleepbalk is 320 px breed en de knop is 12 px breed, dus met `left: 308px;` staat de knop helemaal rechts). Tussen 25% en 100%, tussen 1,25 en 5 seconden, staat de knop gewoon rechts.
Van 0 tot 25% (tussen 0 en 1,25 seconde) verandert de positie van `left: 0;` naar `left: 308px;`. Omdat voor de soort beweging hieronder het sleutelwoord `ease-out` wordt gebruikt, vertraagt de beweging aan het eind.
Zodra de video begint te spelen, zet het script de knop op de positie die bij de ingestelde geluidssterkte hoort.
Een afbeelding – waar niets interessants op is te zien, wat betreft de knop – staat hier iets boven bij `@keyframes change-background`.

`#play-6.not-playing + #sound-6 .sound-slider-button`

De elementen met class=”sound-slider-button” die binnen het element met id=”sound-6” dat gelijk op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de knop van de sleepbalk voor geluid van de zesde videospeler, als de video niet speelt.
Als je ’n + gebruikt, moeten de elementen voor en na de + dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#sound-6`, maar niet bij `#play-6` en `.sound-slider-button`. Vandaar dat het noodzakelijk is `#sound-6` als ’n soort koppelaar tussen `#play-6` en `.sound-slider-button` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de positie van de knop alleen te veranderen, als de video niet speelt.

Die knop is dus echt nep...
Je kunt hier duidelijk zien dat die hele knop nep is. Ook terwijl die knop aan het ronddansen is, kun je de geluidssterkte gewoon instellen door op de balk van de sleepbalk te klikken, of daaroverheen te schuiven met muis of vinger. De geluidssterkte verandert, maar de knop trekt zich daar niets van aan. De geluidssterkte wordt bepaald door de sleepbalk, het script zet alleen voor de sier achteraf de knop op de bijbehorende positie. ’n Soort manager in de zorg: je ziet het, het denkt dat het belangrijk is en het doet helemaal niets, behalve zinloos rondschuiven.

```
    -webkit-animation: move-button 5s ease-out infinite;
        animation: move-button 5s ease-out infinite;
```

Hier staat in feite twee keer hetzelfde: `animation: move-button 5s ease-out infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam 'move-button'. Deze is gelijk hierboven bij `@keyframes move-button` gedefinieerd. Speel deze animatie volledig af in de tijd van 5 seconden.

`ease-out`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de animatie alleen aan het eind vertraagt. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes turn-jerky {
    0% {transform: rotate(0deg);}
    5% {transform: rotate(36deg);}
    10% {transform: rotate(24deg);}
    15% {transform: rotate(72deg);}
    20% {transform: rotate(62deg);}
    25% {transform: rotate(108deg);}
    30% {transform: rotate(98deg);}
    35% {transform: rotate(144deg);}
    40% {transform: rotate(134deg);}
    45% {transform: rotate(180deg);}
    50% {transform: rotate(170deg);}
    55% {transform: rotate(216deg);}
    60% {transform: rotate(206deg);}
    65% {transform: rotate(252deg);}
    70% {transform: rotate(242deg);}
    75% {transform: rotate(288deg);}
    80% {transform: rotate(278deg);}
    85% {transform: rotate(324deg);}
    90% {transform: rotate(314deg);}
    95% {transform: rotate(360deg);}
    100% {transform: rotate(350deg);}
}
```

Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .to-begin` wordt de hier gedefinieerde animatie gebruikt om de knop Naar begin schoksgewijs rond te laten draaien. Daar wordt ook opgegeven dat de animatie 20 seconden duurt en eindeloos doorgaat.

Bij 0% (aan het begin van de animatie) is de knop niet gedraaid. Bij 5% (na 1 seconde) is de knop 36 graden gedraaid. Na 10% (na 2 seconden) is de knop 24% gedraaid, weer 12 graden minder dan na 1 seconde. Na 15% (na 3 seconden) is de knop 72% gedraaid, een fors stuk vooruit. Na 20% (na 4 seconden) is de knop weer een stukje teruggedraaid, naar 62 graden. Steeds een fors stuk vooruit draaien en dan 'n klein stukje terug geeft een schokkerig effect. Op de afbeelding staan drie willekeurige opnames. De draaiende beweging zelf is op de afbeelding uiteraard niet te zien.

`#play-6.not-playing ~ #image-6 .to-begin`

De elementen met class=”to-begin” die binnen het element met id=”image-6” dat op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de knop Naar begin van de zesde videospeler, als de video niet speelt.

Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.to-begin`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.to-begin` te zetten.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de knop alleen te laten draaien, als de video niet speelt.

```
-webkit-animation: turn-jerky 20s infinite; animation: turn-jerky 20s infinite;
```

Hier staat in feite twee keer hetzelfde: `animation: turn-jerky 20s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam ’turn-jerky’. Deze is gelijk hierboven bij `@keyframes turn-jerkey` gedefinieerd. Speel deze animatie volledig af in de tijd van 20 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes jump {
    0%, 10% {top: 30px;}
    50% {top: 54px;}
}
```

Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .five-back` wordt de hier gedefinieerde animatie gebruikt om de knop Vijf procent terug langzaam iets naar beneden te bewegen, daar even te laten rusten en dan weer terug naar boven te laten springen. Daar wordt ook opgegeven dat de animatie 4 seconden duurt, eindeloos doorgaat en aan het begin iets vertraagd is.

Bij 0% (aan het begin van de animatie) en 10% (na 0,4 seconde) staat de knop te hoog (`top: 30px;`) . Na 0, 4 seconde begint de knop te bewegen, om bij 50% (halverwege de animatie, na 2 seconden) op de juiste plaats aan te komen: `top: 54px;`. Daar blijft de knop 2 seconden staan, om dan naar boven te springen.

Er is geen afbeelding, want een afbeelding maakt dit niet echt veel duidelijker.

`#play-6.not-playing ~ #image-6 .five-back`

De elementen met class=”five-back” die binnen het element met id=”image-6” dat volgt op op het element met id=”play-6” en class=”not-playing”. Oftewel: de knop Vijf procent terug van de zesde videospeler, als de video niet speelt.

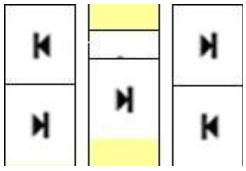
Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.five-back`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.five-back` te zetten.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de knop alleen te laten bewegen, als de video niet speelt.

`-webkit-animation: jump 4s ease-in infinite; animation: jump 4s ease-in infinite;`

Hier staat in feite twee keer hetzelfde: `animation: jump 4s ease-in infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam 'jump'. Deze is gelijk hierboven bij `@keyframes jump` gedefinieerd. Speel deze animatie volledig af in de tijd van 4 seconden.
`ease-in`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de animatie alleen aan het begin langzaam wordt afgespeeld. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes go-down {
    10%, 90% {top: 106px;}
    40%, 60% {top: 159px;}
}
```



Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .ten-back` wordt de hier gedefinieerde animatie gebruikt om de knop Tien seconden terug langzaam naar beneden te bewegen, daar even te laten rusten en dan weer terug naar boven te bewegen. Daar wordt ook opgegeven dat de animatie 10 seconden duurt en eindeloos doorgaat.
Gelijktijdig laat een andere animatie de knop Tien seconden verder, die gelijk onder deze knop staat, in hetzelfde tempo de andere kant op bewegen. Waardoor beide knoppen voortdurend met elkaar van plaats wisselen. Halverwege staan ze over elkaar heen.
Wat hier het nut van is? Geen enkel. Of ik dan geen zinvollere bezigheden heb? Gelukkig niet.
De knop is op 106 px vanaf de bovenkant gepositioneerd, de normale positie. Tot 10% van de animatie (1 seconde) blijft dat zo. Tussen 10% en 40% (tussen 1 en 4 seconden) wordt de knop van 106 px naar 159 px verplaatst. Die beweging duurt dus 3 seconden. Daar rust de knop 2 seconden, tot 60% van de animatie. Van 60% (6 seconden) tot 90% (9 seconden) beweegt de knop weer langzaam terug van 159 px naar 106 px. Om na 'n pauze van 90% tot 10% (2 seconden) weer aan een nieuwe afdaling te beginnen.
De animatie bij `@keyframes go-up` hieronder, die bij de knop Tien seconden terug hoort, is exact het spiegelbeeld van deze animatie.
Op de afbeelding staan drie willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien.

`#play-6.not-playing ~ #image-6 .ten-back`

De elementen met class="ten-back" die binnen het element met id="image-6" dat op het element met id="play-6" en class="not-playing" volgt. Oftewel: de knop Tien seconden terug van de zesde videospeler, als de video niet speelt.
Als je 'n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.ten-back`. Vandaar dat het noodzakelijk is `#image-6` als 'n soort koppelaar tussen `#play-6` en `.ten-back` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Dit maakt het mogelijk de knop alleen te laten bewegen, als de video niet speelt.

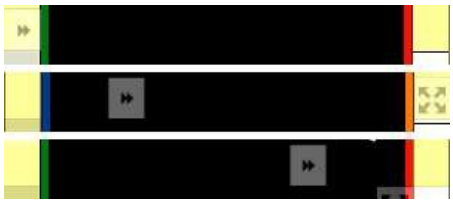
`-webkit-animation: go-down 10s infinite; animation: go-down 10s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: go-down 10s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam 'go-down'. Deze is gelijk hierboven bij `@keyframes go-down` gedefinieerd. Speel deze animatie volledig af in de tijd van 10 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes go-up {
    10%, 90% {top: 159px;}
    40%, 60% {top: 105px;}
}
#play-6.not-playing ~ #image-6 .ten-forward {-webkit-animation:
    go-up 10s infinite; animation: go-up 10s infinite;}
```

De animatie voor de knop Tien seconden verder. Deze is exact het spiegelbeeld van de animatie voor de knop Tien seconden terug gelijk hierboven bij `@keyframes go-down`. Daar is de uitleg te vinden.

```
@keyframes right-left {
    0%, 10% {left: 0;}
    0% {opacity: 1;}
    10%, 100% {opacity: 0.4;}
    100% {left: 528px;}
}
```



Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .five-forward` wordt de hier gedefinieerde animatie gebruikt om de knop Vijf procent verder langzaam van links naar rechts over de video te verplaatsen. Daar wordt ook opgegeven dat de animatie 20 seconden duurt, eindeloos doorgaat en gelijkmatig verloopt.
Van 0% (aan het begin van de animatie) tot 10% (na 2 seconden) staat de knop links op de normale plaats (`left: 0;`) . Bij 100% (aan het eind van de animatie, na 20 seconden) staat de knop met behulp van `left: 528px;` in de rij knoppen rechts van de video. De beweging duurt dus van 10% van de animatie tot 100% van de animatie (van 2 tot 20 seconden: 18 seconden).
Tegelijkertijd verandert de doorzichtigheid. Bij 0% (aan het begin van de animatie) is de knop ondoorzichtig. Van 0% naar 10% (gedurende de eerste 2 seconden) verandert de doorzichtigheid naar 0.4, om tot het eind van de animatie zo te blijven. De knop wordt dus eerst doorzichtig en gaat daarna pas bewegen. Omdat de knop doorzichtig is tijdens het bewegen, is de video door de knop heen zichtbaar.
Op de afbeelding staan drie willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien. Voor de duidelijkheid is de video even zwart gemaakt en zijn de meeste andere knoppen verborgen.

`#play-6.not-playing ~ #image-6 .five-forward`

De elementen met class=”five-forward” die binnen het element met id=”image-6” dat op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de knop Vijf seconden verder van de zesde videospeler, als de video niet speelt.

Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.five-forward`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.five-forward` te zetten.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de knop alleen te laten bewegen, als de video niet speelt.

`-webkit-animation: right-left 20s linear infinite; animation: right-left 20s linear infinite;`

Hier staat in feite twee keer hetzelfde: `animation: right-left 20s linear infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam ’go-down’. Deze is gelijk hierboven bij `@keyframes go-down` gedefinieerd. Speel deze animatie volledig af in de tijd van 20 seconden.

`linear`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de animatie volledig in hetzelfde tempo wordt uitgevoerd. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

`z-index: 10;`

Omdat <video> een z-index van 10 heeft, moet ook deze knop een z-index krijgen. Zonder z-index zou de knop, zodra hij gaat bewegen, onder <video> verdwijnen.

```
@keyframes hide {
    0%, 100% {
        background-color: white;
        opacity: 1;
    }
    50% {
        background-color: black;
        opacity: 0;
    }
}
```

Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .to-end` wordt de hier gedefinieerde animatie gebruikt om de knop Naar eind te laten verdwijnen en verschijnen. Daar wordt ook opgegeven dat de animatie 1 seconde duurt, eindeloos doorgaat en gelijkmatig verloopt.

Bij 0% (aan het begin van de animatie) en 100% (aan het eind van de animatie, na 1 seconde) is de knop gewoon ondoorzichtig, en is de achtergrond wit. Bij 50% (halverwege de animatie, na ’n halve seconde) is de knop volledig doorzichtig en is de achtergrond zwart. Die achtergrond zie je dan ook niet, maar tussen 0% en 50%, en tussen 50% en 100%, is de verkleuring van de achtergrond wel te zien.

Op de afbeelding staan drie willekeurige opnames. De verkleuring zelf is op de afbeelding uiteraard niet te zien.

`#play-6.not-playing ~ #image-6 .to-end`

De elementen met class=”to-end” die binnen het element met id=”image-6” dat op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de knop Naar eind van de zesde videospeler, als de video niet speelt.

Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.to-end`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.to-end` te zetten.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de knop alleen te veranderen, als de video niet speelt.

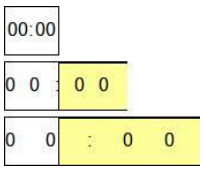
`-webkit-animation: hide 1s linear infinite; animation: hide 1s linear infinite;`

Hier staat in feite twee keer hetzelfde: `animation: right-left 20s linear infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam ’hide’. Deze is gelijk hierboven bij `@keyframes hide` gedefinieerd. Speel deze animatie volledig af in de tijd van 1 seconde.

`linear`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de animatie volledig in hetzelfde tempo wordt uitgevoerd. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes letter-space {
    0%, 100% {letter-spacing: 0;}
    50% {letter-spacing: 30px;}
}
```



Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .elapsed` wordt de hier gedefinieerde animatie gebruikt om de verstreken tijd uit te rekken en weer te laten krimpen. Daar wordt ook opgegeven dat de animatie 5 seconden duurt en eindeloos doorgaat.

Bij 0% (aan het begin van de animatie) en 100% (aan het eind van de animatie, na 5 seconden) staan de cijfers op de normale afstand. Bij 50% (halverwege de animatie, na 2½ seconde) staan de cijfers een eind uit elkaar.

Op de afbeelding staan drie willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien. Op de afbeelding zijn alle cijfers zichtbaar, in werkelijkheid verdwijnen cijfers en dubbele punt boven de gele achtergrond onder de sleepbalk voor afspelen.

`#play-6.not-playing ~ #image-6 .elapsed`

De elementen met class=”elapsed” die binnen het element met id=”image-6” dat op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de waarin de verstreken tijd van de zesde videospeler wordt weergegeven, als de video niet speelt.

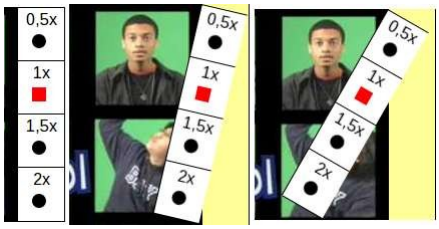
Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.elapsed`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.elapsed` te zetten.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de tijd alleen uit te rekken, als de video niet speelt.

`-webkit-animation: letter-space 5s infinite; animation: letter-space 5s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: letter-space 5s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam 'letter-space'. Deze is gelijk hierboven bij `@keyframes letter-space` gedefinieerd. Speel deze animatie volledig af in de tijd van 5 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes swing {
    10%, 90% {transform: rotate(0deg);}
    50% {transform: rotate(30deg);}
}
```



Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .speed` wordt de hier gedefinieerde animatie gebruikt om de snelheidsregeling heen en weer te laten slingeren. Daar wordt ook opgegeven dat de animatie 11 seconden duurt en eindeloos doorgaat.
Binnen de met de snelheidsregeling zitten vier knoppen. Omdat deze hele wordt bewogen, slingeren alle vier de daarin zittende knoppen ook gezellig mee.
Aan het begin van de animatie staat de gewoon normaal verticaal. Tussen 10% (na 1,1 seconde) en 50% (halverwege de animatie, na 5,5 seconde) draait de 30 graden met de klok mee. Om daarna weer terug te draaien en bij 90% (na 9,9 seconde) weer normaal te staan. Tussen 90% en 10%, gedurende 2,2 seconde, staat de in normale verticale stand.
Normaal genomen draait een element om het centrum. Hieronder is opgegeven dat het draaipunt de linkerbovenhoek van de is, waardoor het er niet als een draaiing, maar als een slingerbeweging uitziet.
Op de afbeelding staan drie willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien. Binnen de voor de snelheidsregeling wisselen ook de vier knoppen zelf van plaats. Dat is op deze afbeelding weggelaten, zodat de slingerbeweging duidelijker te zien is.

`#play-6.not-playing ~ #image-6 .speed`

De elementen met class="speed" die binnen het element met id="image-6" dat op het element met id="play-6" en class="not-playing" volgt. Oftewel: de waar de vier knoppen voor de snelheidsregeling van de zesde videospeler in zitten, als de video niet speelt.
Als je 'n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.speed`. Vandaar dat het noodzakelijk is `#image-6` als 'n soort koppelaar tussen `#play-6` en `.speed` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Dit maakt het mogelijk de alleen te laten bewegen, als de video niet speelt.

`-webkit-animation: swing 11s infinite; animation: swing 11s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: right-left 20s linear infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam 'swing'. Deze is gelijk hierboven bij `@keyframes swing` gedefinieerd. Speel deze animatie volledig af in de tijd van 11 seconden.

`linear`: standaard wordt een animatie in het begin langzaam, in het midden sneller en aan het einde weer langzamer afgespeeld. Dit zorgt ervoor dat de animatie volledig in hetzelfde tempo wordt uitgevoerd. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

`z-index: 10;`

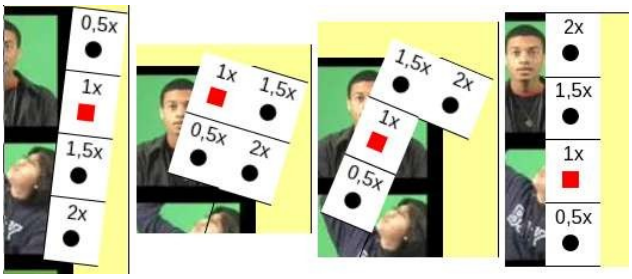
Omdat <video> een z-index van 10 heeft, moet ook deze een z-index krijgen. Zonder z-index zou de met de daarin zittende knoppen, zodra hij gaat bewegen, onder <video> verdwijnen.

`-webkit-transform-origin: 0 0; transform-origin: 0 0;`

Hier staat in feite twee keer hetzelfde: `transform-origin: 0 0;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Normaal genomen vindt een draaiing met `transform: rotate();`, zoals bij de animatie hierboven is opgegeven, plaats vanuit het midden van het element. Hier wordt opgegeven dat het draaipunt 0 px vanaf de bovenkant en 0 px vanaf de linkerkant van het element ligt. Hierdoor draait de rondom de linkerbovenhoek, alsof je als het ware een punaise in die hoek hebt geprikt waaromheen de draait.

```
@keyframes half-speed {
      0%, 100% {top: 0; left: 0;}
      7%, 81% {top: 0;}
      14% {top: 53px; left: 0;}
      20%, 69%, 75% {top: 53px;}
      25%, 31%, 57%, 63% {top: 106px;}
      36% {top: 159px;}
      42%, 93% {left: 0;}
      47%, 87% {left: -45px;}
      52% {top: 159px; left: -45px;}
}
```



Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .speed-half-label` wordt de hier gedefinieerde animatie gebruikt om de <label> bij de knop voor halve snelheid te verplaatsen. Daar wordt ook opgegeven dat de animatie 16 seconden duurt en eindeloos doorgaat.

Omdat de andere drie <label>'s voor de snelheidsregeling in hetzelfde tempo van plaats wisselen, krijg je 'n soort polonaise voor hoogbejaarden.

Op de afbeelding staan vier willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien. Omdat de waar de hele snelheidsregeling in staat zelf heen en weer slingert, slingeren de vier <label>'s ook nog gezamenlijk heen en weer.
Bij 0% (aan het begin van de animatie) en 100% (aan het eind van de animatie, na 16 seconden) staat de <label> op z'n normale plaats: `top: 0; left: 0;`.
Bij 7% (na 1,12 seconde) staat de <label> nog steeds bovenaan.
Bij 14% (na 2,24 seconde) krijgt de <label> `top: 53px;`: de knop gaat als het ware één knop omlaag.
Bij 20% (na 3,2 seconde), 69% (na 11,04 seconde) en 75% (na 12 seconden) staat de <label> ook op die hoogte.
Tussendoor, van 25% t/m 31% (van 4 seconden t/m 4,96 seconde) en van 57% t/m 63% (van 9,12 seconde t/m 10,08 seconde) staat de <label> echter lager: `top: 106px;`.
En daar weer tussendoor staat de <label> helemaal onderaan met `top: 159px;`, van 36% t/m 52% (van 5,76 seconde t/m 8,32 seconde).
Omdat de verplaatsingen geleidelijk gaan, schuift de <label> in de eerste helft van de animatie steeds 'n stukje omlaag, en in de tweede helft weer stapsgewijs omhoog.
Voor `left` geldt hetzelfde verhaal, alleen zijn daar maar twee posities: 0 en -45 px: ongeveer halverwege de animatie schuift de knop 'n plaatsje naar links.
Door de andere <label>'s op dezelfde momenten te bewegen, schuiven ze steeds 'n plaatsje op, om na acht verplaatsingen weer 'normaal' te staan.
De animaties voor de andere drie <label>'s voor de snelheid, die hieronder staan, werken precies hetzelfde, alleen zijn de posities steeds anders.

`#play-6.not-playing ~ #image-6 .speed-half-label`

De elementen met class="speed-half-label" die binnen het element met id="image-6" dat op het element met id="play-6" en class="not-playing" volgt. Oftewel: de <label> die bij de knop voor halve snelheid hoort van de zesde videospeler, als de video niet speelt.
Als je 'n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.speed-half-label`. Vandaar dat het noodzakelijk is `#image-6` als 'n soort koppelaar tussen `#play-6` en `.speed-half-label` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Dit maakt het mogelijk de knoppen alleen te verplaatsen, als de video niet speelt.

`-webkit-animation: half-speed 16s infinite; animation: half-speed 16s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: half-speed 16s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam 'half-speed'. Deze is gelijk hierboven bij `@keyframes half-speed` gedefinieerd. Speel deze animatie volledig af in de tijd van 16 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes default-speed {
     0%, 100% {top: 53px; left: 0;}
     7%, 57%, 93% {top: 53px;}
     14%, 20%, 47%, 52% {top: 106px;}
     25%, 42% {top: 159px;}
     31%, 81% {left: 0;}
     36%, 75% {left: -45px;}
     63% {top: 53px; left: -45px;}
     69% {top: 0;}
     87% {top: 0; left: 0;}
}

#play-6.not-playing ~ #image-6 .speed-default-label {-webkit-
     animation: default-speed 16s infinite; animation: default-
     speed 16s infinite;}

@keyframes one-half-speed {
     0%, 93%, 100% {top: 106px; left: 0;}
     7%, 36%, 42% {top: 106px;}
     14%, 31% {top: 159px;}
     20%, 69% {left: 0;}
     25%, 63% {left: -45px;}
     47%, 52%, 81%, 87% {top: 53px;}
     57% {top: 0;}
     75% {top: 0; left: 0;}
}

#play-6.not-playing ~ #image-6 .speed-one-half-label {-webkit-
     animation: one-half-speed 16s infinite; animation: one-half-
     speed 16s infinite;}

@keyframes double-speed {
     0%, 100% {top: 159px; left: 0;}
     7% {left: 0;}
     14% {left: -45px;}
     20%, 93% {top: 159px;}
     25%, 31% {top: 105px;}
     36%, 42%, 69%, 75% {top: 53px;}
     47% {top: 0;}
     52% {top: 0; left: -45px;}
     57% {top: -1px; left: 0;}
     63% {top: -1px;}
     81%, 87% {top: 106px;}
}
```

De css voor de andere drie <label>'s, die hierboven staat, werkt precies hetzelfde als die voor de eerste <label>, die iets verder naar boven staat.

```
#play-6.not-playing ~ #image-6 .speed-double-label {-webkit-
     animation: double-speed 16s infinite; animation: double-speed
     16s infinite;}

@keyframes round {
     0%, 100% {top: 213px; left: 528px;}
     5%, 85% {left: 528px;}
     6%, 85% {top: 213px; left: 481px;}
     7%, 40% {top: 266px;}
     20%, 65%, 80% {left: 481px;}
     25%, 60% {left: 47px;}
     45%, 80% {top: 0;}
     95% {left: 481px;}
}
```

Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .fullscreen` wordt de hier gedefinieerde animatie gebruikt om de knop Fullscreen naar alle vier de hoeken van de video te verplaatsen. Daar wordt ook opgegeven dat de animatie 13 seconden duurt en eindeloos doorgaat.

Op de afbeelding staan drie willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien.

Van 0% (het begin van de animatie) tot 6% (na 0,78 seconde) staat de knop op de normale hoogte met `top: 213px;`.

Tussen 6% en 7% (tussen 0,78 en 9,1 seconde) wordt de knop met `top: 266px;` naar omlaag verplaatst. Dit is de onderkant van de video. Tot 40% (5,2 seconde) blijft de knop daar staan.

Tussen 40% en 45% (tussen 5,2 en 5,85 seconde) wordt de knop met `top: 0;` gelijk gezet met de bovenkant van de video. Tot 80% (10,4 seconde) blijft de knop daar staan.

Tussen 80% en 85% (tussen 10,4 en 11,5 seconde) wordt de knop met `top: 213px;` verplaatst naar de normale hoogte. Tot 100% (het einde van de animatie, na 13 seconden) blijft hij daar staan.

Omdat de verplaatsingen geleidelijk aan plaatsvinden, glijdt de knop steeds naar een nieuwe positie. Daar blijft de knop even staan, waarna naar de volgende positie wordt gegleden.

Hierboven is alleen `top` genoemd, maar als je dit gaat combineren met de instellingen voor `left`, verplaatst de knop zich van de normale positie naar achtereenvolgens alle vier de hoeken van de video.

`#play-6.not-playing ~ #image-6 .fullscreen`

De elementen met class=”fullscreen” die binnen het element met id=”image-6” dat op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de knop Fullscreen van de zesde videospeler, als de video niet speelt.

Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.fullscreen`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.fullscreen` te zetten.

Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de knop alleen te verplaatsen, als de video niet speelt.

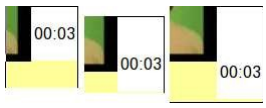
`-webkit-animation: round 13s infinite; animation: round 13s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: round 13s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam ’round’. Deze is gelijk hierboven bij `@keyframes round` gedefinieerd. Speel deze animatie volledig af in de tijd van 13 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

`z-index: 10;`

Omdat <video> een z-index van 10 heeft, moet deze knop ook een z-index krijgen. Anders zou hij onder <video> verdwijnen.

```
@keyframes top-padding {
	0%, 100% {padding-top: 0;}
	50% {padding-top: 40px;}
}
```



Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .duration` wordt de hier gedefinieerde animatie gebruikt om de totale speelduur omlaag te zetten en weer omhoog te laten komen. Daar wordt ook opgegeven dat de animatie 5 seconden duurt en eindeloos doorgaat.
Op de afbeelding staan drie willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien. Links staat de normale positie. Midden staat de tijd lager, maar is nog zichtbaar. Rechts staat de tijd helemaal onderaan. In werkelijkheid is ze dan niet meer zichtbaar, omdat ze onder de met de resterende speelduur is verdwenen. (Op de afbeelding bij `@keyframes bottom-padding` iets hieronder is dat verdwijnen te zien.)
Bij 0% (het begin van de animatie) en 100% (het einde van de animatie, na 5 seconden) is geen padding aan de bovenkant aanwezig. Bij 50% (halverwege de animatie, na 2,5 seconden) is een padding van 40 px aan de bovenkant aanwezig.
Omdat de eerder absoluut is gepositioneerd met behulp van `top: 267px;`, duwt deze padding de tijd naar beneden.

`#play-6.not-playing ~ #image-6 .duration`

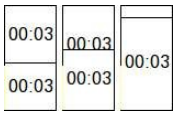
De elementen met class=”duration” die binnen het element met id=”image-6” dat op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de ’s waarin de totale speelduur wordt weergegeven, als de video niet speelt.
Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.duration`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.duration` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de tijd alleen te verplaatsen, als de video niet speelt.

`-webkit-animation: top-padding 5s infinite; animation: top-padding 5s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: top-padding 5s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam 'top-padding'. Deze is gelijk hierboven bij `@keyframes top-padding` gedefinieerd. Speel deze animatie volledig af in de tijd van 5 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes bottom-padding {
      0%, 100% {padding-bottom: 0;}
      50% {
            line-height: 90px;
            padding-bottom: 53px;
      }
}
```



Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .remaining` wordt de hier gedefinieerde animatie gebruikt om de resterende speelduur omhoog te zetten en weer omlaag te laten komen. Daar wordt ook opgegeven dat de animatie 5 seconden duurt en eindeloos doorgaat.
Op de afbeelding staan drie willekeurige opnames. De bovenste tijd is de totale speelduur, die verdwijnt onder de onderste tijd, de resterende speelduur. De beweging zelf is op de afbeelding uiteraard niet te zien. Links staat de normale positie. Midden staat de totale speelduur lager en is de resterende speelduur al wat omhoog gegroeid. Rechts is van de totale speelduur niets meer te zien en vult de met de resterende speelduur vrijwel de hele ruimte voor beide tijden.
Bij 0% (het begin van de animatie) en 100% (het einde van de animatie, na 5 seconden) is geen padding aan de onderkant aanwezig. Bij 50% (halverwege de animatie, na 2,5 seconden) is een padding van 53 px aan de onderkant aanwezig.
Omdat de eerder absoluut is gepositioneerd met behulp van `bottom: 0;`, laat deze padding de naar boven toe groeien. De in de staande tijd wordt ook omhoog geduwd. Om deze tijd in het midden van de te houden, wordt de regelhoogte halverwege de animatie verhoogd naar 90 px.
De beweging voor de totale speelduur is gelijk hierboven bij `@keyframes top-padding` geregeld. Omdat beide animaties dezelfde tijd hebben, verdwijnt de totale speelduur in een regelmatig tempo achter de resterende speelduur, en komt ook weer in een regelmatig tempo te voorschijn.

`#play-6.not-playing ~ #image-6 .remaining`

De elementen met class="remaining" die binnen het element met id="image-6" dat op het element met id="play-6" en class="not-playing" volgt. Oftewel: de 's waarin de totale speelduur wordt weergegeven, als de video niet speelt.
Als je 'n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.remaining`. Vandaar dat het noodzakelijk is `#image-6` als 'n soort koppelaar tussen `#play-6` en `.remaining` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class="not-playing" toegevoegd, als de video niet speelt. Dit maakt het mogelijk de tijd alleen te verplaatsen, als de video niet speelt.

`-webkit-animation: bottom-padding 5s infinite; animation: bottom-padding 5s infinite;`

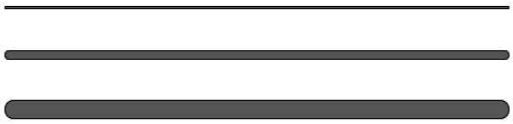
Hier staat in feite twee keer hetzelfde: `animation: bottom-padding 5s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.

Gebruik de animatie met de naam ’bottom-padding’. Deze is gelijk hierboven bij `@keyframes bottom-padding` gedefinieerd. Speel deze animatie volledig af in de tijd van 5 seconden. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

`z-index: 10;`

Omdat `span.remaining` in de html na `span.duration` komt, zou `span.remaining` onder `span.duration` verdwijnen, als de hoger wordt gemaakt. Door er een z-index aan te geven, is `span.remaining` toch zichtbaar.

```
@keyframes small {
     0%, 100% {height: 16px;}
     50% {height: 0;}
}
```



Hier gelijk onder bij `#play-6.not-playing ~ #image-6 .image-slider-beam` wordt de hier gedefinieerde animatie gebruikt om de balk van de sleepbalk voor het afspelen dunner en dikker te maken. Daar wordt ook opgegeven dat de animatie 1 seconde duurt en eindeloos doorgaat.
Op de afbeelding staan drie willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien.
Bij 0% (het begin van de animatie) en 100% (het einde van de animatie, na 1 seconde) is de hoogte 16 px. Bij 50% (halverwege de animatie, na 0,5 seconde) is de hoogte 0 px. Omdat dit elke seconde opnieuw gebeurt, levert dit de enige sleepbalk ter wereld met hyperventilatie op.

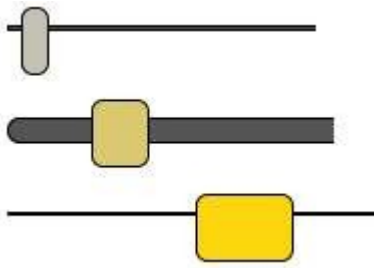
`#play-6.not-playing ~ #image-6 .image-slider-beam`

De elementen met class=”image-slider-beam” die binnen het element met id=”image-6” dat op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de balk van de sleepbalk voor afspelen, als de video niet speelt.
Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.image-slider-beam`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.image-slider-beam` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de balk alleen te veranderen, als de video niet speelt.

`-webkit-animation: small 1s infinite; animation: small 1s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: small 1s infinite;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam ’small’. Deze is gelijk hierboven bij `@keyframes small` gedefinieerd. Speel deze animatie volledig af in de tijd van 1 seconde. `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

```
@keyframes glow {
      0%, 100% {
            background: silver;
            width: 10px;
            left: 0;
      }
      50% {
            background: gold;
            width: 50px;
            left: 100px;
      }
}
```



Hier gelijk onder bij `#play-6.not-playing + #sound-6 .image-slider-button` wordt de hier gedefinieerde animatie gebruikt om de positie, de breedte en de kleur van de knop van de sleepbalk voor het afspelen te veranderen. Daar wordt ook opgegeven dat de animatie 0,9 seconde duurt en eindeloos doorgaat.
Bij 0% (aan het begin van de animatie) en 100% (aan het eind van de animatie, na 0,9 seconde) is de achtergrond zilverkleurig, is de breedte 10 px en staat de knop helemaal links op de sleepbalk. Bij 50% (na 0,45 seconde) is de achtergrond goudkleurig, is de breedte 50 px en staat de knop 100 px van links.
Zodra de video begint te spelen, zet het script de knop op de positie die bij de verstreken speelduur hoort.
Op de afbeelding staan drie willekeurige opnames. De beweging zelf is op de afbeelding uiteraard niet te zien.

`#play-6.not-playing ~ #image-6 .image-slider-button`

De elementen met class=”image-slider-button” die binnen het element met id=”image-6” dat op het element met id=”play-6” en class=”not-playing” volgt. Oftewel: de knoppen van de sleepbalk voor afspelen, als de video niet speelt.
Als je ’n ~ gebruikt, moeten de elementen voor en na de ~ dezelfde ouder hebben. Dat is wel het geval bij `#play-6` en `#image-6`, maar niet bij `#play-6` en `.image-slider-button`. Vandaar dat het noodzakelijk is `#image-6` als ’n soort koppelaar tussen `#play-6` en `.image-slider-button` te zetten.
Aan de Speel-pauzeerknop wordt door het script een class=”not-playing” toegevoegd, als de video niet speelt. Dit maakt het mogelijk de knop alleen te veranderen, als de video niet speelt.

`-webkit-animation: glow 0.9s infinite; animation: glow 0.9s infinite;`

Hier staat in feite twee keer hetzelfde: `animation: glow 0.9s infinite;`.
Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Gebruik de animatie met de naam ’glow’. Deze is gelijk hierboven bij `@keyframes glow` gedefinieerd. Speel deze animatie volledig af in de tijd van 0,9 seconden. (In css gebruik je geen komma, maar een punt voor decimalen.) `infinite` wil zeggen dat de animatie non-stop wordt herhaald.

JavaScript

In het deel over JavaScript worden geen regelnummers gebruikt. Dat heeft ’n simpele reden: als er bovenaan het script één regel wordt toegevoegd, typ ik mij een dubbele RSI aan te wijzigen regelnummers. Als je iets zoekt in ’n script, gebruik dan de zoekfunctie van je editor.

Overzicht van door het script ingevoegde bedieningselementen, classes en id’s

Als een bepaald bedieningselement niet wordt gebruikt, kan dit worden verborgen met `display: none;` Achter de schermen is het nog steeds aanwezig, maar op het scherm merk je er niets meer van.

Het script voegt voornamelijk bedieningselementen voor de videospelers in, maar bovenaan de pagina wordt ook nog een keuzemogelijkheid voor de ingebouwde standaard- of de aangepaste videospelers ingevoegd. Deze keuzemogelijkheid staat in een , die een id=”kiezen-div” krijgt. Deze id (en alle andere hieronder genoemde id’s en classes), kunnen eventueel bovenin het script worden gewijzigd. Hoe dat moet, staat bij ClassId.
Deze wordt altijd helemaal bovenin de html ingevoegd, gelijk na <body>. Met behulp van css kan hij eventueel lager worden geplaatst, onder andere content, maar in de html is de vaste plaats gelijk na <body>.
Binnen deze zitten twee <p>’s. Deze krijgen beide van het script een class=”kiezen-par”. De eerste <p> krijgt daarnaast een id=”kiezen-par-1”, de tweede een id=”kiezen-par-2”.
In de eerste <p> voegt het script een <label> in. Dit <label> krijgt van het script een class=”kiezen-label” en een id=”kiezen-label-1”. Gelijk hierachter zit een <input type=”radio”>, die een class=”kiezen-knop” en een id=”kiezen-knop-1” krijgt.
De radioknop zit dus niet binnen de bijbehorende <label>. Het script koppelt de <label> met een `for` aan de bijbehorende knop. Door de radioknop buiten de <label> te zetten, is een en ander beter op te maken met css.
De in de <p> staande tekst is bovenin het script te wijzigen. Hoe dat moet, staat bij Text.

De inhoud van de tweede <p> is precies hetzelfde als die van de eerste, alleen eindigen de id’s van <p>, <label> en radioknop op ’-2’ in plaats van ’-1’.

Alle bedieningselementen van de videospelers worden door het script ingevoegd. Om css te kunnen koppelen aan de diverse bedieningselementen, moet er iets van een houvast zijn. Het makkelijkste kan dit, door elk bedieningselement een class en een id te geven. Ook deze classes en id’s worden door het script ingevoegd. De namen zijn in het script opgenomen, maar deze kunnen eventueel bovenin het script worden gewijzigd. Hoe dat moet, staat bij ClassId.
De class is voor alle bedieningselementen van dezelfde soort hetzelfde, op de hele pagina. Alle knoppen om naar het einde van de video te gaan bijvoorbeeld hebben een class=”to-end”. Hierdoor kun je in één keer alle bedieningselementen van een bepaalde soort selecteren.
Daarnaast heeft elk bedieningselement een unieke id. De naam van die id is hetzelfde als de naam van de class van het betreffende bedieningselement, gevolgd door een koppelteken en een volgnummer. Bij de eerste video is dat ’-1’, bij de tweede video ’-2’, enz. De code voor de bij de eerste videospeler horende knop om naar het einde te gaan, ziet er dan zo uit:

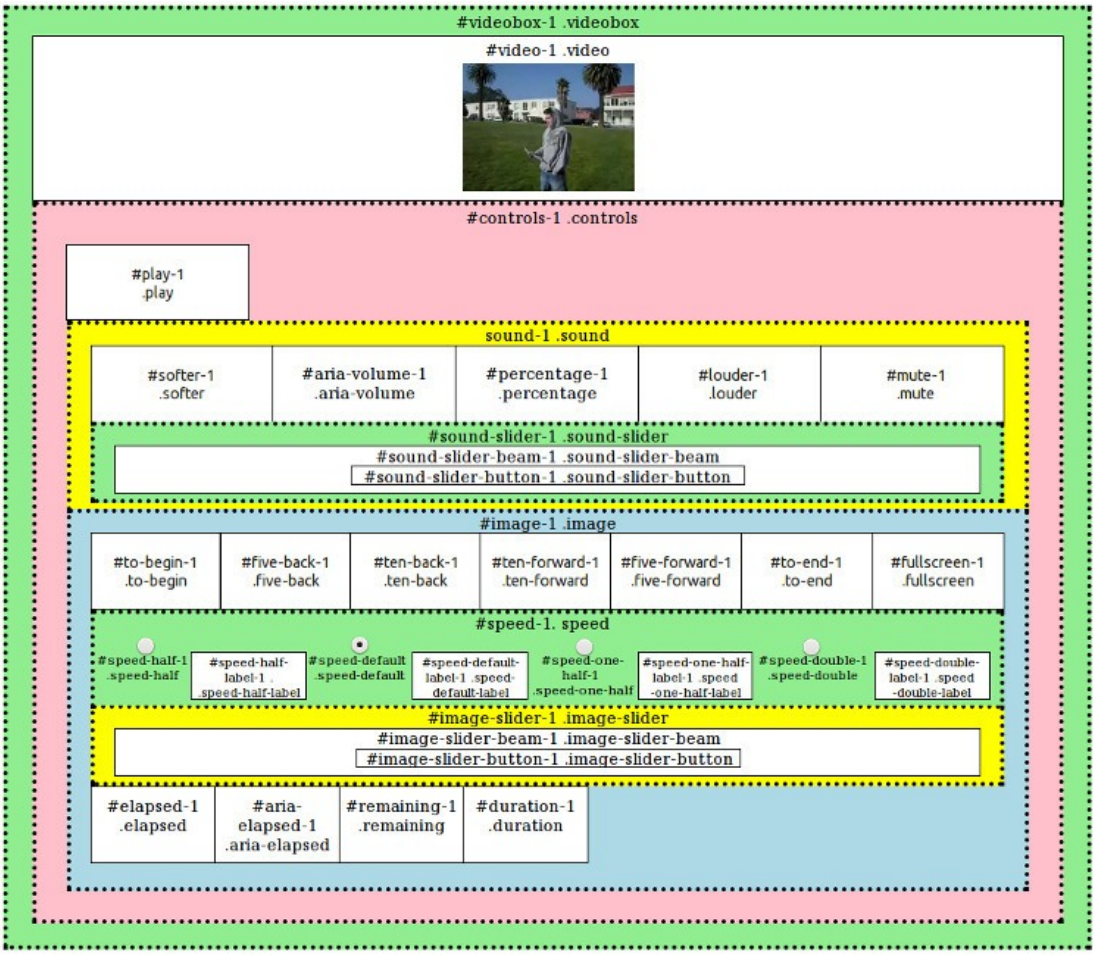
```
<button id="to-end-1" class="to-end">
```

(Er staat nog veel meer in die tag, maar dat heb ik hier even weggelaten.)
De code voor de bij de tweede videospeler horende knop om naar het begin te gaan, ziet er zo uit:

```
<button id="to-begin-1" class="to-end">
```

Ook hier geldt weer dat je de gegenereerde code moet bekijken. Als je gewoon de bron van de pagina bekijkt, zie je alleen de gewone html, zonder wat het script heeft ingevoegd.

Onderstaande afbeelding geeft schematisch de invoeging van bedieningselementen, classes en id's weer. Elk door een stippellijn omsloten vlak is een groep bedieningselementen. Elk door een gewoon lijntje omsloten vlak is een bedieningselement. Dit is de volgorde, zoals de elementen in de html worden ingevoegd. Maar op het scherm kan deze dus volledig worden gewijzigd met behulp van css, zoals bij een aantal videospelers ook is te zien.



Het buitenste door een stippellijn omgeven groene vlak is de , waar de hele handel in staat. Deze moet als class 'videobox' hebben. Dat is de enige class die zelf moet worden ingevoerd. Het script voegt daar een id aan toe. Bij de eerste videospeler is dat 'videobox-1', bij de tweede 'videobox-2', enz.
Een door het script aan te sturen video moet binnen dit element met class="videobox" staan. Het script geeft een class="videobox" en een id="videobox-1" aan de <video>. (En bij de tweede <video> id="videobox-2", enz., maar dat blijf ik niet herhalen.)
De class en id kun je eventueel wijzigen op de manier zoals is beschreven bij ClassId. Dat geldt ook voor alle hieronder genoemde classes en id's.

Het <video>-element zelf krijgt van het script automatisch de class "video". De eerste <video> krijgt een id="video-1", de tweede een id="video-2", enz.

Het roze vak gelijk binnen het buitenste groene vak omsluit alle bedieningselementen. Dit is een , die door het script wordt ingevoegd, met een class="controls" en een id="controls-1". Deze wordt gelijk na het <video>-element ingevoegd. Dit roze vak bevat alle (groepen) bedieningselementen.

De tekst boven de video in de voorbeelden is gewoon in `div.videobox` gezet, boven het <video>-element. De download-links staan ook in `div.videobox`, gewoon onder het <video>-element. Boven en onder <video> kun je gewoon alles neerzetten wat je wilt. Alleen moet je er rekening mee houden, dat de met de bedieningselementen altijd gelijk onder <video> komt te staan. Als er dus iets onder <video> staat, komt dit onder de bedieningselementen te staan. Met behulp van van css kun je het eventueel ergens anders zetten, zoals in een aantal videospelers is gedaan.

SPEEL-PAUZEERKNOP

Deze knop is een <button type=”button”>. Hij staat eenzaam en alleen rechtstreeks in `div.controls`, buiten elke groep. Omdat dit de belangrijkste knop is, leek dit het meest logisch, omdat hij nu makkelijk overal neergezet kan worden. Deze knop krijgt een class=”play” en een id=”play-1”. Als de video niet speelt, wordt aan deze knop nog een extra class toegevoegd: ’not-playing’.

GELUIDSREGELING

Het gele vlak met de gestippelde rand gelijk onder de speel/pauzeerknop is een en bevat alle bedieningselementen voor het volume: zachter en harder, aan en uit, weergave percentage volumesterkte, voorlezen percentage volumesterkte, en de sleepbalk voor het geluid. Deze krijgt een class=”sound” en een id=”sound-1”. Door deze elementen samen in één groep te zetten, is het makkelijker om aan een schermlezer duidelijk te maken waar ze voor dienen. En om in één keer deze hele groep bedieningselementen te passeren.
Binnen deze zitten de volgende bedieningselementen:

ZACHTER

Om het geluid zachter te zetten. Knop <button type=”button”>. Krijgt een class=”softer” en een id=”softer-1”.

VOORLEZEN VOLUMESTERKTE

Deze is alleen voor schermlezers interessant. De enige functie is het voorlezen van de geluidssterkte, als deze is gewijzigd met behulp van een van de andere bedieningselementen. De krijgt een class=”aria-volume” en een id=”aria-volume-1”.

PERCENTAGE VOLUMESTERKTE

 voor het weergeven van de volumesterkte in procenten. Krijgt een class=”percentage” en een id=”percentage-1”.

HARDER

Om het geluid harder te zetten. Knop <button type=”button”>. Krijgt een class=”louder” en een id=”louder-1”.

GELUID AAN-UITKNOP

Om het geluid aan of uit te zetten. Knop <button type=”button”>. Krijgt een class=”mute” en een id=”mute-1”. Als het geluid wordt uitgezet, wordt aan deze knop nog een extra class toegevoegd: ’muted’.

SLEEPBALK VOOR GELUID

De sleepbalk bestaat uit twee delen: de balk en de knop. Deze staan samen in een eigen : op de afbeelding het groene vlak binnen het gele vlak van de

geluidsregeling. Deze krijgt een class=”sound-slider” en een id=”sound-slider-1”.
Binnen deze staat de eigenlijke sleepbalk. Ook dit is een . Deze krijgt een class=”sound-slider-beam” en een id=”sound-slider-beam-1”.
Tenslotte staat binnen deze laatste weer een , die wordt gebruikt voor de knop van de sleepbalk. Door de knop binnen de sleepbalk te zetten, kan het script de juiste positie van de knop binnen de balk berekenen. De voor de knop gebruikte krijgt een class=”sound-slider-button” en een id=”sound-slider-button-1”.

Weergaveregeling

Onder het gele vlak voor de geluidsregeling staat een door een stippellijn omgeven blauw vlak. Hierin staan alle bedieningselementen voor de weergave: Naar begin, Vijf procent terug, Tien seconden terug, Tien seconden verder, Vijf procent vooruit, Naar einde, Fullscreen, snelheidsregeling (met daarin weer 4 radioknoppen met bijbehorende <label>’s), en een sleepbalk voor de weergave.
Door deze elementen samen in één groep te zetten, is het makkelijker om aan een schermlezer duidelijk te maken waar ze voor dienen. En om in één keer deze hele groep bedieningselementen te passeren.
Deze krijgt een class=”image” en een id=”image-1”.
Binnen deze zitten de volgende bedieningselementen:

Naar begin

Om naar het begin van de video te gaan. Knop <button type=”button”>. Deze knop krijgt een class=”to-begin” en een id=”to-begin-1”.

Vijf procent terug

Om vijf procent terug te gaan. Knop <button type=”button”>. Deze knop krijgt een class=”five-back” en een id=”five-back-1”.

Tien seconden terug

Om tien seconden terug te gaan. Knop <button type=”button”>. Deze knop krijgt een class=”ten-back” en een id=”ten-back-1”.

Tien seconden vooruit

Om tien seconden vooruit te gaan. Knop <button type=”button”>. Deze knop krijgt een class=”ten-forward” en een id=”ten-forward-1”.

Vijf procent vooruit

Om vijf procent vooruit te gaan. Knop <button type=”button”>. Deze knop krijgt een class=”five-forward” en een id=”five-forward-1”.

Naar einde

Om naar het einde van de video te gaan. Knop <button type=”button”>. Deze knop krijgt een class=”to-end” en een id=”to-end-1”.

Fullscreen

Om de video fullscreen weer te geven. Knop <button type=”button”>. Deze knop krijgt een class=”fullscreen” en een id=”fullscreen-1”.

Snelheidsregeling

De snelheidsregeling bestaat uit vier radioknoppen met elk een bijbehorend <label>. Deze acht elementen staan samen in een eigen : het groene vlak binnen het blauwe vlak voor de weergaveregeling. Deze krijgt een class=”speed” en een id=”speed-1”.
De radioknoppen hebben een gezamenlijke name: ”knop-snelheid-1”. (En bij de tweede video ”knop-snelheid-2”, enz. Deze name is eventueel te wijzigen bovenin het script. Hoe dat moet, staat bij Name.
De radioknoppen staan buiten de bijbehorende <label>, omdat ze dan makkelijker zijn aan te passen met behulp van css. Koppeling tussen <label> en bijbehorende radioknop gebeurt door middel van het `for`-attribuut. Deze koppeling wordt automatisch door het script aangebracht.
Binnen de snelheidsregeling zitten:

Radioknop halve snelheid

Knop <input type=”radio”>. Deze knop krijgt een class=”speed-half” en een id=”speed-half-1”.

Label bij knop halve snelheid

Krijgt een class=”speed-half-label” en een id=”speed-half-label-1”.

Radioknop normale snelheid

Knop <input type=”radio”>. Deze knop krijgt een class=”speed-default” en een id=”speed-default-1”.

Label bij knop normale snelheid

Krijgt een class=”speed-default-label” en een id=”speed-default-label-1”.

Radioknop anderhalve snelheid

Knop <input type=”radio”>. Deze knop krijgt een class=”speed-one-half” en een id=”speed-one-half-1”.

Label bij knop anderhalve snelheid

Krijgt een class=”speed-one-half-label” en een id=”speed-one-half-label-1”.

Radioknop dubbele snelheid

Knop <input type=”radio”>. Deze knop krijgt een class=”speed-double” en een id=”speed-double-1”.

Label bij knop dubbele snelheid

Krijgt een class=”speed-double-label” en een id=”speed-double-label-1”.

Sleepbalk voor weergave

De sleepbalk bestaat uit twee delen: de balk en de knop. Deze staan in een eigen . Op de afbeelding is dit het gele vlak binnen het blauwe vlak van de weergaveregeling. Deze krijgt een class=”image-slider” en een id=”image-slider-1”.
Binnen deze staat de eigenlijke sleepbalk. Ook dit is een . Deze krijgt een class=”image-slider-beam” en een id=”image-slider-beam-1”.

Tenslotte staat binnen deze weer een , die wordt gebruikt voor de knop van de sleepbalk. Door de knop binnen de sleepbalk te zetten, kan het script de juiste positie van de knop binnen de balk berekenen. De voor de knop gebruikte krijgt een class=”image-slider-button” en een id=”image-slider-button-1”.

Verstreken speelduur

 waarin de verstreken speelduur kan worden weergegeven. Deze krijgt een class=”elapsed” en een id=”elapsed-1”.

Voorlezen verstreken speelduur

Deze is alleen voor schermlezers interessant. De enige functie is het voorlezen van de verstreken speelduur, als deze is gewijzigd met behulp van een van de andere bedieningselementen. De krijgt een class=”aria-elapsed” en een id=”aria-elapsed-1”.

Resterende speelduur

 waarin de resterende speelduur kan worden weergegeven. Deze krijgt een class=”remaining” en een id=”remaining-1”.

Speelduur

 waarin de totale speelduur kan worden weergegeven. Deze krijgt een class=”duration” en een id=”duration-1”.

Door het script gegenereerde waarschuwingen (alerts, pop-ups)

Er kunnen door het script vier waarschuwingen worden gegenereerd: drie standaard JavaScript-alerts en één pop-up.
De tekst van de drie alerts is in het Engels en kan niet worden gewijzigd (althans: niet als je niet enigszins in JavaScript thuis bent). Dit lijkt me geen probleem, omdat deze drie zijn gericht op sitebouwers. Een gewone gebruiker krijgt die dus nooit te zien. En als je als sitebouwer met dit soort voorbeelden wilt werken, is enige kennis van het Engels echt noodzakelijk.
Deze drie meldingen zijn gewone standaard JavaScript-alerts. JavaScript-alerts zijn berucht lelijk. Vooral de alert op Windows 8 is overduidelijk bedacht door de tweelingzus van kapitein Haddock, nadat deze op een woeste zee de voltallige voorraad whisky had opgedronken.
Ook dit lijkt me geen probleem, want een gewone bezoeker krijgt dit lelijks nooit te zien.

De vierde melding kan wel door bezoekers worden gezien. De tekst is Nederlandstalig en kan bovenin het script worden aangepast. Het uiterlijk kan volledig met css worden aangepast. De melding is desgewenst ook simpel helemaal uit te schakelen. Hoe dat allemaal werkt, staat bij Speelduur nog onbekend.

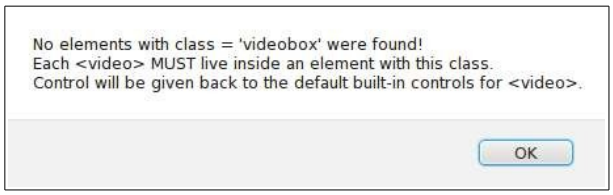
Geen element aanwezig met class=”videobox”

Een <video> moet binnen een element met class=”videobox” staan, anders wordt het door het script genegeerd. Als er geen enkel element met deze class aanwezig is, is er waarschijnlijk sprake van een fout. Want waarom zou iemand zo’n script linken en het vervolgens niet gebruiken?
De tekst van deze melding luidt:

```
alert("No elements with class = '" +
      wrapperDivClassId + "' were found! \nEach
      <video> MUST live inside a div <div> with this
```

```
            class. \nControl will be given back to the
            default built-in controls for <video>.");
```

Binnen de alert() staat voornamelijk gewone tekst. Er staan twee vreemdigheden in:
* Het woord `wrapperDivClassId` is een zogenaamde 'variabele'. Dat wil zeggen dat de inhoud ervan kan variëren, net zoals de inhoud van je portemonnee. De portemonnee blijft 'portemonnee' heten, of er nou bankbiljetten of munten in zitten. En op driekwart van de maand, als het ding leeg is, heet het nog steeds portemonnee. De enkele en dubbele aanhalingstekens eromheen geven de scheiding tussen letterlijke tekst en deze variabele aan, zodat het script niet zodanig de hik krijgt dat het er acuut in blijft.
`wrapperDivClassId` wordt vervangen door de naam van de class, die wordt gebruikt voor het element waar de <video>'s in staan. Als je die naam niet hebt aangepast, is dat 'videobox'.
* Met `\n` geef je in JavaScript aan dat er een nieuwe regel moet beginnen. Het woord 'Each' komt dus op een nieuwe regel te staan.



De alert zoals die er in Firefox uitziet.
`wrapperDivClassId` is vervangen door 'videobox' en
'Each' staat op een nieuwe regel.

Omdat er verder niets te doen is voor het script, stopt dit ermee. Als er toch video's op de pagina staan, worden die aangestuurd door de in de browser ingebouwde standaardvideospeler. Het script kan pas draaien, als deze fout is hersteld.

Een id, class, name of aria-label ontbreekt of is niet in orde

Voor een goede werking van het script is het noodzakelijk, dat elk bedieningselement een class en/of id krijgt. Radioknoppen moeten een gezamenlijke name hebben. Voor de toegankelijkheid moet een klein aantal elementen een `aria-label` hebben.
Als je niets hebt gewijzigd aan het script, zal dit allemaal goed werken. Maar als je bovenin het script een class, id, name of aria-label hebt verandert, kan er iets zijn misgegaan. Overigens is dit een tamelijk oppervlakkige controle. Er wordt alleen gecontroleerd of het gegeven aanwezig is, en of dit de vorm van een karakterstring (tekst) heeft. Als dit niet zo is, wordt de standaardnaam van id, class, name of aria-label gebruikt. (Dat zijn de namen die ook in de voorbeelden en in deze uitleg zijn gebruikt.)
Zoiets kan bijvoorbeeld gebeuren, als je de naam van een id een regel te laag invult, achter een ander gegeven. Omdat dit hoogst verwarrend kan zijn – jij denkt dat de id 'kolereherrie' is en het script heeft het welopgevoede 'louder' ingevuld –, krijg je hier een melding van:

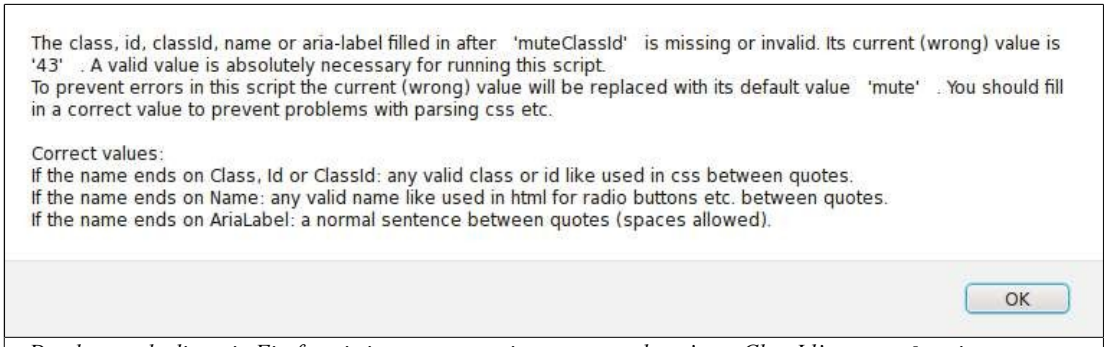
```
    alert("The class, id, classId, name or aria-label
        filled in after   '" + varName + "'   is
        missing or invalid. Its current (wrong) value
        is   '" + varValue + "'   . A valid value is
        absolutely necessary for running this
        script.\nTo prevent errors in this script the
        current (wrong) value will be replaced with its
        default value   '" + defaultValue + "'   . You
```

```
should fill in a correct value to prevent
problems with parsing css etc.\n\nCorrect
values:\nIf the name ends on Class, Id or
ClassId: any valid class or id like used in css
between quotes.\nIf the name ends on Name: any
valid name like used in html for radio buttons
etc. between quotes.\nIf the name ends on
AriaLabel: a normal sentence between quotes
(spaces allowed).");
```

Binnen de alert() staat voornamelijk gewone tekst, maar er staan ook wat vreemdigheden in:

* 'class, id, classId, name or aria-label'. Bovenin het script kunnen dingen als classes en id's worden aangepast. Sommige daarvan zijn onmisbaar voor een goede werking van het script. Als een van deze dingen ontbreekt of verkeerd in elkaar zit (een typefout is snel gemaakt), verschijnt deze waarschuwing.
* Het woord `varName` is een zogenaamde 'variabele'. Dat wil zeggen dat de inhoud ervan kan variëren, net zoals de inhoud van je portemonnee. De portemonnee blijft 'portemonnee' heten, of er nou bankbiljetten of munten in zitten. En op driekwart van de maand, als het ding leeg is, heet het nog steeds portemonnee. De enkele en dubbele aanhalingstekens eromheen geven de scheiding tussen letterlijke tekst en deze variabele aan, zodat het script niet zodanig de hik krijgt dat het er acuut in blijft. `varName` wordt vervangen door de naam van de het eerste woord van de regel, waar een waarde mist of waar een verkeerde waarde is opgegeven. Door te zoeken op de inhoud van `varName`, kun je die regel snel vinden in het script.
* Voor `varValue` geldt hetzelfde als hierboven, alleen wordt deze variabele vervangen door de verkeerd ingevulde waarde. Als er per ongeluk niets is ingevuld, zie je hier alleen `""`.
* Ook voor de variabele `defaultValue` geldt hetzelfde als hierboven, maar deze wordt vervangen door de standaardwaarde, zodat het script kan worden uitgevoerd. Als je de gegenereerde code bekijkt, is dit de waarde die je in de code ziet staan bij de betreffende id, class, name of aria-label.
* Met `\n` geef je in JavaScript aan dat er een nieuwe regel moet beginnen. `\nTo` wil dus alleen maar zeggen, dat het woordje 'To' op een nieuwe regel komt te staan.

Achter `muteClassId` bovenin het script moet een naam voor een class (en id) staan, iets als `muteClassId = "mute"`. Als ik in plaats daarvan foutief een getal invul, iets als `muteClassId = 43`, verschijnt het volgende venstertje:



De alert zoals die er in Firefox uitziet. `varName` is vervangen door 'muteClassId', `varValue` is vervangen door (het foutieve) 43, en `defaultValue` is vervangen door de standaardwaarde 'mute'. 'To' en drie keer het woordje 'if' staan op een nieuwe regel. 'Correct' staat ook op een nieuwe regel, maar omdat daar twee keer `\n` voor staat, staat daar nog een lege regel boven.

Omdat dit gerepareerd kan worden, wordt het script verder gewoon uitgevoerd. Maar wel met de standaardwaarde van het verkeerd ingevulde of missende gegeven, dus als je dat verkeerd ingevulde gegeven bijvoorbeeld in een selector in de css gebruikt, kan dat wel fouten opleveren.

Er is een element met class=”videobox”, waarin geen <video> staat

Het script gaat op zoek naar elementen met class=”videobox”. Video’s binnen een element met die class, worden door het script aangestuurd. Als er een element met class=”videobox” wordt gevonden, waarin geen <video> staat, is het script vreselijk teleurgesteld. Van de weeromstuit stopt het ermee. Zou ik ook doen. Eerst blij maken en dan op het laatste moment je terugtrekken, is stijlloos.
(Bovenin het script kan ’videobox’ desgewenst in iets anders worden veranderd. Hoe dat kan, staat bij ClassId.)
De volgende melding verschijnt:

```
alert("The red bordered element with id '" +
      wrapper.id + "' has no <video>-element inside.
      Because of this error this script cannot run.
      Inside each element with class '" +
      wrapperDivClassId + "' MUST a <video>-element
      be present. This script will terminate now.
      Videos still can be played with the default
      built-in videoplayer.");
```

Binnen de alert() staat voornamelijk gewone tekst. Er staan twee vreemdigheden in:
* Het woord `wrapper.id`: Dit wordt vervangen door de id van het element, waarin de <video> mist. Dit is, als je ’videobox’ niet hebt veranderd bovenin het script, ’videobox-’ gevolgd door het volgnummer van het element, waarin de <video> mist. Op deze manier is dat element snel te vinden aan de hand van de id. De enkele en dubbele aanhalingstekens rondom `wrapper.id` geven de scheiding tussen letterlijke tekst en `wrapper.id` aan, zodat het script niet zodanig de hik krijgt dat het er acuut in blijft.
* `wrapperDivClassId` wordt vervangen door de class van het element, waarin een <video> moet staan. Als je dat niet hebt vervangen bovenin het script, is dat ’videobox’.

Voor de duidelijkheid wordt rondom het element met een missende <video> een rode lijn getekend. Als binnen het eerste element met class=”videobox” de <video> mist, verschijnt het volgende venstertje:

De alert zoals die er in Firefox uitziet. `wrapper.id` is vervangen door ’videobox-1’ en `wrapperDivClassId` is vervangen door ’videobox’. Als de alert wordt gesloten, is de volledige rode lijn rondom de eerste met class=”videobox” te zien.

Na sluiting van de alert, is de volledige rode lijn te zien.
Omdat het script voor het bepalen van de volgnummers van de id’s binnen elk element met class=”videobox”een <video> nodig heeft, stopt het script ermee. Het script kan pas worden uitgevoerd, als deze fout is hersteld.

Speelduur nog onbekend

Dit is een pop-up, waarvan uiterlijk en inhoud zijn aan te passen.

Reden van de melding

Een aantal functies van de videospeler werkt pas goed, als de speelduur van de video bekend is. Het gaat hierbij om de knoppen Naar begin, Naar einde, Vijf procent vooruit, Vijf procent achteruit, Tien seconden vooruit, Tien seconden achteruit, en Fullscreen afspelen, plus nog de sleepbalk voor weergave.

Als je in de html <video> `preload="none"` gebruikt, worden helemaal geen gegevens opgevraagd, totdat de video daadwerkelijk wordt afgespeeld. Het andere uiterste is `preload="auto"`, waarbij de hele video alvast wordt gedownload, voordat hij daadwerkelijk wordt afgespeeld.

Ertussenin zit het in het voorbeeld gebruikte `preload="metadata"`, waarbij de video zelf pas wordt gedownload, als hij wordt afgespeeld. Gegevens als speelduur worden echter gelijk bij het openen van de pagina opgevraagd. Hierdoor blijft het dataverbruik laag, wat van belang is voor bijvoorbeeld mobiele verbindingen, maar zijn speelduur e.d. wel al bekend bij het openen van de pagina.

`preload` is niet meer dan een suggestie voor de browser, deze mag hiervan afwijken. Op iOS negeerde Apple `preload` volledig en werd helemaal niets opgevraagd, totdat de video daadwerkelijk werd afgespeeld. De speelduur was dus onbekend. (Het lijkt erop dat Apple sinds heel kort wel metadata opvraagt, maar in de officiële documentatie staat nog dat ze dat niet doen.)

Sommige browsers, met name Internet Explorer 9 en oudere versies van Android browser, hebben moeite met het opvragen van zes speelduren (voor elke video één). In deze browsers zijn niet alle tijden bekend, en als je pech hebt geen enkele.

(Zolang de speelduur onbekend is, heeft het attribuut `data-duration` bij `div.videobox` een waarde van 'unknown'. Door dit in een selector op te nemen, kan met behulp van css het uiterlijk worden aangepast, zoals beschreven bij data-duration.)

Uiterlijk en code van de melding

Als de speelduur nog niet bekend is, zijn de hierboven genoemde functies nog niet goed bruikbaar en verschijnt een melding met uitleg, als de bezoeker toch een van deze functies probeert te gebruiken:

(Deze melding verschijnt alleen, als je de knoppen of balk echt bedient. Als je alleen met Tab of Shift+Tab naar de vorige of volgende knop of balk gaat, wordt dit genegeerd.)

Het uiterlijk van deze melding is volledig aangepast met behulp van css. Het script voegt alleen een paar elementen en de tekst in. Als je de gegenereerde code bekijkt op het moment dat deze melding is geopend, zie je de volgende html:

```
<div id="duration-div" role="dialog" aria-
     label="Functie nog niet beschikbaar" aria-
     describedby="duration-par">.
     <p id="duration-par" tabindex="-1">
          Deze functie is pas beschikbaar, als
          de speelduur van de video bekend
          is.<br>Om een of andere reden is de
          speelduur van de video nog niet
          bekend. Mogelijk is de speelduur nog
          helemaal niet opgevraagd. Sommige
          browsers vragen deze pas op, als de
          video wordt afgespeeld. Mogelijk is
          de verbinding met de server niet
          optimaal. Of er is 'n andere reden.
          Meestal komt deze functie alsnog
          beschikbaar, als de video wordt
          gestart. Na sluiting van deze melding
          krijgt de Speelknop van de
          bijbehorende video focus.
          <button id="duration-button">Sluit
               melding</button>
     </p>
</div>
```

De melding staat in z'n geheel binnen een . Deze wordt door het script in de html ingevoegd op het moment dat dat nodig is, en weer

verwijderd als de melding niet meer nodig is. De melding wordt ingevoegd als laatste kind van de , waarbinnen de bedieningselementen van de eerste video staan: `div#controls-1`.
Dit is een veilige plaats om in te voegen, omdat `div#controls-1` volledig door het script wordt aangemaakt. Het invoegen van de melding is feitelijk hoe dan ook geen probleem, maar het verwijderen kan wel ’n kleine ramp aanrichten. Als je de bijvoorbeeld als eerste kind van <body> zou invoegen, zou een ander script of wat dan ook een eerste kind bij <body> kunnen invoegen, terwijl de melding is geopend. Als je dan de melding (het eerste kind van <body>) weer wilt verwijderen, zou je in plaats daarvan ’n ander element verwijderen. Je moet er toch niet aan denken dat dat het enige exemplaar van ’n wachtwoord voor de noodstop van Borssele is, of de bevestiging dat je de Kanjerprijs hebt gewonnen...
Door de melding in te voegen in en later weer te verwijderen uit de volledig door het script aangemaakte `div#controls-1`, is er geen enkel risico dat per ongeluk ’n verkeerd element wordt verwijderd.

Aanpassen van uiterlijk, id, tekst, e.d.

Een groot deel van bovenstaande melding is aan te passen.
Alle elementen van de melding (een , een <p> en een <button>) hebben een id, waardoor het uiterlijk met doodnormale css is aan te passen.

Voor de eigenlijke tekst staan een en een <p>:

```
<div id="duration-div" role="dialog" aria-
     label="Functie nog niet beschikbaar" aria-
     describedby="duration-par">
     <p id="duration-par" tabindex="-1">
```

Een aantal van de attributen e.d. van de en de <p> zijn te wijzigen. Deze te wijzigen zaken staan allemaal bovenin het script.
Hoe je de id bij de of bij de <p> kunt wijzigen, staat bij Id.
`role="dialog"` is een zogenaamde WAI-ARIA-code, die niet gewijzigd kan worden. Dit maakt aan schermlezers duidelijk dat er een venstertje wordt geopend. Er is meer over te lezen bij WAI-ARIA-codes binnen de videospeler.
`aria-label="Functie nog niet beschikbaar"` is ook een voor schermlezers gebruikte WAI-ARIA-code. Wat dit precies is, is te lezen bij WAI-ARIA-codes binnen de videospeler. De tekst hiervan is aan te passen. Hoe dat kan, staat bij AriaLabel.
`aria-describedby="duration-par"` kan ook niet worden gewijzigd. Het geeft aan, dat de tekst van het venstertje te vinden is in het element met id=”duration-par”, in dit geval de id van de <p> waarbinnen de tekst van de melding staat. (Als je die id om een of andere reden verandert, wordt die door het script automatisch ook hier veranderd. Indirect is ’duration-par’ dus wel te wijzigen.)
`tabindex="-1"` tenslotte is nodig om de <p> focus te kunnen geven, zodat de melding niet alleen door aanraken of klikken, maar ook door middel van het toetsenbord kan worden gesloten. Dit is wat belangrijk is voor mensen die de muis niet kunnen of willen gebruiken. Wat focus en tabindex precies zijn, vind je bij Focus en Tabindex.

Binnen de <p> staat de eigenlijke tekst van de melding. Deze tekst kan bovenin het script worden aangepast. Enigszins bovenin het script staat `var durationUnknownText=` , gevolgd door aanhalingstekens, de tekst en weer aanhalingstekens. De tekst tussen de aanhalingstekens is de tekst die verschijnt, als de melding wordt geopend. Deze tekst kan volledig worden gewijzigd:

```
var durationUnknownText=" (…) tekst (…) ",
```

De dubbele aanhalingstekens voor en na de tekst moeten blijven staan, en de komma aan het eind van de regel ook. Door de aanhalingstekens weet de browser, waar de letterlijke tekst begint en eindigt. De komma vertelt dat er nog veel meer volgt (wat hier verder niet van belang is).
Binnen de tekst kunnen ook nieuwe regels, vet, cursief, bijzondere tekens, enz., worden aangebracht, zoals in alle karakterstrings. Dit wordt beschreven bij Text.

Na de <p> met de tekst volgt nog een <button>:

```
<button id="duration-button">Sluit
    melding</button>
```

De id hiervan kan worden gewijzigd. Hoe dat moet, staat bij Id.
De tekst 'Sluit melding' verschijnt als tekst op de <button>. Ook deze tekst kan worden gewijzigd, zoals is te lezen bij Text.
De <button> moet altijd 'n tekst hebben, omdat anders volstrekt onduidelijk is, waar die <button> voor dient. Het best kun je zelf 'n duidelijke tekst opgeven. Als er geen tekst is, wordt door het script 'OK' als tekst voor de <button> gebruikt.

Uitschakelen van de melding

Als je de hele melding wilt uitschakelen, haal je gewoon de hele tekst weg bij `durationUnknownText`. Het script controleert of er tekst voor de melding is. Als er geen tekst is, wordt de hele melding gewoon niet getoond. Ook de <button> die op de <p> met de tekst volgt, komt dan niet te voorschijn. Dat ziet er zo uit in het script:

```
var durationUnknownText="",
```

Meer is er niet nodig om de hele melding uit te schakelen.

Openen en sluiten van de melding

De melding wordt geopend, als de speelduur van de video nog onbekend is en een van de knoppen voor vooruit of achteruit, de sleepbalk voor afspelen of de knop Fullscreen wordt gebruikt.
Het openen van de melding is niet echt heel ingewikkeld. Maar voorkomen dat de melding te vroeg of te laat sluit, of dat 'n gebruiker per ongeluk de nog geopende melding verlaat, is 'n heel stuk ingewikkelder.
Als de melding wordt geopend, wordt de focus op de <p> met de tekst gezet. (Meer over focus bij Focus.) Liever had ik de focus op <button> gezet, maar dat kan niet, omdat de tekst van de melding dan niet in elke schermlezer wordt voorgelezen.

Een gebruikelijke manier om dit soort meldingen te sluiten is de Escape-toets. Deze werkt ook hier: als je op Escape drukt, sluit de melding.

Een toetsenbordgebruiker zou met Shift+Tab naar het vorige element kunnen gaan. Dan blijft de melding geopend, terwijl deze geen nut meer heeft. Daarom sluit de melding ook, zodra de Shift-toets wordt ingedrukt.

Een tweede manier om de melding te sluiten is het gebruik van de sluitknop. Bij klikken op de knop 'Sluit melding', sluit (uiteraard) de melding.
Als na opening van de melding één keer op de Tab-toets wordt gedrukt, krijgt de sluitknop focus. Waarna de knop ook gewoon met het toetsenbord is te bedienen: Enter en spatiebalk sluiten de melding. Daarnaast wordt de melding ook met Escape gesloten, en ook als nogmaals de Tab-toets wordt ingedrukt.

Door de melding op al deze manieren te laten sluiten, wordt voorkomen dat schermlezers of gebruikers van het toetsenbord de melding kunnen verlaten, terwijl deze nog is geopend. Zou dat gebeuren, dan kan de melding alleen nog met de muis worden gesloten, en dat is nou juist vaak een probleem voor deze gebruikers.

Als de melding wordt gesloten, wordt de focus automatisch op de Speel-pauzeerknop de video die bij het gebruikte bedieningselement hoort gezet. Een simpele klik, aanraking of toetsindruk start dan gelijk de juiste video.

Wijzigingen aanbrengen in het script

Dit hoofdstukje gaat over de grote, aparte scripts. Wijzigingen aanbrengen in de kleine scriptjes onderaan de html-bestanden staat bij Het JavaScript onderaan de html-bestanden.
Een aantal dingen, zoals namen van classes en id's, is eenvoudig te wijzigen. Al dit soort eenvoudig te wijzigen dingen staan bovenaan in het script.
Een typefout is snel gemaakt. JavaScript is, net zoals alle programmeertalen, uiterst gevoelig voor fouten. Veel meer dan css of html. **MAAK DAAROM ALTIJD EERST EEN BACK-UP!!!!!**, voordat je wat dan ook gaat veranderen. Als je per ongeluk één kommaatje te veel verandert, kun je anders tijdenlang zoeken naar wat er mis is. Als je een back-up hebt gemaakt, kun je altijd terug naar het origineel.
Feitelijk moet je zelfs een hele reeks back-ups maken. Als je iets hebt veranderd en alles werkt goed, maak dan gelijk een nieuwe back-up met die verandering. Blijkt later dat er toch iets niet goed werkt, dan kun je eventueel 'n paar back-ups teruggaan, tot de laatste waar alles goed werkte. Je bent dan niet al je veranderingen kwijt, wat wel zo zou zijn, als je alleen van het originele script een back-up hebt gemaakt.
Een reden om iets te wijzigen zou kunnen zijn, dat het script een class of id toewijst, die al in gebruik is. Of dat je 'n bepaalde tekst wilt aanpassen. Of dat je de beroepsopleiding voor chagrijnen succesvol hebt afgesloten en het woord 'play' dus niet meer mag gebruiken als naam voor een class. Of dat de inhoud van een title je niet bevalt.
Uiteraard moet je in de selectors in de css ook de namen van classes en id's aanpassen, als je die in het script verandert.
Achter elke ingang hieronder staat, achter het woordje 'standaard', wat het script gebruikt, als je zelf niets wijzigt. De te wijzigen teksten e.d. staan steeds achter een zogenaamde 'variabele'. Deze variabelen eindigen op een bepaalde uitgang, die te maken heeft met waar ze voor worden gebruikt. Hieronder staat groepsgewijs, waar ze voor dienen en hoe je ze kunt wijzigen.

AriaLabel (eindigend op 'AriaLabel', voor aria-label):

`aria-label` is één van de speciale codes voor schermlezers. Afhankelijk van de instellingen van de schermlezer wordt de tekst hiervan voorgelezen. Het is vergelijkbaar met de title die bij hoveren over een knop verschijnt.

Als je geen aria-label gebruikt, is er een kans dat een schermlezer niet voorleest wat de werking van een knop e.d. is. Afhankelijk van de instellingen kan de schermlezer ook een title gebruiken om voor te lezen, maar zeker is dat niet. En je moet zeker niet aria-label én title beide verwijderen.
Een aria-label kan ook belangrijk zijn voor mensen die niet volledig blind zijn, maar alleen slecht zien. En je hebt ook bijvoorbeeld muziekvideo's, waarvan je het geluid prima zonder je ogen kunt horen.

Het is belangrijk dat de opgegeven tekst tussen aanhalingstekens staat. Aan het eind van de regel moet een komma staan. (Door de aanhalingstekens weet het script dat het hier om een letterlijke tekst gaat, de komma laat weten dat er nog meer komt.)
Waar de tekst voor het aria-label staat, is te herkennen aan het laatste deel van het woord voor het isgelijkteken (de 'variabele', waarin de naam wordt opgeslagen): deze eindigt altijd op 'AriaLabel'.
Als je 'n aria-label niet wilt gebruiken, vul je gewoon geen tekst in, maar alleen "", bijvoorbeeld:

```
playAriaLabel = "",
```

Hieronder staat in alfabetische volgorde een rijtje met in het script aanwezige aria-labels. Deze zijn allemaal te veranderen op bovenbeschreven wijze. Achter 'standaard' staat de naam die het script gebruikt, als je niets verandert.

`chooseDivAriaLabel`
: standaard: 'Kiezen tussen standaard- of aangepaste videospelers'. Aria-label voor waarbinnen de keuze voor standaard- of aangepaste videospeler staat.

`durationAriaLabel`
: standaard: 'Speelduur'. Aria-label voor om speelduur weer te geven.

`durationDivAriaLabel`
: standaard: 'Functie nog niet beschikbaar.' Aria-label voor met melding dat speelduur nog onbekend is.

`elapsedAriaLabel`
: standaard: 'Verstreken'. Aria-label voor om verstreken speelduur weer te geven.

`fiveBackAriaLabel`
: standaard: 'Vijf procent terug'. Aria-label voor knop om vijf procent terug te gaan.

`fiveForwardAriaLabel`
: standaard: 'Vijf procent verder'. Aria-label voor knop om vijf procent vooruit te gaan.

`fullScreenAriaLabel`
: standaard: 'Volledig scherm'. Aria-label voor knop om fullscreen af te spelen.

`imageSliderBeamAriaLabel`
: standaard: 'Sleepbalk afspelen'. Aria-label voor sleepbalk voor afspelen. (De knop van deze sleepbalk is verborgen voor schermlezers met `aria-hidden`, dus deze krijgt geen aria-label.)

`louderAriaLabel`
: standaard: 'Harder'. Aria-label voor knop om geluid harder te zetten.

`muteAriaLabel`
: standaard: ’Geluid uit’. Aria-label Aan-uitknop geluid als het geluid aan staat. (Wordt alleen gebruikt als geluid aan staat, dit wordt geregeld door het script.)

`pauseAriaLabel`
: standaard: ’Pauzeren’. Aria-label voor Speel-pauzeerknop als de video speelt. (Wordt alleen gebruikt als video speelt, dit wordt geregeld door het script.)

`percentageAriaLabel`
: standaard: ’Geluidssterkte’. Aria-label voor om geluidssterkte weer te geven.

`playAriaLabel`
: standaard: ’Afspelen’. Aria-label voor Speel-pauzeerknop als de video niet speelt. (Wordt alleen gebruikt als video niet speelt, dit wordt geregeld door het script.)

`remainingAriaLabel`
: standaard: ’Resterend’. Aria-label voor om resterende speelduur weer te geven.

`softerAriaLabel`
: standaard: ’Zachter’. Aria-label voor knop om geluid zachter te zetten.

`soundSliderBeamAriaLabel`
: standaard: ’Sleepbalk volume’. Aria-label voor sleepbalk voor geluid. (De knop van deze sleepbalk is verborgen voor schermlezers met `aria-hidden`, dus deze krijgt geen aria-label.)

`speedAriaLabel`
: standaard: ’Snelheid afspelen instellen’. Aria-label voor waarbinnen snelheidsregeling staat.

`speedDefaultAriaLabel`
: standaard: ’Normale snelheid’. Aria-label voor knop voor normale snelheid.

`speedDoubleAriaLabel`
: standaard: ’Dubbele snelheid’. Aria-label voor knop voor dubbele snelheid.

`speedHalfAriaLabel`
: standaard: ’Halve snelheid’. Aria-label voor knop voor halve snelheid.

`speedOneHalfAriaLabel`
: standaard: ’Anderhalve snelheid’. Aria-label voor knop voor anderhalve snelheid.

`tenBackAriaLabel`
: standaard: ’Tien seconden terug’. Aria-label voor knop om tien seconden terug te gaan.

`tenForwardAriaLabel`
: standaard: ’Tien seconden verder’. Aria-label voor knop om tien seconden vooruit te gaan.

`toBeginAriaLabel`
: standaard: ’Naar begin video’. Aria-label voor knop om naar begin video te gaan.

`toEndAriaLabel`
: standaard: ’Naar einde video’. Aria-label voor knop om naar einde video te gaan.

`unmuteAriaLabel`

standaard: ’Geluid aan’. Aria-label Aan-uitknop geluid als het geluid uit staat. (Wordt alleen gebruikt als geluid uit staat, dit wordt geregeld door het script.)

`videoAriaLabel`

standaard: ’Video nummer’ plus volgnummer van video, bijvoorbeeld ’Video nummer 1’. Dit wordt alleen gebruikt als er in de html bij <video> geen `aria-label`, `aria-describedby` of `aria-labelledby` staat. Meer hierover bij aria-describedby.
Dit laatste aria-label is onmisbaar voor een goede werking van het script. Als deze onbruikbaar is, wordt de standaardwaarde gebruikt, zoals beschreven bij Een id, class, name of aria-label ontbreekt of is niet in orde.
De standaardwaarde is echt ’n soort noodsprong, omdat ’video nummer 1’ nou niet echt informatief is. Veel beter is het om in de html een duidelijke titel of omschrijving van de video op te nemen met behulp van `aria-label`, `aria-describedby` of `aria-labelledby`.

Class (eindigend op ’Class’, voor alleen class):

De Speel-pauzeerknop en de Geluid aan-uitknop zijn feitelijk aan-uitschakelaars. De video speelt óf speelt niet. Het geluid staat aan óf uit.
Om met css in te kunnen spelen op de stand van deze knoppen, krijgen ze een extra class. De Speel-pauzeerknop als de video niet afspeelt, en de Aan-uitknop voor geluid als het geluid uit staat. Anders dan hieronder bij de variabelen die eindigen op ’ClassId’, worden deze dus niet gebruikt voor het aanmaken van id’s. (Als een id is nodig is, kun je de id van de Speel-pauzeerknop of de Geluid aan-uitknop combineren met deze class.)

In principe zijn deze classes ook elders op de pagina te gebruiken, omdat het script alleen kijkt naar de <button>’s spelen/pauzeer en geluid aan/uit. Maar ik zou dat niet doen, want het wordt er bepaald niet duidelijker op, als je alles door elkaar heen overal neer gaat zetten.

Het is belangrijk dat de opgegeven naam tussen aanhalingstekens staat. Aan het eind van de regel moet een komma staan. (Door de aanhalingstekens weet het script dat het hier om een letterlijke tekst gaat, de komma laat weten dat er nog meer komt.)
Waar de naam voor de classes wordt opgegeven, is te herkennen aan het laatste deel van het woord voor het isgelijkteken (de ’variabele’, waarin de naam wordt opgeslagen): deze eindigt altijd op ’Class’.
Alle namen van variabelen die eindigen op ’Class’ zijn onmisbaar voor een goede werking van het script. Als er eentje onbruikbaar is, wordt de standaardwaarde gebruikt, zoals beschreven bij Een id, class, name of aria-label ontbreekt of is niet in orde..

`muteClass`

standaard: ’muted’. Class die wordt toegevoegd aan de Aan-uitknop voor geluid, als het geluid uit staat. De id en classes zien er dan als volgt uit:

`<button id="mute-1" class="mute muted"` (... rest attributen ...) >

`notPlayingClass`

standaard: ’not-playing’. Class die wordt toegevoegd aan de Speel-pauzeerknop als de video niet speelt. De id en classes zien er dan als volgt uit:

```
<button id="play-1" class="play not-playing"
    type="button" (... rest attributen …) >
```

Als je een van deze classes wilt vervangen, vervang je gewoon het deel achter het isgelijkteken:

```
notPlayingClass="bekijk-het-maar",
```

Als de video niet speelt, wordt nu niet 'not-playing' toegevoegd als extra class, maar 'bekijk-het-maar'.

ClassId (eindigend op 'ClassId', voor class én id):

Elk door het script ingevoegd element heeft een class en een id.
(De aan-uitknop voor geluid en de Speel-pauzeerknop hebben soms twee classes, die tweede class staat iets hierboven bij Class. De waarin de keuze tussen standaard- en aangepaste videospeler staat, heeft alleen een id, die staat hieronder bij Id. Hetzelfde geldt voor de , de <p> en de <button> in de melding die je krijgt, als je bepaalde knoppen gebruikt en de speelduur nog onbekend is.)
De naam van deze classes en id's wordt bovenaan in het script opgegeven. Een van de bovenste is die voor het <video>-element:

```
videoClassId="video",
```

'video' is de class, die bij elke <video> hetzelfde is. Bij de eerste <video> met class='video' wordt '-1' achter 'video' gezet. Dit wordt gebruikt als id: 'video-1'. Bij de tweede <video> met class='video' wordt er '-2' achter gezet, waardoor deze <video> id='video-2' krijgt. Elke volgende videospeler krijgt 'n volgend volgnummer. Bij de zesde videospeler eindigen alle id's dus op '-6'.
Ditzelfde patroon wordt voor alle bedieningselementen gevolgd. Alle knoppen voor harder hebben een class='louder'. De eerste knop heeft daarnaast een id='louder-1', de tweede een id='louder-2', enz.
Als je in het script 'video' verandert in 'joechei', krijgen de <video>-elementen een class="joechei" en een id="joechei-1", "joechei-2", enz. Het enige wat je hiervoor hoeft te doen, is het veranderen van wat tussen de aanhalingstekens staat:

```
videoClassId="joechei",
```

Als je bij de knoppen voor harder de class wilt veranderen in 'niet-zo-hard' (en dus de id's in 'niet-zo-hard-1', 'niet-zo-hard-2' , enz. verander je gewoon het iets lager in het script staande `louderClassId="louder",` in

```
louderClassId="niet-zo-hard",
```

De id's van de knoppen voor harder worden nu 'niet-zo-hard-1', 'niet-zo-hard-2', enz.
(`chooseLabelClassId`, `chooseParClassId` en `chooseRadioClassId` worden gebruikt om te kiezen voor de standaard- of aangepaste videospelers. Omdat het hier maar om een keuze uit twee mogelijkheden gaat, worden hier ook maar twee id's gemaakt.)

Het is belangrijk dat de opgegeven naam tussen aanhalingstekens staat. Aan het eind van de regel moet een komma staan. (Door de aanhalingstekens weet het script dat het hier om een letterlijke tekst gaat, de komma laat weten dat er nog meer komt.)
Waar de naam voor de classes en id's wordt opgegeven, is te herkennen aan het laatste deel van het woord voor het isgelijkteken (de 'variabele', waarin de naam wordt opgeslagen): deze eindigt altijd op 'ClassId'.
(Er zijn ook variabelen die alleen op 'Id' eindigen in plaats van op ClassId, maar dat zijn aparte gevallen, die hieronder worden behandeld bij Id. Hetzelfde geldt voor variabelen die alleen op 'Class' eindigen, die staan hierboven bij Class.)

In principe zijn deze classes ook elders op de pagina te gebruiken, omdat het script controleert of ze wel bij een bedieningselement horen (wel binnen `div.controls` staan). Maar ik zou dat niet doen, wat het wordt er bepaald niet duidelijker op, als je alles door elkaar heen overal neer gaat zetten. (Id's mogen maar één keer voorkomen op een pagina, dus die zijn niet elders op de pagina te gebruiken.)

Alle namen van variabelen die eindigen op 'ClassId' zijn onmisbaar voor een goede werking van het script. Als er eentje onbruikbaar is, wordt de standaardwaarde gebruikt, zoals beschreven bij Een id, class, name of aria-label ontbreekt of is niet in orde..
Hieronder staat in alfabetische volgorde een rijtje met in het script opgegeven classen en id's. Deze zijn allemaal te veranderen op bovenbeschreven wijze. Achter 'standaard' staat de naam die het script gebruikt, als je niets verandert.

`ariaElapsedClassId`
: standaard: 'aria-elapsed'. Class en id's voor om verstreken speelduur voor te lezen.

`ariaVolumeClassId`
: standaard: 'aria-volume'. Class en id's voor om geluidssterkte voor te lezen.

`chooseLabelClassId`
: standaard: 'kiezen-label'. Class en id's voor <label> bij radioknoppen waarmee standaard- of aangepaste videospeler wordt gekozen. (Omdat er maar twee van deze <label>'s zijn, zijn er ook maar twee bijbehorende id's: 'kiezen-label-1' en 'kiezen-label-2'.)

`chooseParClassId`
: standaard: 'kiezen-par'. Class en id's voor <p> waarin de knoppen voor kiezen standaard- of aangepaste videospeler staan. (Omdat er maar twee van deze <p>'s zijn, zijn er ook maar twee bijbehorende id's: 'kiezen-par-1' en 'kiezen-par-2'.)

`chooseRadioClassId`
: standaard: 'kiezen-knop'. Class en id's voor radioknoppen waarmee standaard- of aangepaste videospeler wordt gekozen. (Omdat er maar twee van deze knoppen zijn, zijn er ook maar twee bijbehorende id's: 'kiezen-knop-1' en 'kiezen-knop-2'.)

`controlsDivClassId`
: standaard: 'controls'. Class en id's voor waarbinnen alle bedieningselementen van de videospeler staan.

`durationClassId`
: standaard: 'duration'. Class en id's voor om speelduur weer te geven.

`elapsedClassId`
: standaard: 'elapsed'. Class en id's voor om verstreken speelduur weer te geven.

`fiveBackClassId`
: standaard: 'five-back'. Class en id's voor knop Vijf procent terug.

`fiveForwardClassId`
: standaard: 'five-forward'. Class en id's voor knop Vijf procent vooruit.

`fullScreenClassId`
: standaard: 'fullscreen'. Class en id's voor knop Fullscreen.

`imageDivClassId`
: standaard: ’image’. Class en id’s voor waarbinnen alle bedieningselementen voor afspelen staan.

`imageSliderBeamClassId`
: standaard: ’image-slider-beam’. Class en id’s voor met de balk van de sleepbalk voor afspelen.

`imageSliderButtonClassId`
: standaard: ’image-slider-beam’. Class en id’s voor de met de knop van de sleepbalk voor afspelen.

`imageSliderClassId`
: standaard: ’image-slider’. Class en id’s voor waarbinnen sleepbalk voor afspelen staat.

`louderClassid`
: standaard: ’louder’. Class en id’s voor knop Harder.

`muteClassId`
: standaard: ’mute’. Class en id’s voor Aan-uitknop voor geluid.

`percentageClassId`
: standaard: ’percentage’.Class en id’s voor om geluidssterkte weer te geven.

`playPauseClassId`
: standaard: ’play’. Class en id’s voor Speel-pauzeerknop.

`remainingClassid`
: standaard: ’remaining’. Class en id’s voor om resterende speelduur weer te geven.

`softerClassId`
: standaard: ’softer’. Class en id’s voor knop Zachter.

`soundDivClassId`
: standaard: ’sound’. Class en id’s voor waarbinnen alle bedieningselementen voor volume staan.

`soundSliderBeamClassId`
: standaard: ’sound-slider-beam’. Class en id’s voor met de balk van de sleepbalk voor volume.

`soundSliderButtonClassId`
: standaard: ’sound-slider-button’. Class en id’s voor met de knop van de sleepbalk voor volume.

`soundSliderClassId`
: standaard: ’sound-slider’. Class en id’s voor waarbinnen sleepbalk voor volume staat.

`speedClassId`
: standaard: ’speed’. Class en id’s voor waarbinnen snelheidsregeling staat.

`speedDefaultClassId`
: standaard: ’speed-default’. Class en id’s voor knop voor normale snelheid.

`speedDefaultLabelClassId`
: standaard: ’speed-default-label’. Class en id’s voor <label> bij knop voor normale snelheid.

`speedDoubleClassId`
: standaard: ’speed-double’. Class en id’s voor knop voor dubbele snelheid.

`speedDoubleLabelClassId`
: standaard: ’speed-double-label’. Class en id’s voor <label> bij knop voor dubbele snelheid.

`speedHalfClassId`
: standaard: ’speed-half’. Class en id’s voor knop voor halve snelheid.

`speedHalfLabelClassId`
: standaard: ’speed-half-label’. Class en id’s voor <label> bij knop voor halve snelheid.

`speedOneHalfClassId`
: standaard: ’speed-one-half’. Class en id’s voor knop voor anderhalve snelheid.

`speedOneHalfLabelClassId`
: standaard: ’speed-one-half-label’. Class en id’s voor <label> bij knop voor anderhalve snelheid.

`tenBackClassId`
: standaard: ’ten-back’. Class en id’s voor knop Tien seconden terug.

`tenForwardClassId`
: standaard: ’ten-forward’. Class en id’s voor knop Tien seconden vooruit.

`toBeginClassId`
: standaard: ’to-begin’. Class en id’s voor knop Naar begin video.

`toEndClassId`
: standaard: ’to-end’. Class en id’s voor knop Naar einde video.

`videoClassId`
: standaard: ’video’. Class en id’s voor <video>-element.

`wrapperDivClassId`
: standaard: ’videobox’. Class en id’s voor element waarbinnen de <video>, bijbehorende titels, e.d. staan.
Deze wijkt iets af van alle andere. De class is ’videobox’ en de id ’videobox-1’, enz., net als bij alle andere. Je kunt deze op precies dezelfde manier wijzigen als alle andere. Alleen wordt deze class niet door het script ingevoegd, maar moet deze in de html worden gegeven aan de elementen waarbinnen een <video>-element staat. Als je ’videobox’ wijzigt in het script, moet je dat dus ook in de html wijzigen.
De id’s ’videobox-1’, ’videobox-2’, enz. worden wel door het script ingevoegd. En als je ’videobox’ wijzigt, wijzigt het script automatisch ook de id’s.

Id (eindigend op ’Id’, voor alleen id):
: Er zijn maar enkele elementen die alleen een id krijgen. De meeste elementen kunnen vaker op de pagina voorkomen en krijgen daarom (een id en) een class.
Het is belangrijk dat de opgegeven naam tussen aanhalingstekens staat. Aan het eind van de regel moet een komma staan. (Door de aanhalingstekens weet het script dat het hier om een letterlijke tekst gaat, de komma laat weten dat er nog meer komt.)
Waar de naam voor de id wordt opgegeven, is te herkennen aan het laatste deel van het woord voor het isgelijkteken (de ’variabele’, waarin de naam wordt opgeslagen): deze eindigt altijd op ’Id’.
Alle namen van variabelen die eindigen op ’Id’ zijn onmisbaar voor een goede werking van het script. Als er eentje onbruikbaar is, wordt de standaardwaarde gebruikt, zoals beschreven bij Een id, class, name of aria-label ontbreekt of is niet in orde.

Als je een van deze id's wilt vervangen, vervang je gewoon het deel achter het isgelijkteken:

```
chooseDivId="voor-jou-liefste",
```

De krijgt nu een id='voor-jou-liefste'. (Mocht je op zoek zijn naar 'n partner, probeer dit dan 'ns uit. Dit is beslist origineler dan: "Ken ik jou niet?")

`chooseDivId`
: standaard: 'kiezen-div'. id voor waarbinnen de keuze voor standaard- of aangepaste speler staat.

`durationButtonId`
: standaard: 'duration-button'. id van <button> waarmee tekst van melding over onbekende speelduur kan worden gesloten.

`durationDivId`
: standaard: 'duration-div'. id voor waarbinnen de melding over een onbekende speelduur staat.

`durationParId`
: standaard: 'duration-par'. id van <p> waarbinnen de tekst van de melding over een onbekende speelduur staat.

Name (eindigend op 'Name', voor de name bij radioknoppen):

Er zijn twee groepen radioknoppen, die met behulp van het attribuut `name` aan elkaar worden geknoopt. De eerste is een groep van twee knoppen, waarmee tussen standaard- en aangepaste videospeler wordt gekozen. De tweede is een groep van vier knoppen, waarmee de snelheid kan worden ingesteld.
Het is belangrijk dat de opgegeven naam tussen aanhalingstekens staat. Aan het eind van de regel moet een komma staan. (Door de aanhalingstekens weet het script dat het hier om een letterlijke tekst gaat, de komma laat weten dat er nog meer komt.)
Waar de naam voor de name wordt opgegeven, is te herkennen aan het laatste deel van het woord voor het isgelijkteken (de 'variabele', waarin de naam wordt opgeslagen): deze eindigt altijd op 'Name'.

Alle namen van variabelen die eindigen op 'Name' zijn onmisbaar voor een goede werking van het script. Als er eentje onbruikbaar is, wordt de standaardwaarde gebruikt, zoals beschreven bij Een id, class, name of aria-label ontbreekt of is niet in orde..
Hieronder staat in alfabetische volgorde een rijtje met in het script opgegeven names. Deze zijn allemaal te veranderen op bovenbeschreven wijze. Achter 'standaard' staat de naam die het script gebruikt, als je niets verandert.

`chooseName`
: standaard: 'kiezen'. Name voor radioknoppen waarmee standaard- of aangepaste videospeler wordt gekozen.

`speedName`

standaard: ’knop-snelheid’. Name voor radioknoppen waarmee snelheid wordt ingesteld. Omdat elke videospeler een eigen snelheidsregeling heeft, wordt hier een volgnummer aan toegevoegd. Voor de eerste videospeler is de name ’knop-snelheid-1’, voor de tweede ’knop-snelheid-2’, enz.

soundStartVolume (voor geluidssterkte bij openen pagina):

De geluidssterkte wordt aangegeven met een getal tussen 0 en 1. Hoewel je dit vrijwel altijd in tienden ziet, werken honderdsten ook. Het script gebruikt honderdsten, waardoor de geluidssterkte in procenten kan worden weergegeven. (Voor JavaScript-kenners: het gaat hier om een getal, dus de aanhalingstekens zijn overbodig. Maar voor de eenduidigheid heb ik het ook tussen aanhalingstekens gezet. Zonder aanhalingstekens werkt het trouwens ook.)
Deze variabele is onmisbaar voor een goede werking van het script. Als hij onbruikbaar is, wordt de standaardwaarde gebruikt, zoals beschreven bij Een id, class, name of aria-label ontbreekt of is niet in orde.

`soundStartVolume`

standaard: 0.5. Geluidssterkte bij openen van de pagina. Er moet een getal tussen 0 en 1 worden opgegeven, bijvoorbeeld 0.6 of 0.13. Als decimaalteken moet geen komma, maar een punt worden gebruikt, omdat JavaScript nou eenmaal uit Amerika afkomstig is.

TabIndex (eindigend op ’TabIndex’, voor de tabindex):

Het nut van een tabindex, hoe wordt voorkomen dat alle Speel-pauzeerknoppen dezelfde tabindex krijgen, enz. staat bij Tabindex. Hier staat alleen maar, hoe je de tabindex in het script kunt aanpassen.

Er zijn drie bijzondere waarden voor de tabindex:
* Geen waarde: als je de tabindex helemaal niet wilt gebruiken, vul je alleen `""` in, bijvoorbeeld:

```
playPauseTabIndex = "",
```

Alleen als je precies `""` opgeeft, dus helemaal niets tussen de aanhalingstekens, wordt geen tabindex ingevoegd door het script.
Als je de tabindex niet gebruikt, worden de sleepbalken voor geluid en afspelen nooit bereikt met de Tab-toets. Alleen knoppen, links, e.d. worden afgelopen met behulp van de Tab-toets, en de sleepbalken zijn gewone ’s. Beter is om ’0’ te gebruiken, zoals in het script voor de eerste en tweede pagina is gedaan. Hierdoor worden ook de sleepbalken door de Tab-toets bezocht. (De andere pagina’s hebben totaal gewijzigde tabindexen.)
* 0 of -1: als je als waarde ’0’ of ’-1’ invult, wordt deze waarde gewoon gebruikt. Een `tabindex="0"` en een `tabindex="-1"` hebben een bepaalde betekenis, zoals beschreven bij Tabindex. Als je een negatieve waarde opgeeft, gebruik dan alleen ’-1’. Elke andere negatieve waarde helpt de tabindex grondig om zeep.

* Een positief getal. Dit wordt gebruikt voor ’n gewone tabindex. Om te voorkomen dat dezelfde tabindex bij meerdere elementen voorkomt, wordt de tabindex bij de bedieningselementen voor de eerste video met 100, verhoogd, voor de tweede met 200, enz.

Als je bijvoorbeeld de Speel-pauzeerknop een tabindex van 10 geeft, en de Geluid aan-uitknop een tabindex van 7, wordt bij gebruik van de Tab-toets de Aan-uitknop voor geluid altijd bezocht vóór de Speel-pauzeerknop.
Omdat niet helemaal duidelijk is, hoe een combinatie van `tabindex="0"` en helemaal geen tabindex moet worden afgehandeld, hebben op de eerste en tweede pagina alle links e.d. ook een `tabindex="0"` gekregen (voor zover ze al geen andere tabindex hadden). De andere pagina's hebben een volledig aangepaste tabindex, dus daar geldt dit niet voor.

Het gaat hier om een getal, en dat hoeft niet tussen aanhalingstekens te staan, in tegenstelling tot tekst. Maar voor de eenduidigheid heb ik het ook tussen aanhalingstekens gezet. (Zonder aanhalingstekens werkt het trouwens ook.) Aan het eind van de regel moet een komma staan. (De komma laat weten dat er nog meer komt.)
Waar de waarde voor de tabindex wordt opgegeven, is te herkennen aan het laatste deel van het woord voor het isgelijkteken (de 'variabele', waarin de naam wordt opgeslagen): deze eindigt altijd op 'TabIndex'.
Hieronder staat in alfabetische volgorde een rijtje met in het script opgegeven tabindexen. Deze zijn allemaal te veranderen op bovenbeschreven wijze. Er is geen standaardwaarde voor de tabindex, omdat deze in de diverse scripts verschilt.

Overigens is het voor de toegankelijkheid verreweg het beste, als de volgorde op het scherm (vrijwel) hetzelfde is als de volgorde in de html. Dan hoef je ook niet aan de tabindex te sleutelen, maar kun je die gewoon overal op '0' houden.

`chooseRadioTabIndex`
: tabindex voor radioknoppen waarmee tussen standaard- en aangepaste videospeler wordt gekozen. De twee bij elkaar horende radioknoppen krijgen dezelfde tabindex.

`fiveBackTabIndex`
: tabindex voor knop Vijf procent terug.

`fiveForwardTabIndex`
: tabindex voor knop Vijf procent vooruit.

`fullScreenTabIndex`
: tabindex voor knop Fullscreen.

`imageSliderBeamTabIndex`
: tabindex voor de met de balk van de sleepbalk voor afspelen. Omdat de knop van de sleepbalk feitelijk helemaal nep is, hoeft deze geen tabindex te krijgen.

`louderTabIndex`
: tabindex voor knop Harder.

`muteTabIndex`
: tabindex voor Aan-uitknop voor geluid.

`playPauseTabIndex`
: tabindex voor Speel-pauzeerknop.

`softerTabIndex`
: tabindex voor knop Zachter.

`soundSliderBeamTabIndex`
: tabindex voor de met de balk van de sleepbalk voor volume. Omdat de knop van de sleepbalk feitelijk helemaal nep is, hoeft deze geen tabindex te krijgen.

`speedTabIndex`
: tabindex voor radioknoppen waarmee de snelheid wordt ingesteld. De vier bij elkaar horende radioknoppen krijgen dezelfde tabindex.

`tenBackTabIndex`
: tabindex voor knop Tien seconden terug.

`tenForwardTabIndex`
: tabindex voor knop Tien seconden vooruit.

`toBeginTabIndex`
: tabindex voor knop Naar begin video.

`toEndTabIndex`
: tabindex voor knop Naar einde video.

TEXT (eindigend op 'Text', voor tekst op knoppen, in meldingen e.d.):

Op een aantal plaatsen wordt gewone tekst gebruikt. Ook die teksten kunnen worden aangepast. Als je iets niet wilt gebruiken, laat je het gewoon leeg, bijvoorbeeld

```
chooseLabelCustomText = "",
```

Dus tussen de aanhalingstekens wordt gewoon niets gezet. Maar normaal genomen is dat bepaald niet aan te raden, want het voorlezen van de verstreken speelduur is dan niet meer iets als '1 uur 13 minuten 17 seconden', maar '1 13 17'. En kiezen tussen standaard- en aangepaste videospeler is ook wat lastig, als je alleen maar twee radioknoppen zonder tekst ziet.
Op bovenstaande manier wordt iets voor de hele pagina verborgen. Je kunt iets ook slechts voor één of meer videospelers verbergen, door gebruik te maken van iets als `color: transparent;` in de css. Dit is op de derde pagina met videospelers gedaan bij de snelheid. 0,5x, 1x, 1,5x en 2x zijn bij de derde en vijfde video doorzichtig gemaakt, maar ze zijn wel aanwezig.

Het is belangrijk dat de opgegeven tekst tussen aanhalingstekens staat. Aan het eind van de regel moet een komma staan. (Door de aanhalingstekens weet het script dat het hier om een letterlijke tekst gaat, de komma laat weten dat er nog meer komt.) Waar de tekst wordt opgegeven, is te herkennen aan het laatste deel van het woord voor het isgelijkteken (de 'variabele', waarin de naam wordt opgeslagen): deze eindigt altijd op 'Text'.

Een nieuwe regel in de tekst kun je gewoon aangeven met `<br>` binnen de tekst, net zoals je dat in html zou doen. Op precies dezelfde manier kun je ook tags als `<i></i>` of zelfs `<span></span>` gebruiken binnen de tekst.
Een speciaal teken, zoals een letter met een accent, kan op dezelfde manier worden ingevoerd als in gewone html. Ook utf-8-codes, zoals `⅔` (⅔) kunnen op de normale manier worden gebruikt.
Er zijn maar twee tekens die problemen kunnen opleveren: dubbele aanhalingstekens en de \ ('backslash' in het Engels, een echte Nederlandse naam is er niet).
Als je " binnen de tekst gebruikt, denkt de niet al te snuggere browser dat dat het einde van de letterlijke tekst aangeeft en gaat proberen de rest van de tekst als commando's uit te voeren, waarna de ziel zich gruwelijk in het script verslikt en de hele zaak acuut uitspuugt. Resultaat: een pagina zonder JavaScript.

Je kunt wel dubbele aanhalingstekens in de tekst gebruiken, maar die moet je dan laten voorafgaan door een \: Ik zei: `\"Kom nou\"`. Door de \ weet de browser dat het teken erna letterlijk moet worden genomen, en dus niet het begin of einde van een stuk tekst of zo is.
Als je in de tekst een of zoiets met een id op wilt nemen, moet je dubbele aanhalingstekens ook op bovenbeschreven manier gebruiken. Je kunt ook enkele aanhalingstekens gebruiken, want die leveren geen enkel probleem op.
De \ in JavaScript geeft dus aan, dat het teken erna letterlijk moet worden genomen, of dat het een speciale betekenis heeft. Maar daardoor is de \ zelf niet meer te gebruiken. Tenzij je er een \ voor zet, want dan wordt de tweede \ weer letterlijk genomen. Als je dus in de tekst 'n gewone \ wilt gebruiken, moet je die intypen als `\\`.

Hieronder staat in alfabetische volgorde een rijtje met in het script opgegeven teksten. Deze zijn allemaal te veranderen op bovenbeschreven wijze. Achter 'standaard' staat de tekst die het script gebruikt, als je niets verandert.

`ariaEndVideoText`
: standaard: 'Einde video'. Tekst die door schermlezers voorgelezen, als het einde van de video wordt bereikt.

`ariaHourText`
: standaard: 'uur'. Tekst die door schermlezers wordt gebruikt bij het voorlezen van de verstreken speelduur.

`ariaMinutesText`
: standaard: 'minuut'. Tekst die door schermlezers wordt gebruikt bij het voorlezen van de verstreken speelduur.

`ariaSecondsText`
: standaard: 'seconden'. Tekst die door schermlezers wordt gebruikt bij het voorlezen van de verstreken speelduur.

`ariaStartVideoText`
: standaard: 'Begin video'. Tekst die door schermlezers wordt voorgelezen, als het einde van de video wordt bereikt.

`ariaVolumeText`
: standaard: 'procent'. Tekst die door schermlezers wordt gebruikt bij het voorlezen van de geluidssterkte.

`chooseLabelCustomText`
: standaard: 'O, iets nieuws, dolletjes! Daar raak ik opgewonden van, stouterd, hihihi! Ik kies voor de aangepaste besturing voor de video: '
Deze tekst hoort bij de radioknop, waarmee je kiest voor de aangepaste videospeler. Hoe je in deze tekst nieuwe regels kunt aangeven, aanhalingstekens kunt gebruiken, e.d. staat iets hierboven bij Een nieuwe regel...

`chooseLabelDefaultText`
: standaard: 'Iets nieuws. Jasses, wat eng. Misschien zijn het wel gesluierde knoppen. Ik eet alleen wat ik ken en wil m'n eigenste standaardbesturing voor de video: '
Deze tekst hoort bij de radioknop, waarmee je kiest voor de standaardvideospeler. Hoe je in deze tekst nieuwe regels kunt aangeven, aanhalingstekens kunt gebruiken, e.d. staat iets hierboven bij Een nieuwe regel...

`durationButtonText`

standaard: ”Melding sluiten”. Tekst op de knop om de melding over onbekende speelduur te sluiten. Als geen tekst is opgegeven, wordt ’OK’ gebruikt. Deze tekst wordt in een melding gebruikt en is daarom een apart geval. Uitgebreidere info staat bij Speelduur nog onbekend.

`durationUnknownText`

dit is een tamelijk lange tekst, die in een melding wordt gebruikt. Hoe je deze tekst kunt veranderen, de melding kunt uitschakelen, het uiterlijk kunt aanpassen, enz. wordt verder beschreven bij Speelduur nog onbekend.

`speedDefaultLabelText`

standaard: ’1x’. Tekst die hoort bij de <label> voor de radioknop die bij de normale snelheid hoort.

`speedDoubleLabelText`

standaard: ’2x’. Tekst die hoort bij de <label> voor de radioknop die bij de dubbele snelheid hoort.

`speedHalfLabelText`

standaard: ’0,5x’. Tekst die hoort bij de <label> voor de radioknop die bij halve snelheid hoort.

`speedOneHalfText`

standaard: ’1,5x’. Tekst die hoort bij de <label> voor de radioknop die bij anderhalve snelheid hoort.

Title (eindigend op ’Title’, voor de title):

De title verschijnt, als je met de muis boven een knop, sleepbalk, e.d. gaat hangen. Als je geen aria-label gebruikt, lezen sommige schermlezers hem voor.
Als je ’n title niet wilt gebruiken, vul je gewoon geen tekst in:

```
softerTitle = "",
```

De knop Zachter krijgt nu geen title.

Het is belangrijk dat de opgegeven naam tussen aanhalingstekens staat. Aan het eind van de regel moet een komma staan. (Door de aanhalingstekens weet het script dat het hier om een letterlijke tekst gaat, de komma laat weten dat er nog meer komt.)
Waar de tekst voor de title wordt opgegeven, is te herkennen aan het laatste deel van het woord voor het isgelijkteken (de ’variabele’, waarin de naam wordt opgeslagen): deze eindigt altijd op ’Title’.
Je kunt in de title een nieuwe regel opgeven, door de code `\n` te gebruiken:

```
Normale\nsnelheid
```

Van de `\n` zie je niets, maar ’snelheid’ komt op een nieuwe regel te staan in de tekstballon die bij hoveren over het element verschijnt.

Hieronder staat in alfabetische volgorde een rijtje met in het script opgegeven titles. Deze zijn allemaal te veranderen op bovenbeschreven wijze. Achter ’standaard’ staat de tekst die het script gebruikt, als je niets verandert.

`chooseRadioCustomTitle`

standaard: ’Aangepaste besturing’. Title bij radioknop waarmee aangepaste videorecorder wordt gekozen.

`chooseRadioDefaultTitle`

standaard: ’Standaardbesturing’. Title bij radioknop waarmee standaardvideospeler wordt gekozen.

`durationTitle`
: standaard: ’Speelduur’. Title voor om speelduur weer te geven.

`elapsedTitle`
: standaard: ’Verstreken’. Title voor om verstreken speelduur weer te geven.

`fiveBackTitle`
: standaard: ’Vijf procent terug’. Title voor knop Vijf procent terug.

`fiveForwardTitle`
: standaard: ’Vijf procent verder’. Title voor knop Vijf procent vooruit.

`fullScreenTitle`
: standaard: ’Volledig scherm’. Title voor knop Fullscreen.

`imageSliderBeamTitle`
: standaard: ’Sleepbalk afspelen’. Title voor met balk van de sleepbalk voor afspelen.

`imageSliderButtonTitle`
: standaard: ’Sleepbalk afspelen’. Title voor met knop van sleepbalk voor afspelen. (De knop krijgt ook een title, omdat je met de muis ook precies boven de knop kunt hangen. In dat geval verschijnt de bij de sleepbalk horende title niet.)

`louderTitle`
: standaard: ’Harder’. Title voor knop Harder.

`muteTitle`
: standaard: ’Geluid uit’. Title voor Aan-uitknop voor geluid. Deze title verschijnt alleen als het geluid aan staat. Dit wordt door het script geregeld.

`pauseTitle`
: standaard: ’Pauzeren’. Title voor Speel-pauzeerknop. Deze title verschijnt alleen als de video speelt. Dit wordt door het script geregeld.

`percentageTitle`
: standaard: ’Geluidssterkte’. Title voor om geluidssterkte weer te geven.

`playTitle`
: standaard: ’Afspelen’. Title voor Speel-pauzeerknop. Deze title verschijnt alleen als de video niet speelt. Dit wordt door het script geregeld.

`remainingTitle`
: standaard: ’Resterend’. Title voor om resterende speelduur weer te geven.

`softerTitle`
: standaard: ’Zachter’. Title voor knop Zachter.

`soundSliderBeamTitle`
: standaard: ’Sleepbalk volume’. Title voor met balk van de sleepbalk voor geluidssterkte.

`soundSliderButtonTitle`
: standaard: ’Sleepbalk volume’. Title voor met knop van sleepbalk voor geluidssterkte. (De knop krijgt ook een title, omdat je met de muis ook precies boven de knop kunt hangen. In dat geval verschijnt de bij de sleepbalk horende title niet.)

`speedDefaultLabelTitle`
: standaard: ’Normale snelheid’. Title voor knop voor normale snelheid.

`speedDefaultTitle`
: standaard: 'Normale snelheid'. Title voor <label> bij knop voor normale snelheid.

`speedDoubleLabelTitle`
: standaard: 'Dubbele snelheid'. Title voor knop voor dubbele snelheid.

`speedDoubleTitle`
: standaard: 'Dubbele snelheid'. Title voor <label> bij knop voor dubbele snelheid.

`speedHalfLabelTitle`
: standaard: 'Halve snelheid'. Title voor knop voor halve snelheid.

`speedHalfTitle`
: standaard: 'Halve snelheid'. Title voor <label> bij knop voor halve snelheid.

`speedOneHalfLabelTitle`
: standaard: 'Anderhalve snelheid'. Title voor knop voor anderhalve snelheid.

`speedOneHalfTitle`
: standaard: 'Anderhalve snelheid'. Title voor <label> bij knop voor anderhalve snelheid.

`tenBackTitle`
: standaard: 'Tien seconden terug'. Title voor knop Tien seconden terug.

`tenForwardTitle`
: standaard: 'Tien seconden verder'. Title voor knop Tien seconden vooruit.

`toBeginTitle`
: standaard: 'Naar begin video'. Title voor knop Naar begin video.

`toEndTitle`
: standaard: 'Naar einde video'. Title voor knop Naar einde video.

`unmuteTitle`
: standaard: 'Geluid aan'. Title voor Aan-uitknop voor geluid. Deze title verschijnt alleen als het geluid uit staat. Dit wordt door het script geregeld.

Onderdelen in het script uitschakelen

Veel onderdelen in het script zijn relatief simpel uit te schakelen. Maar dan moet je echt aan het script zelf gaan sleutelen. Het voert te ver, om dat hier allemaal te gaan bespreken. Eén voorbeeld. In het element waarbinnen de <video> staat, wordt in het attribuut `data-played` het percentage verstreken speelduur steeds bijgewerkt. Als je in het script de regel

```
wrapperDivNodeList[indexNr].setAttribute("data-played",
    "percent-" + twoDigits(Math.round((elapsed /
    duration) * 100)));
```

wegcommentarieert, wordt het percentage niet meer bijgewerkt, terwijl de rest van het script gewoon blijft werken. (Als je niet weet, wat wegcommentariëren betekent, zou ik echt helemaal van het script afblijven.)

Door het script aangestuurde data-attributen (data-...="...")

Met behulp van JavaScript wordt aan de html een aantal data-attributen toegevoegd. Een data-attribuut begint altijd met `data-`, gevolgd door een woord, zoals `data-duration`. Hierachter volgt een isgelijkteken, gevolgd door de waarde van het attribuut tussen aanhalingstekens, bijvoorbeeld `data-duration="unknown"`.

Data-attributen zijn iets nieuws van html5. De enige voorwaarde is dat de naam moet beginnen met `data-`. Ze geven de mogelijkheid om zelfgemaakte attributen aan een element toe te voegen. Op het aan- of afwezig zijn van zo'n attribuut, of op de waarde ervan,

kan dan later worden getest met behulp van een css-selector zoals `[]`, waardoor de lay-out kan worden aangepast. Daar wordt in dit voorbeeld gebruik van gemaakt.
Ook de waarde van het data-attribuut kan worden veranderd met behulp van JavaScript. Ook daar wordt hier gebruik van gemaakt. Bij `data-duration` bijvoorbeeld verandert 'unknown' in 'known', zodra de speelduur van de video bekend is.
Door in de css met behulp van selectors als `[]` op de aan- of afwezigheid van een bepaald data-attribuut te testen, en eventueel op de waarde van dat data-attribuut, kan de opmaak worden aangepast met behulp van deze data-attributen. Dat geeft bijvoorbeeld de mogelijkheid om op iOS, waar de geluidssterkte alleen met behulp van de knoppen op het apparaat zelf kan worden geregeld, de volumeregeling in de videospeler te verbergen. Of er een rood kruis door te zetten, maar alleen op iOS.
Omdat deze data-attributen door JavaScript worden toegevoegd, zijn ze niet te zien in de normale code van de pagina. Bij het bekijken van de code van de pagina wordt alleen de 'normale' html getoond, geen dingen die later zijn toegevoegd. Om de data-attributen te zien, moet je de gegenereerde code bekijken. Dat is de code zoals de browsers die uiteindelijk te zien krijgt, inclusief bijvoorbeeld data-attributen.
(In dit geval toont de normale broncode helemaal weinig, omdat de hele videospeler uit gegenereerde code bestaat, en daardoor alleen te zien is op de bovengenoemde manier.)

De in dit voorbeeld gebruikte data-attributen zijn de volgende:

`data-duration`

Bij opening van de pagina wordt aan elk element met de class="videobox" een attribuut `data-duration` toegevoegd. De waarde hiervan is "unknown": `data-duration="unknown"`. Zodra de speelduur van de video bekend is, wordt de waarde veranderd in "known": `data-duration="known"`.
Hiervan wordt bijvoorbeeld gebruik gemaakt bij de video's op de eerste pagina. Bij sommige systemen of browsers is soms de speelduur van de video bij opening van de pagina nog onbekend. Omdat dat zo is ingebouwd, of omdat de browser zo krakkemikkig is.
Door met behulp van een selector te testen op de inhoud van `data-duration`, kan het uiterlijk van de videospeler met css worden aangepast. In de voorbeelden zijn de bedieningselementen die zonder bekende speelduur niet werken met behulp van css doorzichtig gemaakt. Zodra de speelduur bekend is, zien ze er weer gewoon uit.
Omdat niet alle browsers en systemen bij opening van de pagina altijd de speelduur van elke video weten vast te stellen, wordt ook op latere momenten nog geprobeerd die speelduur vast te stellen. De speelduur is in ieder geval bekend, zodra met afspelen wordt begonnen.

`data-fullscreen`

Bij opening van de pagina wordt o.a. een knop aangemaakt om de video fullscreen af te kunnen spelen. Deze knop heeft als id "fullscreen-volgnummer" en als class "fullscreen".
Bij opening van de pagina wordt aan elk van die knoppen een attribuut `data-fullscreen` toegevoegd met als waarde "off": `data-fullscreen="off"`.
Als de knop Fullscreen wordt aangeraakt of aangeklikt, wordt met behulp van JavaScript gekeken of de browser de functie `requestFullscreen()` kent. Dit is een wat nieuwere functie, die nog niet alle browsers kennen. Als de browser deze functie kent, wordt hiervan gebruik gemaakt om de video fullscreen af te spelen.
Als de browser `requestFullscreen()` niet kent, krijgt `data-fulsscreen` de waarde "on": `data-fullscreen="on"`. Hierdoor kun je met behulp van css

een soort `requestFullscreen()` simuleren, waardoor ook in oudere browsers de video fullscreen kan worden afgespeeld.

`data-hour`

Zodra de speelduur van de video bekend is, wordt aan elk element met de class=”videobox” een attribuut `data-hour` toegevoegd. De waarde hiervan is ”yes” of ”no”: `data-hour="yes"` of `data-hour="no"`.
Als de video langer dan een uur duurt, wordt de waarde van `data-hour` ”yes”, als de video korter dan een uur duurt ”no”. Met behulp van css kan op die manier het wel of niet weergeven van uren in verstreken speelduur e.d. worden geregeld.
Omdat niet alle browsers en systemen bij opening van de pagina altijd de speelduur van elke video weten vast te stellen, wordt ook op latere momenten nog geprobeerd `data-hour` de juiste waarde te geven. De speelduur is in ieder geval bekend, zodra met afspelen wordt begonnen. Vuistregel: zodra de speelduur bekend is, heeft `data-hour` automatisch de juiste waarde.
Het daadwerkelijke invoegen van het uur of de uren wordt geregeld door het script. Als de video korter dan ’n uur is, worden geen uren weergegeven. Als de video langer dan ’n uur is, worden de uren of het uur in cijfers getoond, gevolgd door een : (dubbele punt). Maar het uiterlijk van die weergave wordt met behulp van css geregeld.

`data-played`

Bij opening van de pagina wordt aan elk element met de class=”videobox” een attribuut `data-played` toegevoegd. Dit attribuut heeft de waarde ”percent-”, gevolgd door het percentage van de verstreken speelduur. Bij de opening van de pagina is het percentage 0. Bij opening van de pagina ziet dit data-attribuut er dan ook als volgt uit:

`data-played="percent-00".`

Bij verandering van de verstreken speelduur wordt het percentage aangepast. Hiervan wordt gebruik gemaakt bij bijvoorbeeld de tweede videospeler op de vijfde pagina: de video draait in de rondte. Het aantal graden van de draaiing is afhankelijk van de verstreken speelduur.

`data-smscr`

Deze doet hier niet mee, omdat deze niet door JavaScript wordt aangemaakt, maar gewoon handmatig in de html is opgenomen. Meer hierover vind je bij data-smscr=”480”.

`data-sound`

Bij opening van de pagina wordt aan elk element met de class=”videobox” een attribuut `data-sound` toegevoegd. Dit attribuut heeft de waarde ”yes” of ”no”: ”yes” als de geluidssterkte met behulp van de knoppen in de videospeler kan worden geregeld, ”no” als dat niet kan.
Door in het script te testen of geluidsregeling met behulp van JavaScript mogelijk is, kan worden bepaald of ”yes” of ”no” als waarde moet worden ingevuld.
Op iOS kan de geluidssterkte alleen met behulp van de knoppen op het apparaat worden geregeld. Op iOS is de waarde van `data-sound` dan ook ”no”: `data-sound="no"`. In alle andere geteste systemen en browsers is de waarde ”yes”: `data-sound=”yes”`. Dit geeft de mogelijkheid met behulp van css de knoppen voor de geluidsregeling op iOS weg te laten, of ze er anders uit te laten zien.

Hiervan wordt gebruik gemaakt in veel video's, zoals op die op de eerste pagina, waar voor iOS de geluidsregeling ontbreekt. Bij de video's op de derde pagina staat op iOS, behalve bij de vierde videospeler, een rood kruis door de knoppen van de geluidsregeling.
(In sommige browsers op Android kan de snelheid niet worden gewijzigd. Helaas kan daar niet dezelfde truc worden toegepast, omdat wordt gemeld dat de snelheid is gewijzigd, terwijl dat in werkelijkheid niet zo is. Hetzelfde geldt voor de geluidssterkte in Android 4.0.2. Deze kan alleen met de knoppen van het apparaat worden gewijzigd, maar meldt ten onrechte dat de geluidssterkte is gewijzigd, als je dat in de videospeler doet.)

`data-volume`

Bij opening van de pagina wordt aan elk element met de class="videobox" een attribuut `data-volume` toegevoegd. Dit attribuut heeft de waarde "percent-", gevolgd door het percentage van de geluidssterkte. In het script kan een beginwaarde voor de geluidssterkte worden opgegeven. In dit voorbeeld is dat 50. Bij opening van de pagina ziet dit data-attribuut er dan ook als volgt uit:

```
data-volume="percent-50"
```

Bij verandering van de geluidssterkte wordt het percentage aangepast. Hiervan wordt gebruik gemaakt op bijvoorbeeld de eerste video op de vijfde pagina: de teksten onder de video veranderen bij een andere geluidssterkte.

Door het script ingevoegde css

De opmaak van de videospelers wordt volledig met behulp van css gedaan. Op de normale manier, wat normaal genomen een extern stylesheet zal zijn. Maar daar zijn een paar kleine uitzonderingen op.
Een heel klein beetje css is absoluut onmisbaar voor een goede werking van sommige onderdelen van het script. Die css wordt ingevoegd als een inline-style: css in de tag van het html-element zelf, bijvoorbeeld `<div style="height: 100px;">`.
Dit soort css 'wint' altijd van css die in een extern stylesheet of in de <head> van de pagina staat. Dat kan dus tot botsingen met een extern stylesheet leiden. Als jij een rode achtergrond wilt, en het script wil een groen, dan wint het script. Waar dat nodig is, is daarom een controle ingebouwd om dit soort conflicten te voorkomen.
Het gaat hier om css die door het script is gegenereerd. Daardoor is hij niet te zien in de normale broncode van de pagina, maar alleen in de gegenereerde code.
De door het script gegenereerde css is in drie groepen te verdelen.

CSS DIE ALTIJD WORDT INGEVOEGD

```
controlsDivNodeList[i].setAttribute("style",
    "-ms-touch-action: none; touch-action: none; -moz-
    user-select: none; -ms-user-select: none; -webkit-
    user-select: none; -user-select: none;");
```

Het eerste stukje van bovenstaande regel:
`controlsDivNodeList[i].setAttribute("style", "…")`
Ik ga dit niet helemaal uitleggen, want dan wordt het een JavaScript-verhaal. Dus je moet me maar gewoon geloven. Ik ben niet altijd eerlijk geweest in m'n leven, maar ik beloof echt dat ik nu goudeerlijk ben.
Bovenstaand stukje code zorgt ervoor dat er iets gebeurt bij elk element met class="controls'. (Even er vanuit gaande, dat je de naam van de class niet hebt veranderd bovenin het script.)

Wat er gebeurt, staat achter de punt: er wordt een attribuut toegevoegd. Het attribuut 'style'. Oftewel: een inline-style. De waarde van het attribuut staat in het tweede deel: achter de komma tussen de aanhalingstekens.
Het volgende wordt ingevoegd:

```
style="-ms-touch-action: none; touch-action:
    none; -moz-user-select: none; -ms-user-
    select: none; -webkit-user-select: none;
    -user-select: none;"
```

Het eerste deel hiervan:

```
-ms-touch-action: none; touch-action: none;
```

Hier staat in feite twee keer hetzelfde: `touch-action: none;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Op een touchscreen wordt normaal genomen de pagina gescrold, als je je vinger over het scherm beweegt. Dat gebeurt ook, als je je vinger over de sleepbalken voor geluid of beeld beweegt. Maar daar is dat natuurlijk niet de bedoeling. Daarom wordt in het element met class="controls", waarbinnen alle bedieningselementen van de video staan, het scrollen uitgeschakeld. Dit is met name nodig voor Internet Explorer 11, omdat de sleepbalken in die browser anders volstrekt onbruikbaar zijn.

Het tweede deel:

```
-moz-user-select: none; -ms-user-select: none;
    -webkit-user-select: none; -user-select:
    none;
```

Hier staat in feite vier keer hetzelfde: `user-select: none;`. Waarom dat zo is, staat bij De voorvoegsels -moz-, -ms- en -webkit-.
Deze eigenschap staat (nog) in geen enkele officiële specificatie. Als je deze eigenschap zou valideren, zou je daarom een foutmelding krijgen. Maar omdat de css als inline-style in de html wordt ingevoegd, geeft de validator van w3c toch geen foutmelding.
Omdat dit (nog) in geen enkele specificatie staat, is het niet helemaal zonder risico dit te gebruiken. In de toekomst zou het gedrag van de eigenschap kunnen veranderen. Omdat de exacte details van de werking hier niet zo heel belangrijk zijn (er moet gewoon heel zwart-wit helemaal niets worden geselecteerd), durf ik het toch aan het te gebruiken.
Behalve Internet Explorer 9 en Opera Mini kent elke browser deze eigenschap. Je voorkomt ermee dat tekst wordt geselecteerd. Als je op een touchscreen een knop iets langer aanraakt, loop je anders het risico dat de tekst op de knop wordt geselecteerd om te kopiëren of zo.
Bijzonder irritant als er allemaal behulpzame knoppen en zo op het verkeerde moment verschijnen. Bovendien werkt de knop dan soms ook nog 'ns niet door al die ongevraagde hulp.

`soundSliderButtonNodeList[i].style.position = "absolute";`

Het stukje `soundSliderButtonNodeList` verwijst naar een lijst, waarin het script als het ware alle knoppen van de sleepbalken voor het geluid heeft opgeslagen.
Om de knop op de sleepbalk voor geluid goed neer te kunnen zetten, móét deze absoluut zijn gepositioneerd ten opzichte van de balk van de sleepbalk.
Om er zeker van te zijn dat deze knop absoluut is gepositioneerd, voegt het script bij alle knoppen voor de sleepbalk voor het geluid deze inline-style in:

```
style="position: absolute;"
```

`soundSliderButtonNodeList[i].style.visibility = "hidden";`

Het stukje `soundSliderButtonNodeList` verwijst naar een lijst, waarin het script als het ware alle knoppen van de sleepbalken voor het geluid heeft opgeslagen.
Bij openen van de pagina wordt de knop van de sleepbalk voor het geluid op een bepaalde positie neergezet, afhankelijk van de opgegeven geluidssterkte. Deze positie moet worden berekend, wat een klein beetje tijd kost. Daardoor wordt de knop in eerste instantie helemaal links neergezet, om pas even daarna op de juiste plaats neergezet te worden: de knop verspringt.
Om dat te voorkomen, wordt de knop bij openen van de pagina verborgen met de inline-style:

```
style="visibility: hidden;"
```

Dit verbergen moet gebeuren met `visibility: hidden;` en niet met `display: none;`, want de knop is wel nodig om de juiste positie te berekenen. En met `display: none;` is er domweg helemaal geen knop.

`soundSliderButtonNodeList[i].style.visibility = "visible";`

Het stukje `soundSliderButtonNodeList` verwijst naar een lijst, waarin het script als het ware alle knoppen van de sleepbalken voor het geluid heeft opgeslagen.
Hier gelijk boven zijn de knoppen van de sleepbalken voor het geluid met `visibility: hidden;` verborgen, om verspringen te voorkomen. Als de pagina helemaal is geladen, mogen de knoppen weer tevoorschijn komen, want dan verspringen ze niet meer.
Als de pagina helemaal is geladen, wordt de functie `afterPageLoaded()` uitgevoerd. Daarin staat o.a. bovenstaande regel, die de knoppen weer zichtbaar maakt met de inline-style:

```
style="visibility: visible;"
```

`imageSliderButtonNodeList[i].style.position = "absolute";`

Het stukje `imageSliderButtonNodeList` verwijst naar een lijst, waarin het script als het ware alle knoppen van de sleepbalken voor het afspelen heeft opgeslagen.
Om de knop op de sleepbalk voor afspelen goed neer te kunnen zetten, móét deze absoluut zijn gepositioneerd ten opzichte van de balk van de sleepbalk.

Om er zeker van te zijn dat deze knop absoluut is gepositioneerd, voegt het script bij alle knoppen voor de sleepbalk voor het afspelen deze inline-style in:

```
style="position: absolute;"
```

```
document.body.setAttribute("style", "zoom: 1.0001");
document.body.setAttribute("style", "zoom: 1");
```

`document.body`: dit is gewoon de <body> van de pagina.

`setAttribute`: er wordt, o grote verrassing, een attribuut neergezet.

`style` is het attribuut, `zoom` met het getal erachter is de waarde:

`style="zoom: 1.0001"` of `style="zoom: 1"`

Deze twee regels zijn eeneiige tweelingzusjes. Pas vier cijfers na de komma is er verschil te bespeuren. Maar net als bij echte tweelingen hebben ze wel degelijk allebei een eigen functie.

Safari op OS X heeft een vreemde bug. Als je kiest voor standaard- en daarna weer voor aangepaste videospelers, wordt het scherm niet meer bijgewerkt. Alle knoppen blijven werken e.d., maar het uiterlijk verandert niet meer. Kennelijk heeft de css geen effect meer. Dit gebeurt alleen als op de pagina ook @keyframes wordt gebruikt in combinatie met `opacity` en/of `transform: rotate()`.

Dit kan worden opgelost door minimaal te zoomen. Dat is de vreemde `zoom: 1.0001`. Een zoom van 0,0001% is niet te zien, maar het lost wel dit probleem op.

Als je kiest voor de standaardvideospelers, wordt gezoomd naar 1. Oftewel: er gebeurt helemaal niets. Ga je dan weer terug naar de aangepaste spelers, dan wordt gezoomd naar 1.0001. Deze kleine zoom dwingt Safari kennelijk het scherm opnieuw op te bouwen.

Deze door het script ingevoegde inline-style overschrijft een eventueel al aanwezige inline-style bij <body>. Het gebruik van een inline-style is altijd al een bijzonder slechte gewoonte, maar in dit geval is het bij <body> dus gewoon onmogelijk, omdat je de kans loopt dat die style gewoon volledig wordt overschreven.

```
soundSliderButtonNodeList[indexNr].style.left =
    Math.round(Math.max(0,
    Math.min(availableSpaceSliderBeam, correction *
    e.clientX - xSliderBeam))) + "px";
soundSliderButtonNodeList[indexNr].style.left =
    Math.round(Math.max(0,
    Math.min(availableSpaceSliderBeam, correction *
    screenX - xSliderBeam))) + "px";
soundSliderButtonNodeList[indexNr].style.left =
    Math.round(availableSpaceSliderBeam / 100 *
    percentage) + "px";
```

```
soundSliderButtonNodeList[i].style.left =
     Math.round(((soundSliderBeamNodeList[i].clientWidth
     / (100 / (100 * soundStartVolume)))) -
     (soundSliderButtonNodeList[i].offsetWidth / (100 /
     (100 * soundStartVolume)))) + "px";
```

Het begin van deze vier regels, `soundSliderButtonNodeList`, verklapt dat deze regels iets doen met de knop van de sleepbalk voor het geluid. En wel met `left`.

Alle vier de regels doen hetzelfde, maar steeds op een iets andere manier, afhankelijk van waar ze staan: ze rekenen uit waar de knop van de sleepbalk voor geluid moet komen te staan. Als je uit bovenstaande kunt aflezen, hoe dat werkt, hoef ik het niet uit te leggen. En als je dat niet kunt, begin ik er niet aan, want dat kost ’n half boek. En uiteindelijk is deze site toch meer op css gericht.

Kort samengevat: kijk hoe breed de balk van de sleepbalk is, kijk hoe breed de knop is (inclusief evt. borders e.d.), bereken op welk percentage de knop moet worden neergezet, gebaseerd op de geluidssterkte en de schoenmaat van de bezoeker, zet de knop op dat percentage vanaf de linkerkant van de balk in de sleepbalk neer.

Dat levert dus iets op als:

```
     style="left: 100px;"
```

Wat, als je al die indrukwekkende code ziet, toch een wat tegenvallend, weinig indrukwekkend eindresultaat is.

In de onderste regel is trouwens de beginsterkte van het geluid bij openen van de pagina te herkennen: `soundStartVolume`. Deze regel code bepaalt de stand van de knop bij openen van de pagina.

```
imageSliderButtonNodeList[indexNr].style.left =
     Math.round(Math.min(imageSliderBeamNodeList[indexNr]
     .clientWidth, Math.max(0, (elapsed * pxPerSecond))))
     + "px";
imageSliderButtonNodeList[indexNr].style.left =
     Math.round(Math.max(0,
     Math.min(availableSpaceSliderBeam, correction *
     (e.clientX - xSliderBeam)))) + "px";
imageSliderButtonNodeList[indexNr].style.left =
     Math.round(Math.max(0,
     Math.min(availableSpaceSliderBeam, correction *
     (screenX - xSliderBeam)))) + "px";
```

Hiervoor geldt precies hetzelfde als gelijk hierboven voor de knop van de sleepbalk voor geluid, maar nu voor de knop van de sleepbalk voor het afspelen. Omdat deze bij openen van de pagina altijd helemaal links staat, is er – anders dan bij de knop voor de sleepbalk voor het geluid – geen aparte regel om de plaats bij het openen van de pagina te bepalen.

```
controlsDivNodeList[i].style.display = "block";
controlsDivNodeList[i].style.display = "none";
```

Iets hierboven hadden we ’n tweelingzusje, hier hebben we ’n tweelingbroertje.

`controlsDivNodeList` is een soort lijst, waarin elke met bedieningselementen als het ware is opgeslagen. Er moet iets gebeuren met `display` bij die elementen. De bovenste regel levert `style="display: block;"` op, de onderste `style="display: none;"`.

De bovenste regel zorgt dat de wordt getoond, en dus de erin zittende bedieningselementen, de onderste verbergt de met inhoud juist. De bovenste inline-style wordt ingevoegd als je kiest voor de aangepaste videospelers, de tweede als je kiest voor de standaardvideospeler.

```
wrapper.style.borderColor = "red";
wrapper.style.borderWidth = "3px";
wrapper.style.borderStyle = "solid";
```

Deze drie stijlen worden gebruikt, als er een element met class=”videobox” is, waarin geen <video>-element zit. (Aangenomen dat je de naam van de class ’videobox’ niet bovenin het script hebt veranderd, anders moet je natuurlijk voor ’videobox’ de nieuwe naam invullen.)

Als dit gebeurt, raakt het script echt volledig van slag. Daarom verschijnt een waarschuwing, waarin verteld wordt wat er mis is. Tegelijkertijd wordt het element, waaruit kennelijk een <video> is weggelopen, rood omlijnd, zodat je gelijk kunt zien, waar het probleem zit. Voor dat omlijnen worden deze stijlen gebruikt. Zodra de fout is hersteld en de pagina wordt herladen, zijn de inline-styles verdwenen.

```
video.style.width = "100%";
video.style.height = "100%";
video.style.position = "fixed";
video.style.top = "30px";
video.style.left = "0";
video.style.zIndex = "137720";
```

Als de video fullscreen moet worden afgespeeld, wordt eerst gekeken of de browser de nieuwere functie `requestFullScreen()` al kent. Als dat zo is, spelen deze inline-styles verder geen enkele rol, omdat `requestFullScreen()` alles afhandelt.

Als de browser deze functie nog niet kent, wordt aan het <video>-element waarbij de knop Fullscreen is ingedrukt, deze inline-style toegevoegd:

```
style="width: 100%; height: 100%; position:
    fixed; top: 30px; left: 0px; z-index:
    137720;"
```

Video afspelen op een breedte en hoogte van 100%. Fixed gepositioneerd, waardoor deze breedte en hoogte gelden ten opzichte van het venster van de browser. Aan de bovenkant een ruimte van 30 px leeg laten voor een eventuele tekst of zoiets, links tegen de zijkant van het browservenster zetten. (Je zou ook de ruimte tot bovenaan kunnen gebruiken voor de video en een melding laten verdwijnen met css. Maar dat is geen goed idee, want browsers

die `requestFullScreen()` niet kennen, zullen ook geen css als `animate` kennen.)
De ietwat eigenaardige z-index (`z-index` mag niet in JavaScript, daar heet het `zIndex`) is een trucje. Als van fullscreen weer wordt teruggegaan naar normale weergave, moet een aantal dingen worden hersteld, zoals breedte en hoogte van de videospeler. Eén van de manieren om van fullscreen terug te gaan naar normaal, is het indrukken van Escape. Om technische redenen werkt die Escape-toets overal, op de hele pagina. Maar als iemand ergens linksboven per ongeluk Escape indrukt, is het niet de bedoeling dat de video's opeens over het scherm gaan polonaisen, hoe gezellig sommige mensen de polonaise ook vinden.
Daarom wordt een test uitgevoerd als Escape wordt ingedrukt. Alleen als de positie van de video, waarmee het laatst is gewerkt, fixed is, én als de z-index daarvan 137720 is, worden deze waarden teruggezet.
Het maximum aantal z-indexen is niet precies bekend, maar het is minimaal vele tientallen miljoenen. Dus de kans dat dit getal toevallig ook in de gewone css wordt gebruikt, in combinatie met een fixed positie, is aanzienlijk kleiner dan de kans dat je tien keer op rij de jackpot in de Staatsloterij wint. Of hoe dat ding ook heet.

```
video.style.width = width;
video.style.height = height;
video.style.position = position;
video.style.top = top;
video.style.left = left;
video.style.zIndex = zIndex;
```

De tegenhanger van het verhaal hier gelijk boven. Het grootste deel van de uitleg staat daar. Als `requestFullScreen()` niet werkte, worden deze waarden teruggezet bij de video, die fullscreen is afgespeeld.
In `width`, `height`, `position`, `top`, `left` en `zIndex` (zogenaamde 'variabelen') zijn de waarden van breedte, hoogte, positie, top, left en z-index van het het <video>-element opgeslagen, zoals die waren gelijk voordat fullscreen werd afgespeeld.
Dit zijn waarden die, als ze zijn gebruikt, in de gewone externe stylesheet staan. Omdat deze netjes worden teruggezet, zullen de tijdelijke waarden die bij fullscreen afspelen zijn opgegeven, hiermee niet botsen.
(Deze serie inline-styles komt twee keer in het script voor, maar dat maakt verder voor de werking niet uit. In beide gevallen wordt de oude hoogte enz. als inline-style teruggezet. Voor de volledigheid: één keer is voor als Escape wordt ingedrukt, de andere is voor wanneer op de fullscreen-video wordt geklikt of wanneer deze wordt aangeraakt.)

CSS DIE ALLEEN WORDT INGEVOEGD, ALS DAT NODIG IS

```
if (getComputedStyle(soundSliderBeamNodeList[i]).position
    === "static")
    soundSliderBeamNodeList[i].style.position =
    "relative";
if (getComputedStyle(imageSliderBeamNodeList[i]).position
    === "static")
    imageSliderBeamNodeList[i].style.position =
    "relative";
```

Twee regels die hetzelfde doen, alleen met een ander element. `soundSliderBeamNodeList` is een soort lijst met alle 's met balken van de sleepbalk voor geluid, `imageSliderBeamNodeList` is datzelfde voor de 's met de balken van de sleepbalk voor afspelen. De balken van de sleepbalk zijn een . Daarbinnen staan de knoppen van de sleepbalk, die ook een zijn. Die knoppen worden absoluut gepositioneerd ten opzichte van de met de balk. Dat kan alleen, als die met de balk balk zelf ook gepositioneerd is. Daarom wordt met behulp van `getComputedStyle` gekeken of de met de balk toevallig een statische positie heeft (dus niet absoluut of fixed of zo is gepositioneerd). Als dat zo is, kan de knop niet worden gepositioneerd ten opzichte van de balk. In dat geval wordt daarom een relatieve positie aan de balk gegeven:

```
style="position: relative;"
```

Omdat dit een relatieve positie is en verder niets wordt opgegeven, heeft dit verder geen invloed op de weergave van de met de balk.

Overzicht van eventlisteners en daardoor aangeroepen functies

In de eerste kolom staat het element, waaraan de eventlistener wordt toegevoegd. Als het om een knop of zo gaat, wordt tussen aanhalingstekens de class toegevoegd, zodat duidelijk is om welke knop e.d. het precies gaat. (Hierbij wordt er vanuit gegaan dat de namen van classes bovenin het script niet zijn veranderd. Is dat wel zo, dan moet je hieronder natuurlijk andere classes lezen.) De aanroep gebeurt vaak vanuit 'n nodelist, waarin het element zit, of iets als Array.prototype.map.call. Maar hier staat alleen het element, waar de aanroep uiteindelijk terecht komt. Als er meerdere gelijksoortige elementen zijn, zoals <video>, de Speel-pauzeerknop, enz., krijgen deze allemaal een eigen eventlistener (waarbij de aangeroepen functie wel steeds dezelfde naam heeft).
In de tweede kolom staat de event.
In de derde kolom staat de aangeroepen functie. Als het om een anonieme functie gaat, wordt (een deel van) de code toegevoegd, zodat er op gezocht kan worden.

Element, knop, e.d. met class of id	Event	Aangeroepen functie of (een deel van) de code
button ”duration-button”	blur	{textParent.removeChild(textParent.lastChild); (…) text = null;})
	click	{textParent.removeChild(textParent.lastChild); (…) text = null;})
	keyup	{if (e.keyCode !== 27) return false; (…) text = null;})
button ”five-back”	keydown	keyBackForward
	keyup	showDurationText
	pointerdown	pointerBackForward
	pointerout	stopPointerRepeat
button ”five-forward”	keydown	keyBackForward
	keyup	showDurationText
	pointerdown	pointerBackForward
	pointerout	stopPointerRepeat
button ”fullscreen”	click	fullscreen
button ”kiezen-knop”	change	{this.setAttribute(”aria-checked”, true); (…) document.body.setAttribute(”style”, ”zoom: 1”);} of: {this.setAttribute(”aria-checked”, true); (…) document.body.setAttribute(”style”, ”zoom: 1.0001”);}
button ”louder”	keydown	keySofterLouder
	pointerdown	pointerSofterLouder
	pointerout	stopPointerRepeat
button ”mute”	click	mute
button ”play”	click	playPause
button ”softer”	keydown	keySofterLouder
	pointerdown	pointerSofterLouder
	pointerout	stopPointerRepeat
button ”speed-half”	change	changeSpeed
button ”speed-default”	change	changeSpeed
button ”speed-one-half”	change	changeSpeed

Element, knop, e.d. met class of id	Event	Aangeroepen functie of (een deel van) de code
button ”speed-double”	change	changeSpeed
button ”ten-back”	keydown	keyBackForward
	keyup	showDurationText
	pointerdown	pointerBackForward
	pointerout	stopPointerRepeat
button ”ten-forward”	keydown	keyBackForward
	keyup	showDurationText
	pointerdown	pointerBackForward
	pointerout	stopPointerRepeat
button ”to-begin”	click	toBegin
button ”to-end”	click	toEnd
div ”controls”	contextmenu	noContext
div ”image-slider-beam”	keydown	imageSliderKey
	pointerdown	imageBeginSliding
		imageBeginSliding
	pointermove	imageSliding
	touchmove	imageSliding
div ”image-slider-button”	pointerdown	imageBeginSliding
	pointermove	imageSliding
	touchmove	imageSliding
div ”sound-slider-beam”	keydown	soundSliderKey
	pointerdown	soundBeginSliding
	pointermove	soundSliding
	touchmove	soundSliding
div ”sound-slider-button”	pointerdown	soundBeginSliding
	pointermove	soundSliding
	touchmove	soundSliding

Element, knop, e.d. met class of id	Event	Aangeroepen functie of (een deel van) de code
document	DOMContentLoaded	cssvCustomControls
	keydown	{clearTimeout(ariaSpeakTimeLater); clearTimeout(ariaSpeakVolumeLater);}
		nextVideo
		{if (e.keyCode === 27 && getComputedStyle(video).position === ”fixed” && parseInt(getComputedStyle(video).zIndex, 10) === 137720) (...) fullscreenNodeList[indexNr].setAttribute(”data-fullscreen”, ”off”);}}
	keyup	{ariaSpeakTimeLater = setTimeout(speakTime, 100); ariaSpeakVolumeLater = setTimeout(speakVolume, 100);}
	pointerdown	{clearTimeout(ariaSpeakTimeLater); clearTimeout(ariaSpeakVolumeLater);}
		{setTimeout(function() {pointerWindowedVideo(e, indexNr, video, width, height, position, top, left, zIndex);}, 500);}
	pointerup	{soundSlidingAllowed = -1; (…) ariaSpeakVolumeLater = setTimeout(speakVolume, 100);}
		stopPointerRepeat
p ”duration-par”	keydown	{if (e.keyCode === 16) { textParent.removeChild(textParent.lastChild); (…) text = null;}});
	keyup	{if (e.keyCode !== 27) return false; (…) text = null;})
video ”video”	durationchange	{setTimeout(function() {showDuration(indexNr);}, 1);}
	ended	finished
	playing	{duration = toMinutes(Math.round(videoNodeList[indexNr].duration)); (…) (videoNodeList[indexNr].duration > 3600) ? ”yes” : ”no”);}

Element, knop, e.d. met class of id	Event	Aangeroepen functie of (een deel van) de code
window	load	{setTimeout(function() {afterPageLoaded();}, 1);}
		{setTimeout(function() {obtainDurationAgain();}, 1000);}

Alfabetisch overzicht van functies

Functie	Korte omschrijving
addControls	Bedieningselementen toevoegen aan element waar <video> in staat
afterPageLoaded	Wordt uitgevoerd na laden pagina, wanneer op essentiële css e.d. kan worden getest
changeSpeed	Stuurt snelheidsregeling
checkValue	Controleren of onmisbare id's e.d. aanwezig zijn
cssvCustomControls	Functie waarbinnen de hele handel staat
finished	Als video aan einde is gekomen
fullscreen	Fullscreen afspelen
getPositionX	x-coördinaat van linkerzijkant element verkrijgen
imageBeginSliding	Stuurt beginnen slepen van sleepbalk voor afspelen
imageSliderKey	Stuurt toetsbediening sleepbalk beeld
imageSliding	Stuurt voortzetten slepen sleepbalk afspelen
keyBackForward	Stuurt toetsbediening knoppen voor- en achteruit
keySofterLouder	Stuurt toetsbediening knoppen zachter en harder
makeChoose	Radioknoppen e.d. voor kiezen standaard- of aangepaste videospelers maken
makeElement	Eén specifiek element maken
makeWrapperDivNodeList	nodelist met elementen waarin <video> staat maken
mute	Aan-uitknop voor geluid
nextVideo	Met sneltoets naar vorige/volgende video gaan
noContext	Contextuele menu blokkeren binnen bediening videospeler
obtainDurationAgain	Speelduur na 1 seconde nogmaals opvragen
playPause	Afspelen/pauzeren
pointerBackForward	Stuurt pointer knoppen voor- en achteruit
pointerSofterLouder	Stuurt pointer knoppen zachter en harder
pointerWindowedVideo	Video weer op normale formaat weergeven voor browsers die requestFullScreen niet kennen
showDuration	Speelduur op scherm zetten
showDurationLater	Handelt speelduur af, tot opvragen daarvan is gelukt
showDurationText	Regelt openen/sluiten melding speelduur nog onbekend

Functie	Korte omschrijving
showPercentageKey	Stelt geluidssterkte in en toont percentage geluid. Wordt aangestuurd door toetsen geluid en pointer op harder/zachter
showPercentageSliding	Stelt geluidssterkte in en toont percentage, aangestuurd door sleepbalk geluid
showTime	Opvragen en weergeven van verstreken en resterende speelduur
showTimePointer	Past weergave tijd aan, als pointer blijft ingedrukt
soundBeginSliding	Stuurt beginnen slepen sleepbalk geluid
soundSliderKey	Stuurt toetsbediening sleepbalk geluid
soundSliding	Stuurt voortzetten slepen sleepbalk geluid
speakTime	Leest nieuw ingestelde verstreken speelduur voor
speakVolume	Leest nieuw ingestelde geluidssterkte in procenten voor
startTime	Opvragen en weergeven van verstreken en resterende speelduur starten
stopOthers	Afspelen van alle video's, behalve de gekozen, stoppen
stopPointerRepeat	Stuurt stoppen van herhalingen, als pointer omlaag is
toBegin	Naar begin video gaan
toEnd	Naar einde video gaan
toMinutes	Seconden omzetten naar uren, minuten en seconden
twoDigits	Getal van 1 cijfer vooraf laten gaan door 0

Het JavaScript onderaan de html-bestanden

Onderaan de derde, vierde en vijfde pagina met videospelers staan kleine JavaScriptjes. Dat heeft te maken met de hoeveelheid verschillende videospelers op dezelfde pagina. Normaal genomen zullen videospelers er hetzelfde uitzien en zijn deze scriptjes niet of veel minder nodig.
In de scriptjes worden twee dingen veranderd: de tabindex en de aria-code `role="group"`. Dit heeft te maken met de volgorde van de bedieningselementen van de videospelers op het scherm. Deze wijkt af van de volgorde in de html. Met behulp van deze scriptjes worden ze weer gelijk getrokken. Beter is trouwens om gewoon de volgorde van de html aan te houden, dan heb je dit gedoe helemaal niet nodig.
Je kunt de tabindex ook bovenin het script veranderen, zoals beschreven bij TabIndex (wijzigen). Maar die veranderingen gelden dan voor álle videospelers op dezelfde bladzijde. Met de scriptjes onderaan een html-bestand kunnen ook individuele spelers worden gewijzigd.
Hier staat alleen maar beschreven, hoe je tabindex en aria-codes kunt wijzigen. Hoe de tabindex werkt, staat bij Tabindex (uitleg). Hoe aria-codes werken, staat bij WAI-ARIA-codes binnen de videospeler.

Anders dan in het aparte script is in deze kleine scriptjes geen enkele foutcontrole ingebouwd. Dat zou ze veel ingewikkelder maken. En in dit geval maakt het weinig uit, want in het ergste geval wordt een tabindex of aria-code niet aangepast, verder blijft alles gewoon werken.

Als op ’n andere pagina de tabindex moet worden aangepast, kan een van de scriptjes gewoon worden gekopieerd, waarbij alleen de nieuw te gebruiken tabindex moet worden opgegeven, zoals hieronder omschreven.

Overigens is het verreweg het beste om de volgorde op het scherm (vrijwel) hetzelfde te houden als de volgorde in de html. Dan hoef je helemaal niet te sleutelen aan de tabindex, wat het beste is voor de toegankelijkheid.

Het scriptje onderaan pagina vijf

Dit is de makkelijkste van de drie, want hier wordt alleen de tabindex aangepast, dus met deze begin ik.

```
<script>
```

Gewoon de openingstag van een JavaScript.

```
document.addEventListener("DOMContentLoaded",
    correctTabIndex);
```

Dit vertelt de browser, dat dit script pas moet worden uitgevoerd, als de pagina helemaal is geladen. (Feitelijk: als de DOM, het schema waar de browser intern mee werkt, is opgebouwd.) Je kunt pas iets wijzigen, als het aanwezig is. Deze regel wordt nooit gewijzigd.

```
function correctTabIndex() {
```

Dit is het begin van het uitvoerende deel van het script. Ook dit hoeft nooit gewijzigd te worden.

```
var changeTabIndex = [
```

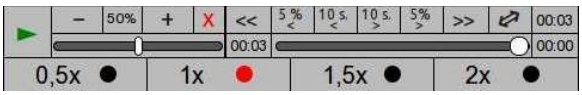
Dit is om een lijst in op te slaan van elementen, waarvan de tabindex moet worden gewijzigd, en van de bijbehorende nieuwe tabindex. Ook dit hoeft niet gewijzigd te worden.

Nu komt het deel, dat eventueel aangepast kan worden. Op pagina vijf wordt alleen de tabindex aangepast.

```
"speed-half-3", 314,
"speed-default-3", 314,
"speed-one-half-3", 314,
"speed-double-3", 314,
"image-slider-beam-3", 313,
```

Voor de eerste komma staat, tussen aanhalingstekens, de id van het element waarvan de tabindex moet worden aangepast. De id eindigt op ’-3’, waaraan is te zien dat deze id’s bij de derde video horen. Om precies te zijn: bij de knoppen die de snelheid regelen, en bij de sleepbalk voor afspelen.
(Als je bovenin het eigenlijke script de namen van de id’s hebt veranderd, moet je hier natuurlijk ook die nieuwe namen lezen in plaats van de hier gebruikte standaardnamen.)
Achter de eerste komma staat de nieuwe tabindex. Dit is een getal, en daarom hoeft dat niet tussen aanhalingstekens. Achter de tabindex staat weer een komma, zodat de browser weet dat er nog meer volgt. De
knoppen die de snelheid regelen zijn radioknoppen, daarom krijgen die dezelfde tabindex.

De originele tabindex was precies omgekeerd, waardoor de sleepbalk voor afspelen dus eerder werd bezocht dan de knoppen voor snelheidsregeling. Terwijl die



Zonder correctie zou bij de derde video, bij gebruik van de Tab-toets, de snelheidsregeling vóór de sleepbalk voor weergave worden bezocht. Wat een tamelijk onlogische volgorde is.

sleepbalk op het scherm boven de snelheidsregeling staat.

De tabindex van de bedieningselementen van de eerste video is een getal van 101 of hoger, van de tweede 201 of hoger, enz. Hoe dit getal precies wordt samengesteld, is te lezen bij Tabindex. Het beste kun je even in de gegenereerde code de huidige tabindex bekijken, dan zie je makkelijk welke er eventueel aangepast moeten worden.

Bij gebruik van de Tab-toets wordt nu de volgorde gevolgd, waarin de bedieningselementen van de videospeler op het scherm staan, en niet de volgorde waarin ze in de html staan.

Dit zijn de veranderingen die bij de vierde videospeler horen:

```
"image-slider-beam-4", 412,
"speed-half-4", 413,
"speed-default-4", 413,
"speed-one-half-4", 413,
"speed-double-4", 413,
```

Het verhaal is precies hetzelfde als hierboven voor de derde videospeler. Maar omdat dit de vierde speler is, eindigen de id's op '-4' en beginnen de tabindexen met '4'. Hier staat 'image-slider-beam-4' boven de id's van de radioknoppen voor de snelheidsregeling, maar die andere volgorde maakt niets uit voor de werking. Bij de derde speler heb ik de volgorde op het scherm gevolgd, hier is de volgorde in de html gevolgd.

Ook bij de vijfde en zesde videospeler is een kleine aanpassing nodig:

```
"speed-half-5", 512,
"speed-default-5", 512,
"speed-one-half-5", 512,
"speed-double-5", 512,
"fullscreen-5", 513,

"speed-half-6", 612,
"speed-default-6", 612,
"speed-one-half-6", 612,
"speed-double-6", 612,
"fullscreen-6", 613
```

Bij deze videospelers worden snelheidsregeling en knop om fullscreen af te spelen omgewisseld. Voor de rest is alles precies hetzelfde, op een klein detail na: na de laatste gewijzigde tabindex staat geen komma. Die komma aan het eind van de regel is alleen nodig, zodat de browser weet dat er nog meer volgt. Omdat dit de laatste is, is die komma hier dus niet nodig.

Dan volgt nog het afsluitende deel van het scriptje. Ook hier hoeft niets aan gewijzigd te worden:

```
],
```

Om de browser duidelijk te maken dat hier het eind van de lijst met id's en tabindexen is bereikt.

```
len = changeTabIndex.length;
for (var i = 0; i < len; i++) {
     document.getElementById(changeTabIndex[i]).
          tabIndex = changeTabIndex[++i];
}
```

Dit stukje gruwelijkheden werkt één voor één alle id's en bijbehorende nieuwe tabindexen af. Maakt niet uit hoeveel of hoe weinig er gewijzigd moet worden, de hele lijst wordt gewoon afgewerkt.

```
</script>
```

Afsluiting van het script. (Twee accolades zijn weggelaten, vanwege de duidelijkheid.)

Het scriptje onderaan pagina vier

Op deze pagina staan de bedieningselementen voor geluid en beeld door elkaar heen. Maar in de code is voor schermlezers aangegeven dat het twee groepen zijn. Dat is verwarrend, daarom wordt de aria-code voor groepen `group="role"` weggehaald. Omdat een schermlezer de volgorde van de html volgt en niet de volgorde op het scherm, zou je ook kunnen zeggen dat geluid en beweging gewoon groepen kunnen blijven. Voor beide is wat te zeggen. Waaruit maar weer blijkt, dat het echt veel beter is in de html dezelfde volgorde aan te houden als op het scherm, als dat ook maar enigszins mogelijk is.
Aan dit script is verder eigenlijk weinig te wijzigen. Het wordt gebruikt, of niet gebruikt.

```
<script>
document.addEventListener("DOMContentLoaded",
     correctAttributes);
```

Het begin is precies hetzelfde als dat van het scriptje hierboven voor de vijfde pagina: voer dit script pas uit als de pagina is geladen.

```
function correctAttributes() {
     var removeSoundGroup =
          document.getElementsByClassName("sound"),
     removeImageGroup =
          document.getElementsByClassName("image"),
     len = removeSoundGroup.length;
```

In dit geval moet bij álle videospelers deze groep worden weggehaald. Daarom wordt er nu niet met id's gewerkt, maar met classes. Alle elementen met class=”sound” of ”class=”image” worden in een lijstje gezet.

```
for (var i = 0; i < len; i++) {
     removeSoundGroup[i].removeAttribute("role");
     removeImageGroup[i].removeAttribute("role");
```

De hele lijst wordt doorgelopen (feitelijk twee lijsten, eentje voor het geluid en eentje voor het beeld), en bij elk element wordt de `role` verwijderd.

`</script>`

Afsluiting van het script. (Twee accolades zijn weggelaten, vanwege de duidelijkheid.)

Het scriptje onderaan pagina drie

Op de derde pagina wordt de tabindex aangepast én de `role="group"` wordt verwijderd. Maar niet bij elke speler, dus nu moet voor het verwijderen van de groep met id's worden gewerkt. Dit scriptje is 'n soort combinatie van dat op de vierde en vijfde pagina.

Het begin is weer hetzelfde:

```
<script>
document.addEventListener("DOMContentLoaded",
    correctAttributes);
```

Voer het script pas uit, als de pagina is geladen. Dit hoeft nooit gewijzigd te worden.

```
function correctAttributes() {
```

Dit is het begin van het uitvoerende deel van het script. Ook dit hoeft nooit gewijzigd te worden.

```
var removeGroup = [
```

Dit is om een lijst in op te slaan van elementen, waarbij `role="group"` moet worden verwijderd.

```
"sound-1",
"image-1",
"sound-2",
"image-2",
"sound-5",
"image-5",
"sound-6",
"image-6"
```

Dit zijn de id's van de 's, waarbinnen de bedieningselementen voor geluid en beeld staan. Anders dan hierboven bij de vierde pagina, moet niet bij alle videospelers de aria-code voor groep worden weggehaald. Aan de volgnummers is te zien dat het hier om de eerste, tweede, vijfde en zesde speler gaat.

```
],
```

Om de browser duidelijk te maken dat hier het eind van de lijst met id's van elementen, waar de aria-code voor groep moet worden verwijderd, is bereikt.

Ook de tabindex moet bij sommige videospelers op deze pagina worden aangepast. Dat gaat op dezelfde manier als hierboven bij de vijfde pagina is beschreven.

```
changeTabIndex = [
```

Dit is om een lijst in op te slaan van elementen, waarvan de tabindex moet worden gewijzigd, en van de bijbehorende nieuwe tabindex. Ook dit hoeft niet gewijzigd te worden.

Id's van elementen waar de tabindex moet worden gewijzigd, staan tussen aanhalingstekens voor de eerste komma, de nieuwe tabindex staat achter de eerste komma, gevolgd door een afsluitende komma aan het eind van de regel. Alleen bij de laatste regel hoeft geen afsluitende komma:

```
"five-back-2", 206,
"five-forward-2", 207,
"ten-back-2", 208,
"ten-forward-2", 209,
"sound-slider-beam-2", 210,
"image-slider-beam-2", 211,
"to-begin-2", 212,
"to-end-2", 213,
"fullscreen-2", 214,
"speed-half-2", 215,
"speed-default-2", 215,
"speed-one-half-2", 215,
"speed-double-2", 215,

"to-begin-4", 402,
"to-end-4", 403,
"ten-back-4", 404,
"ten-forward-4", 405,
"softer-4", 406,
"louder-4", 407,
"mute-4", 408,

"five-back-5", 506,
"five-forward-5", 507,
"ten-back-5", 508,
"ten-forward-5", 509,
"sound-slider-beam-5", 510,
"image-slider-beam-5", 511,
"to-begin-5", 512,
"to-end-5", 513,
"fullscreen-5", 514,
"speed-half-5", 515,
"speed-default-5", 515,
"speed-one-half-5", 515,
"speed-double-5", 515
```

Het zijn wat meer tabindexen die moeten worden veranderd dan op de vijfde pagina hierboven, maar het principe is precies hetzelfde.

Dan volgt nog het afsluitende deel van het scriptje. Ook hier hoeft nooit iets aan gewijzigd te worden:

```
],
```

Om de browser duidelijk te maken dat hier het eind van de lijst met id's en tabindexen is bereikt.

```
len = changeTabIndex.length;
for (var i = 0; i < len; i++) {
     document.getElementById(changeTabIndex[i])
          .tabIndex = changeTabIndex[++i];
}

len = removeGroup.length;
for (i = 0; i < len; i++) {
     document.getElementById(removeGroup[i]).
          removeAttribute("role");
```

Deze code werkt eerst één voor één alle id's en bijbehorende nieuwe tabindexen af. Maakt niet uit hoeveel of hoe weinig er gewijzigd moet worden, de hele lijst wordt gewoon afgewerkt. Vervolgens wordt de lijst met elementen, waar de aria-code voor groep moet worden verwijderd, afgewerkt.

```
</script>
```

Afsluiting van het script. (Twee accolades zijn weggelaten, vanwege de duidelijkheid.)

Volledige code

HTML

De html die te maken heeft met de basis van dit voorbeeld is rood gekleurd. Alle niet-essentiële code html in een afwijkende `zwarte` lettersoort. (In de inhoudsopgave staat alles in een gewone zwarte letter vanwege de leesbaarheid.)

De hieronder staande html is een kopie van afbeelding-103-1-dl.html. De html voor de andere pagina's zit wel in de download, maar is hieronder niet weergegeven. De enige verschillen met die andere pagina's zijn – afhankelijk van de pagina – het volgnummer van de tabindex, de link naar bijbehorend css-bestand, de link naar bijbehorend JavaScript en het scriptje onderaan de pagina. En uiteraard de beschrijving van de verschillende videospelers bovenin de pagina.

```
<!DOCTYPE html>
<html lang="nl">
<head>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-
          scale=1">
     <meta name="description" content="Volledig met css aan te
          passen videospelers - voorbeeld: pagina 1">
```

```
    <title>Volledig met css aan te passen videospelers -
        voorbeeld: pagina 1</title>
    <link rel="stylesheet" href="../103-css-dl/afbeelding-103-1-
        dl.css">
</head>
<body>
<main role="main">
    <h1>Volledig met <abbr lang="en" title="Cascading Style
        Sheets">css</abbr> aan te passen videospelers - pagina
        1</h1>
    <nav id="links" role="navigation">
        <h2>Navigatie door voorbeeld</h2>
        <a href="afbeelding-103-5-dl.html" tabindex="0">Pagina
            5</a> | <a href="../afbeelding-103-dl.html"
            tabindex="0">Overzicht</a> | <a href="afbeelding-
            103-2-dl.html" tabindex="0">Pagina 2</a>
    </nav>
    <p>Alle bedieningselementen gebruikt. Alleen op iOS: geen
        geluidsregeling. Binnen de knoppen zijn alleen gewone
        karakters uit een standaardfont gebruikt.</p>
    <aside id="wrapper-help" role="complementary" tabindex="-1">
        <span id="focus-for-tab" aria-hidden="true"
            tabindex="0"></span>
        <input id="checkbox-help" type="checkbox" aria-
            hidden="true">
        <div id="wrapper-icons" aria-hidden="true">
        <label id="icons" for="checkbox-help" title="Openen of
            sluiten van het hulpschermpje"
            accesskey="8"></label>
        </div>
        <div id="helptext">
            <h2>Werking van de aangepaste besturing</h2>
            <h3>Algemene toetsen</h3>
            <p>Openen/sluiten van dit hulpschermpje (buiten de
                knoppen met vraagteken/sluitkruisje): Alt+8,
                Alt+Shift+8 of Alt+Control+8 (afhankelijk van
                browser, werkt niet in Internet Explorer).</p>
            <p>Naar vorige/volgende video: Control+←/→ of
                Control+Alt+←/→ (afhankelijk van browser)</p>
            <p>Naar vorige/volgende knop: Tab/Shift+Tab</p>
            <h3>Knoppen</h3>
            <p>Alle knoppen werken zoveel mogelijk op de
                standaardmanier.</p>
            <p>Afspelen/pauzeren/geluid aan/uit: omschakelen
                door aanraken, klikken of Enter/Spatie.</p>
```

```
            <p>Harder/zachter/5% voor-/achteruit/10 seconden
                voor-/achteruit: idem. Als je blijft aanraken,
                klikken of Enter/Spatie blijft indrukken, gaat
                de knop repeteren.</p>
            <p>Naar begin/einde: aanraken, klikken of
                Enter/Spatie.</p>
            <p>Volledig scherm: openen door aanraken, klikken of
                Spatie/Enter. Sluiten volgens aanwijzing.</p>
            <p>Snelheid wijzigen: aanraken, klikken of
                pijltjestoetsen.</p>
            <h3>Sleepbalken</h3>
            <p>Slepen door aanraken van de knop met vinger of
                muis (als de muis per ongeluk buiten de
                sleepbalk komt, kan worden teruggegaan, zolang
                de muisknop blijft ingedrukt).</p>
            <p>Naar bepaald volume/tijdstip springen: sleepbalk
                aanraken/aanklikken op gewenste
                volume/tijdstip.</p>
            <p>Als de sleepbalk focus heeft (balk is wit, knop
                is rood), werken de volgende toetsen (sommige
                toetsen gaan repeteren, als ze blijven
                ingedrukt):</p>
            <p>Geluid: 1% harder/zachter: pijltjestoetsen; 10%
                harder/zachter: PgUp/PgDn; 0% (uit): Home;
                100%: End.</p>
            <p>Beeld: 10 seconden terug/vooruit:
                pijltjestoetsen; 5% voor-/achteruit: PgUp/PgDn;
                Begin: Home; Einde: End.</p>
        </div>
    </aside>
    <div class="videobox">
        <h2 id="titel-1" lang="en">S. Block - Cellular
            Respiration - B Block</h2>
        <p class="origineel">Pagina met <a
            href="https://archive.org/details/S.Brock-
            CellularRespiration-BBlock" title="Pagina met
            originele (onverkorte) video" tabindex="0">originele
            (onverkorte) video</a>.</p>
        <video data-smscr="480" aria-describedby="titel-1"
            controls preload="metadata" poster="../103-
            images/s_brock_cellular_respiration.jpg">
            <source src="../103-video/s_brock_cellular_
                respiration.webm" type="video/webm">
```

```
            <source data-smscr="480" src="../103-video/s_brock_
                cellular_respiration_klein.webm"
                type="video/webm">
            <source src="../103-video/s_brock_cellular_
                respiration.mp4" type="video/mp4">
            <source data-smscr="480" src="../103-video/
                s_brock_cellular_respiration_klein.mp4"
                type="video/mp4">
            Je browser ondersteunt het afspelen van video's
                niet. Je kunt hieronder de video nog wel
                downloaden.
        </video>
        <p class="groot">Video downloaden? Kies <a lang="en"
            href="../103-video/s_brock_cellular_respiration.mp4"
            download="S. Brock - Cellular Respiration - B
            Block.mp4" title="Download video in mp4-formaat"
            tabindex="0">mp4</a> <span>(12,4 <abbr
            title="megabyte">MB </abbr>)</span> of <a lang="en"
            href="../103-ideo/s_brock_cellular_respiration.webm"
            download="S. Brock - Cellular Respiration - B
            Block.webm" title="Download video in webm-formaat"
            tabindex="0">webm</a> <span>(21,6 <abbr>MB</abbr>)
            </span>.</p>
        <p class="klein">Video downloaden? Kies <a lang="en"
            href="../103-video/s_brock_cellular_respiration_
            klein.mp4" download="S. Brock - Cellular Respiration
            - B Block.mp4" title="Download video in mp4-formaat"
            tabindex="0">mp4</a> <span>(704,4 <abbr
            title="kilobyte">kB</abbr>)</span> of <a lang="en"
            href="../103-video/s_brock_cellular_respiration_
            klein.webm" download="S. Brock - Cellular
            Respiration - B Block.webm" title="Download video in
            webm-formaat" tabindex="0">webm</a> <span>(1,1
            <abbr>MB</abbr>)</span>.</p>
    </div>
    <div class="videobox">
        <h2 id="titel-2" lang="en">Elliot B senior project with
            after effects</h2>
        <p class="origineel">Pagina met <a
            href="https://archive.org/details/ElliotBSeniorProje
            ctWithAfterEffects" title="Pagina met originele
            (onverkorte) video" tabindex="0">originele
            video</a>.</p>
```

```
        <video data-smscr="480" aria-describedby="titel-2"
            controls preload="metadata" poster="../103-images/
            elliot_b_senior_project_with_after_effects.jpg">
            <source src="../103-video/elliot_b_senior_project_
                with_after_effects.webm" type="video/webm">
            <source data-smscr="480" src="../103-video/elliot_
                b_senior_project_with_after_effects_klein.webm"
                type="video/webm">
            <source src="../103-video/elliot_b_senior_project_
                with_after_effects.mp4" type="video/mp4">
            <source data-smscr="480" src="../103-video/
                elliot_b_senior_project_with_after_effects_klei
                n.mp4" type="video/mp4">
            Je browser ondersteunt het afspelen van video's
                niet. Je kunt hieronder de video nog wel
                downloaden.
        </video>
        <p class="groot">Video downloaden? Kies <a lang="en"
            href="../103-video/elliot_b_senior_project_with_
            after_effects.mp4" download="Elliot B senior project
            with after effects.mp4" title="Download video in
            mp4-formaat" tabindex="0">mp4</a> <span>(1,6
            <abbr>MB</abbr>)</span> of <a lang="en"
            href="../103-video/elliot_b_senior_project_with_
            after_effects.webm" download="Elliot B senior
            project with after effects.webm" title="Download
            video in webm-formaat" tabindex="0">webm</a>
            <span>(2,1 <abbr>MB</abbr>)</span>.</p>
        <p class="klein">Video downloaden? Kies <a lang="en"
            href="../103-video/elliot_b_senior_project_with_
            after_effects_klein.mp4" download="Elliot B senior
            project with after effects.mp4" title="Download
            video in mp4-formaat" tabindex="0">mp4</a> <span>
            (1,1 <abbr>MB</abbr>)</span> of <a lang="en"
            href="../103-video/elliot_b_senior_project_with_
            after_effects_klein.webm" download="Elliot B senior
            project with after effects.webm" title="Download
            video in webm-formaat" tabindex="0">webm</a>
            <span>(1,4 <abbr>MB</abbr>)</span>.</p>
    </div>
    <div class="videobox">
        <h2 id="titel-3" lang="en">Photography of Music by
             Victoria Holt</h2>
```

```
        <p class="origineel">Pagina met <a href="https://
            archive.org/details/PhotographyOfMusicByVictoriaHolt
            " title="Pagina met originele (onverkorte) video"
            tabindex="0">originele (onverkorte) video</a>.</p>
        <video data-smscr="480" aria-describedby="titel-3"
            controls preload="metadata" poster="../103-
            images/photography_of_music_by_victoria_holt.jpg">
            <source src="../103-video/photography_of_music_by_
                victoria_holt.webm" type="video/webm">
            <source data-smscr="480" src="../103-video/
                photography_of_music_by_victoria_holt_klein.web
                m" type="video/webm">
            <source src="../103-video/photography_of_music_by_
                victoria_holt.mp4" type="video/mp4">
            <source data-smscr="480" src="../103-video/
                photography_of_music_by_victoria_holt_klein.mp4
                " type="video/mp4">
            Je browser ondersteunt het afspelen van video's
                niet. Je kunt hieronder de video nog wel
                downloaden.
        </video>
        <p class="groot">Video downloaden? Kies <a lang="en"
            href="../103-video/photography_of_music_by_
            victoria_holt.mp4" download="Photography of Music by
            Victoria Holt.mp4" title="Download video in mp4-
            formaat" tabindex="0">mp4</a> <span>(212,2
            <abbr>kB</abbr>)</span> of <a lang="en"
            href="../103-video/photography_of_music_by_
            victoria_holt.webm" download="Photography of Music
            by Victoria Holt.webm" title="Download video in
            webm-formaat" tabindex="0">webm</a> <span>(262,4
            <abbr>kB</abbr>)</span>.</p>
        <p class="klein">Video downloaden? Kies <a lang="en"
            href="../103-video/photography_of_music_by_
            victoria_holt_klein.mp4" download="Photography of
            Music by Victoria Holt.mp4" title="Download video in
            mp4-formaat" tabindex="0">mp4</a> <span>(576,6
            <abbr>kB</abbr>)</span> of <a lang="en"
            href="../103-video/photography_of_music_by_
            victoria_holt_klein.webm" download="Photography of
            Music by Victoria Holt.webm" title="Download video
            in webm-formaat" tabindex="0">webm</a> <span>(475,2
            <abbr>kB</abbr>)</span>.</p>
    </div>
    <div class="videobox">
```

```
<h2 id="titel-4" lang="en">How to Drive Fast with David
     Rodriguez</h2>
<p class="origineel">Pagina met <a href="https://
     archive.org/details/HowToDriveFastWithDavidRodriguez
     " title="Pagina met originele (onverkorte) video"
     tabindex="0">originele (onverkorte) video</a>.</p>
<video data-smscr="480" aria-describedby="titel-4"
     controls preload="metadata" poster="../103-
     images/how_to_drive_fast_with_david_rodriguez.jpg">
     <source src="../103-video/how_to_drive_fast_with_
          david_rodriguez.webm" type="video/webm">
     <source data-smscr="480" src="../103-video/
          how_to_drive_fast_with_david_rodriguez_klein.we
          bm" type="video/webm">
     <source src="../103-video/how_to_drive_fast_with_
          david_rodriguez.mp4" type="video/mp4">
     <source data-smscr="480" src="../103-video/how_to_
          drive_fast_with_david_rodriguez_klein.mp4"
          type="video/mp4">
     Je browser ondersteunt het afspelen van video's
          niet. Je kunt hieronder de video nog wel
          downloaden.
</video>
<p class="groot">Video downloaden? Kies <a lang="en"
     href="../103-video/how_to_drive_fast_with_
     david_rodriguez.mp4" download="How to Drive Fast
     with David Rodriguez.mp4" title="Download video in
     mp4-formaat" tabindex="0">mp4</a> <span>(195,6
     <abbr>kB</abbr>)</span> of <a lang="en"
     href="../103-video/how_to_drive_fast_with_david_
     rodriguez.webm" download=="How to Drive Fast with
     David Rodriguez.webm" title="Download video in webm-
     formaat" tabindex="0">webm</a> <span>(167,2
     <abbr>kB</abbr>)</span>.</p>
<p class="klein">Video downloaden? Kies <a lang="en"
     href="../103-video/how_to_drive_fast_with_david_
     rodriguez_klein.mp4" download="How to Drive Fast
     with David Rodriguez.mp4" title="Download video in
     mp4-formaat" tabindex="0">mp4</a> <span>(631,1
     <abbr>kB</abbr>)</span> of <a lang="en"
     href="../103-video/how_to_drive_fast_with_david_
     rodriguez_klein.webm" download="How to Drive Fast
     with David Rodriguez.webm" title="Download video in
     webm-formaat" tabindex="0">webm</a> <span>(818,3
     <abbr>kB</abbr>)</span>.</p>
```

```
</div>
<div class="videobox">
     <h2 id="titel-5" lang="en">Peer Tutoring Program by
          Angelina G</h2>
     <p class="origineel">Pagina met <a
          href="https://archive.org/details/PeerTutoringProgra
          mByAngelinaG" title="Pagina met originele
          (onverkorte) video tabindex="0"">originele
          (onverkorte) video</a>.</p>
     <video data-smscr="480" aria-describedby="titel-5"
          controls preload="metadata" poster="../103-
          images/peer_tutoring_program_by_angelina_g.jpg">
          <source src="../103-video/peer_tutoring_program_
               by_angelina_g.webm" type="video/webm">
          <source data-smscr="480" src="../103-video/peer_
               tutoring_program_by_angelina_g_klein.webm"
               type="video/webm">
          <source src="../103-
               video/peer_tutoring_program_by_angelina_g.mp4"
               type="video/mp4">
          <source data-smscr="480" src="../103-video/peer_
               tutoring_program_by_angelina_g_klein.mp4"
               type="video/mp4">
          Je browser ondersteunt het afspelen van video's
               niet. Je kunt hieronder de video nog wel
               downloaden.
     </video>
     <p class="groot">Video downloaden? Kies <a lang="en"
          href="../103-video/peer_tutoring_program_by_
          angelina_g.mp4" download="Peer Tutoring Program by
          Angelina G.mp4" title="Download video in mp4-
          formaat" tabindex="0">mp4</a> <span>(274,7
          <abbr>kB</abbr>)</span> of <a lang="en"
          href="../103-video/peer_tutoring_program_by_
          angelina_g.webm" download=="Peer Tutoring Program by
          Angelina G.webm" title="Download video in webm-
          formaat" tabindex="0">webm</a> <span>(364,6
          <abbr>kB</abbr>)</span>.</p>
     <p class="klein">Video downloaden? Kies <a lang="en"
          href="../103-video/peer_tutoring_program_by_
          angelina_g_klein.mp4" download="Peer Tutoring
          Program by Angelina G.mp4" title="Download video in
          mp4-formaat" tabindex="0">mp4</a> <span>(543,6
          <abbr>kB</abbr>)</span> of <a lang="en"
          href="../103-video/peer_tutoring_program_by_
```

```
            angelina_g_klein.webm" download="Peer Tutoring
            Program by Angelina G.webm" title="Download video in
            webm-formaat" tabindex="0">webm</a> <span>(576,6
            <abbr>kB</abbr>)</span>.</p>
</div>
<div class="videobox">
     <h2 id="titel-6" lang="en">Making Video a Game with Bill
          Decker</h2>
     <p class="origineel">Pagina met <a
          href="https://archive.org/details/MakingVideoAGameWi
          thBillDecker" title="Pagina met originele
          (onverkorte) video" tabindex="0">originele
          (onverkorte) video</a>.</p>
     <video data-smscr="480" aria-describedby="titel-6"
          controls preload="metadata" poster="../103-
          images/making_video_a_game_with_bill_decker.jpg">
          <source src="../103-video/making_video_a_game_
               with_bill_decker.webm" type="video/webm">
          <source data-smscr="480" src="../103-video/making_
               video_a_game_with_bill_decker_klein.webm"
               type="video/webm">
          <source src="../103-video/making_video_a_game_with_
               bill_decker.mp4" type="video/mp4">
          <source data-smscr="480" src="../103-video/making_
               video_a_game_with_bill_decker_klein.mp4"
               type="video/mp4">
          Je browser ondersteunt het afspelen van video's
               niet. Je kunt hieronder de video nog wel
               downloaden.
     </video>
     <p class="groot">Video downloaden? Kies <a lang="en"
          href="../103-video/making_video_a_game_with_
          bill_decker.mp4" download="Making Video a Game with
          Bill Decker.mp4" title="Download video in mp4-
          formaat" tabindex="0">mp4</a> <span>(333,7
          <abbr>kB</abbr>)</span> of <a lang="en"
          href="../103-video/making_video_a_game_with_bill_
          decker.webm" download="Making Video a Game with Bill
          Decker.webm" title="Download video in webm-formaat"
          tabindex="0">webm</a> <span>(578,9
          <abbr>kB</abbr>)</span>.</p>
     <p class="klein">Video downloaden? Kies <a lang="en"
          href="../103-video/making_video_a_game_with_bill_
          decker_klein.mp4" download="Making Video a Game with
          Bill Decker.mp4" title="Download video in mp4-
```

```
                formaat" tabindex="0">mp4</a> <span>(529,9
                <abbr>kB</abbr></span> of <a lang="en" href="../103-
                video/making_video_a_game_with_bill_decker_klein.web
                m" download=="Making Video a Game with Bill
                Decker.webm" title="Download video in webm-formaat"
                tabindex="0">webm</a> <span>(839,6 <abbr>kB</abbr>)
                </span>.</p>
        </div>
</main>
<script src="../103-js/afbeelding-103.js"></script>
<script src="../103-js/hand.minified-1.3.8.js"></script>
</body>
</html>
```

CSS

Normaal genomen staat hier een kopie van de stylesheet, waarin de voor het voorbeeld essentiële code rood is gekleurd. Dat rood kleuren is hier zinloos, omdat álle css essentieel is (of juist helemaal niets, afhankelijk van hoe je het bekijkt.) Daarmee is het ook zinloos geworden de volledige code hieronder neer te zetten.
In de download zitten 6 stylesheets. afbeelding-103-dl.css hoort bij de overzichtspagina. afbeelding-103-dl-1 t/m afbeelding-103-dl-5 horen bij de video's op pagina 1 t/m pagina 5.

JavaScript

Het script is in principe voor alle pagina's hetzelfde. Maar de plaatsing van de bedieningselementen van de videospelers op het scherm verschilt. Daarom is voor een aantal pagina's de tabindex aangepast. Die tabindex wordt bovenin het script opgegeven. Daarom zijn er meerdere JavaScripts. Als je de tabindex buiten beschouwing laat, is het hetzelfde script dat alle pagina's aanstuurt.
Het is zinloos hier het hele script weer te geven, want er is geen essentiële code of zo om te kleuren. Het script met de naam afbeelding-103-met-commentaar.js is hetzelfde als alle andere scripts (op de tabindex van sommige scripts na dus), maar is voorzien van commentaar: uitleg hoe de code werkt.