

## Geluid afspelen met behulp van het <audio>-element



### BELANGRIJKE INFORMATIE

Alles op deze site kan vrij worden gebruikt, met drie 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](http://www.css-voorbeelden.nl) en medewerkers daarvan op geen enkele manier verantwoordelijk.
- \* 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](http://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.
- \* Het kan zijn dat materiaal is gebruikt dat van anderen afkomstig is. Dat materiaal kan onder een bepaalde licentie vallen, waardoor het mogelijk niet onbeperkt gebruikt mag worden. Als dat zo is, wordt dat vermeld onder [Inhoud van de download en licenties](#).

Een link naar [www.css-voorbeelden.nl](http://www.css-voorbeelden.nl) wordt trouwens altijd op prijs gesteld.

De volledige code vind je helemaal achteraan dit document. Deze code is exact hetzelfde als die van de 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.

Alle code is geschreven in een afwijkende lettersoort. De code die te maken heeft met de basis van dit voorbeeld (essentiële code), is in de hele uitleg **blauw** gekleurd. Alle niet-essentiële code is zwart. (In de inhoudsopgave staat alles vanwege de leesbaarheid in een gewone letter.)

### Inhoudsopgave

- [Korte omschrijving](#)
- [Opmerkingen](#)
- [Achterliggend idee](#)
- [De code aanpassen aan je eigen ontwerp](#)
- [Toegankelijkheid en zoekmachines](#)
- [Getest in](#)
- [Bekende problemen \(en oplossingen\)](#)
- [Wijzigingen](#)
- [Inhoud van de download en licenties](#)

Uitleg code:

[HTML](#) (in deze inhoudsopgave staat alleen de html, waar iets interessants over is te melden):

[<!doctype html>](#)

[<html lang="nl">](#)

[<meta charset="utf-8">](#)

[<meta name="viewport" content="width=device-width, initial-scale=1">](#)

[<link rel="stylesheet" href="126-css-dl/afbeelding-126-dl.css">](#)

[Het <audio>-element](#)

[De download-links <p>Muziek downloaden Kies <a href="..."](#)

[download="..." title="..."> of ...](#)

[CSS](#) (in deze inhoudsopgave staat slechts de css die nodig is voor een werkend voorbeeld):

En dat is in dit voorbeeld niets.

Volledige code:

[HTML](#)

[CSS](#)

## Korte omschrijving

Geluid afspelen zonder plug-in of JavaScript met behulp van het <audio>-element.

## Opmerkingen

De gebruikte muziek is het nummer 'Amazing Grace' van Jason Shaw, afkomstig van [audionautix.com](#). Dit nummer wordt verspreid onder de Creative Commons-licentie [Attribution 3.0 Unported \(CC BY 3.0\)](#).

Het <audio>-element is niet goed op te maken met css, vandaar dat in dit voorbeeld heel weinig css wordt gebruikt. <audio> is alleen zichtbaar als in de html aan <audio> het attribuut 'controls' wordt toegevoegd. Als je dit weglaat, is het volledig onzichtbaar (en kan afspelen e.d. alleen worden aangestuurd met behulp van JavaScript.)

Eigenlijk valt <audio> in twee delen uiteen:

\* de knoppen en dergelijke waarmee de speler kan worden bediend. Je zou dit de 'inhoud' van <audio> kunnen noemen. Knoppen, schuifregelaar, en dergelijke kunnen niet met css worden opgemaakt.

(Dit klopt niet helemaal. Voor verschillende weergavemachines zijn er diverse css-eigenschappen, waarmee knoppen en dergelijke kunnen worden opgemaakt. Deze bestaan echter niet voor elke browser en verschillen ook nog 'ns per browser. Bovendien zijn deze geen onderdeel van welke standaard dan ook, dus ze kunnen altijd opeens worden veranderd.)

\* De 'buitenkant' van <audio>. Van zichzelf is <audio> een inline-element. In dit voorbeeld is het veranderd in een blok-element. Deze 'buitenkant' kan wel enigszins worden aangepast met behulp van css, maar daar heb je weinig aan.

Hieronder staan twee spelers. De bovenste is die van Firefox op Linux, de onderste die van Google Chrome op Linux.



De voor de afbeeldingen hierboven gebruikte css is:

```
audio {display: block; width: 400px; height: 100px;
outline: blue dotted 10px; border: red solid 10px;
border-bottom-left-radius: 30px; padding-right:
30px;}
```

(De achtergrondkleur en de kleur van outline zijn iets aangepast, omdat de kleur van de spelers in Firefox en Google Chrome verschilt. In andere browsers kan de speler er weer heel anders uitzien, ook in browsers die op de webkit-weergave-machine zijn gebaseerd.) Duidelijk is te zien dat deze css geen enkele invloed heeft op de speler zelf. De hoogte van 100 px bijvoorbeeld zorgt alleen maar voor lege ruimte boven de eigenlijke speler. De enige eigenschap die echt invloed heeft op de speler zelf, is de breedte. De breedte van de speler past zich aan aan de breedte van <audio>, waarbij die aanpassing vooral gebeurt door de balk van de schuifregelaar langer of korter te maken.

Als je het uiterlijk van de knoppen en dergelijke wilt aanpassen, moet je eigen knoppen en dergelijke maken, die met behulp van JavaScript worden bediend. Als je op internet zoekt op iets als 'custom audio element' vind je tal van handleidingen.

Links in deze uitleg, vooral links naar andere sites, kunnen verouderd zijn. Op [www.css-voorbeelden.nl/links.html](http://www.css-voorbeelden.nl/links.html) vind je steeds de meest recente links.

Dit voorbeeld is gemaakt op een systeem met Linux ([Kubuntu](http://kubuntu.org)). Daarbij is vooral gebruik gemaakt van [Visual Studio Code](http://visualstudio.codeplex.com), [GIMP](http://gimp.org) en [Firefox](http://firefox.com) met extensies. De pdf-bestanden zijn gemaakt met [LibreOffice](http://libreoffice.org).

Vragen of opmerkingen? Fout gevonden? Ga naar het [forum](http://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

Tot niet zo heel lang geleden kon een browser alleen geluid afspelen, als daarvoor een plug-in werd gebruikt, bijvoorbeeld Flash. Met behulp van het <audio>-element uit html5 kan de browser dit nu zelf.

De speler is alleen te zien, als in de html aan het <audio>-element het attribuut 'controls' wordt toegevoegd. Laat je dit weg, dan is <audio> volledig onzichtbaar en kan de speler alleen worden bediend met behulp van JavaScript.

Naast 'controls' bestaat nog een aantal andere attributen, waarmee de speler kan worden aangestuurd. Niet al deze attributen lijken even zinvol, tenzij je de speler aanstuurt met JavaScript. Als je bijvoorbeeld het attribuut 'muted' (gedempt) toevoegt aan <audio>, kun je

het geluid afspelen, zonder dat je iets hoort. Wat mij betreft zou dat in winkelcentra en bij een aantal politici een zegen zijn, maar op je site wil je waarschijnlijk liever geen onhoorbaar geluid.

Het attribuut 'autoplay' laat het geluid direct bij het laden van de pagina afspelen, zonder dat de bezoeker het afspelen hoeft te starten. Steeds meer browsers blokkeren dit. En terecht. Het zonder toestemming afspelen van geluid (en video met geluid) is gewoon ronduit onbeschoft en kan uiterst storend zijn.

Alle attributen worden verder beschreven bij [Het <audio>-element](#).

## De code aanpassen aan je eigen ontwerp

- Als je dit voorbeeld gaat aanpassen voor je eigen site, houd het dan in eerste instantie zo eenvoudig mogelijk. Ga vooral geen details invullen.
- Gebruik 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, als je al iets ziet.  
Publisher en Word zijn niet bedoeld om websites mee te maken. FrontPage is zwaar verouderd en wordt al jaren niet meer onderhouden door Microsoft.  
Ook OpenOffice en LibreOffice leveren een uiterst beroerd soort html af. Tekstverwerkers met al hun toeters en bellen zijn gewoon niet geschikt om websites mee te bouwen.  
Je kunt beter 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.
- 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: het is nooit hetzelfde als testen op een écht apparaat. Zo kun je bijvoorbeeld aanrakingen alleen echt testen op een echt touchscreen.  
Inmiddels hebben veel browsers in de ontwikkelgereedschappen mogelijkheden voor het simuleren van weergave op een kleiner scherm ingebouwd. Ook dit blijft een simulatie, maar geeft vaak wel een beter beeld dan zoomen.
- Ik maak zelf het liefst een site in Firefox. Als je 'n site maakt in Firefox, Opera, Safari, Google Chrome of Edge, is er 'n hele grote kans dat hij in alle browsers werkt. Ik geef de voorkeur aan Firefox, omdat het de enige grote browser is die niet bij een bedrijf hoort dat vooral op je centen of je data uit is.  
Google Chrome wordt ook door veel mensen gebruikt, maar ik heb dus wat moeite met hoe Google je hele surfgedrag, je schoenmaat en de kleur van je onderbroek vastlegt. Daarom gebruik ik Google Chrome zelf alleen om in 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](http://www.w3.org/qa/2002/04/valid-dtd-list).  
Gebruik het volledige doctype, inclusief de eventuele 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 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.

Het doctype voor html5 is uiterst simpel: `<!doctype html>`. Omdat het doctype voor html5 in alle browsers werkt, zelfs in de gelukkig vrijwel uitgestorven nachtmerrie Internet Explorer 6, is er geen enkele reden dit uiterst simpele doctype niet te gebruiken.

- Als tweede voer je de charset in. Dit vertelt de browser, welke tekenset er gebruikt moet worden, zodat letters met accenten e.d. overal goed worden weergegeven. 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 dit voorbeeld in is getest. Ook van Internet Explorer kun je meerdere versies naast elkaar draaien. Op de pagina met [links](#) staan onder het kopjes Gereedschap → Weergave e.d. testen 'n aantal links die daarbij kunnen helpen. De compatibiliteitsweergave in Internet Explorer is niet geschikt om in te testen, omdat deze enigszins verschilt van de weergave in échte browsers.

(Overigens wordt op deze site alleen nog in Internet Explorer 11 getest. Oudere versies van Internet Explorer worden niet meer ondersteund en zijn daardoor per definitie uiterst onveilig. In Edge, de opvolger van Internet Explorer, wordt wel gewoon getest.)

- 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. Ga daarna 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 verstoren. 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 `<br>1<br>2<br>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 eind (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, kan het helpen 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 eigenschappen verandert in de css, 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.
- `margin`, `padding` en `border` worden bij de hoogte en breedte van het element 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 met behulp van `box-sizing` de `padding` en `border` binnen de breedte en hoogte van de inhoud te zetten, als je dat handiger vindt.  
Met nieuwere css-eigenschappen als `grid` en `flexbox`, die speciaal zijn gemaakt om een lay-out mee te maken, spelen dit soort problemen veel minder. Maar hier moet je weer goed op de ondersteuning door browsers letten, want niet elke browser ondersteunt al alles van deze eigenschappen, en zeker niet foutloos.
- In plaats van een absolute eenheid als `px` kun je ook een relatieve eenheid gebruiken, met name `em` en `rem`. Voordeel van `em` en `rem` is dat een lettergrootte, regelhoogte, e.d. in `em` en `rem` in alle browsers 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` en `rem` vrijwel altijd beter dan absolute eenheden als `px`.  
(De minder bekende `rem` is ongeveer hetzelfde als de `em`. Alleen is de lettergrootte bij `rem` gebaseerd op de lettergrootte van het `<html>`-element, waardoor de `rem` overal op de pagina precies even groot is. Bij de `em` kan de lettergrootte worden beïnvloed door de voorouders van het element, bij de `rem` niet.)  
Zoomen kan trouwens altijd, ongeacht welke eenheid je gebruikt.
- Valideren, valideren, valideren en dan voor 't slapen gaan nog 'ns valideren.  
Valiwié???
- 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.  
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. Zeker met nieuwere html en css wil de validator nog wel eens achterlopen, terwijl dat al prima is te gebruiken.  
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. Je kunt je css en html valideren als 't online staat, maar ook als het nog in je computer staat. Html kun je valideren op: [validator.w3.org/nu](http://validator.w3.org/nu).  
Css kun je valideren op: [jigsaw.w3.org/css-validator](http://jigsaw.w3.org/css-validator).



## Toegankelijkheid en zoekmachines

De tekst in dit hoofdstukje is een algemene tekst, die voor elke pagina geldt. Eventueel specifiek voor dit voorbeeld geldende problemen en eventuele aanpassingen om die problemen te voorkomen staan bij [Bekende problemen \(en oplossingen\)](#).

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 in een link niet duidelijk wordt, 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.

Hierdoor zijn accesskeys (vrijwel) niet te gebruiken. Misschien kunnen ze nog enig nut hebben op sites, die gericht zijn op 'n specifieke groep gebruikers. Maar voor algemene sites is het advies: normaal genomen 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 het attribuut `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.

- Als, zoals hierboven beschreven, een gebruiker van de Tab-toets bij een link, invoerveld, e.d. is aangekomen, heeft dit element 'focus'. Dit wordt aangegeven door de link, invoerveld, e.d. extra te markeren met een kadertje. Dat kadertje mag je alleen weghalen, als op een andere manier wordt duidelijk gemaakt, welk element focus heeft. Een gebruiker van de Tab-toets kan anders niet zien, waar zij of hij zit, en welk element gaat reageren op bijvoorbeeld een Enter.
- 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 zogenaamde skip-link is vaak nog wel zinvol. Dat is 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. (Alleen in de schermlezer TalkBack op Android werkt zo'n buiten het scherm geplaatste link niet. TalkBack leest zo'n link wel voor, maar de link kan niet worden gevolgd, als deze buiten het scherm staat.) Een skip-link staat bovenaan de pagina, nog boven menu, header, e.d., en linkt naar de eigenlijke inhoud van de pagina. Hierdoor kunnen mensen met één toetsaanslag naar de eigenlijke inhoud van de pagina gaan.  
Een skip-link is vooral 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. Op pagina's en in voorbeelden waar dat nuttig is, wordt op deze site een skip-link gebruikt. (Althans: nog niet in alle voorbeelden die daarvoor in aanmerking komen, zit een skip-link. Maar geleidelijk aan worden dat er steeds meer.)
- 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.
- 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 volstreekte ontoegankelijkheid voor schermlezers e.d. en als extra bonus vaak 'n lagere plaats in een zoekmachine.



- Frames horen bij een volstrekt verouderde techniek, die heel veel nadelen met zich meebrengt. `<iframe>`'s hebben voor een deel dezelfde nadelen. Eén van die nadelen is dat de verschillende frames voor zoekmachines, schermlezers, e.d. als los zand aan elkaar hangen, omdat ze los van elkaar worden weergegeven. Ze staan wel naast elkaar op het scherm, maar er zit intern geen verband tussen.  
Als je 'n stuk code vaker wilt gebruiken, zoals 'n menu dat op elke pagina hetzelfde is, voeg dat dan in met PHP. Dan wordt de pagina niet pas in de browser, maar al op de server samengesteld. Hierdoor zien zoekmachines, schermlezers, e.d. één pagina, net zoals wanneer je maar één pagina met html zou hebben geschreven.  
(Je kunt ook invoegen met behulp van SSI (Server Side Includes). Maar tegenwoordig kun je beter PHP dan SSI gebruiken, omdat SSI min of meer aan het uitsterven is en PHP veel meer mogelijkheden heeft. Op deze site wordt in enkele voorbeelden nog SSI gebruikt, maar zodra die worden bijgewerkt, gaat dat vervangen worden door PHP.)
- Geef de taal van het document aan, en bij woorden en dergelijke die afwijken van die taal de afwijkende taal met behulp van `lang=" . . . "`. Op deze site gebeurt dat maar af en toe, omdat de tekst (en vooral de code) een mengsel is van Engels, Nederlands en eigengemaakte namen. Dat 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.
- Geef een verandering niet alleen door kleur aan. Een grote groep mensen heeft moeite met het onderscheiden van kleuren en/of het herkennen van kleuren. Verander bijvoorbeeld een ronde rode knop niet in een ronde groene knop, maar in een vierkante groene knop. Door ook de vorm te veranderen, is het herkennen van de verandering niet alleen van een kleur afhankelijk.
- Zorg voor voldoende contrast tussen achtergrond- en voorgrondkleur, tussen `background-color` en `color`. Soms zie je heel lichtgrijze tekst op een donkergrijze achtergrond, en dan ook nog in een mini-formaat. Dat is dus voor heel veel mensen stomweg volledig onleesbaar. Een uitstekende online contrastchecker is bijvoorbeeld te vinden op [snook.ca](http://snook.ca).
- 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. Hetzelfde geldt voor die fantastisch mooie flash-pagina's, als daarbij geen voorzieningen voor dit soort programma's zijn aangebracht. (Omdat flash wordt uitgefaseerd en op steeds minder machines is te bekijken, zijn flash-pagina's hoe dan ook geen goed idee meer.)  
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](http://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](http://www.webbie.org.uk). (LET OP: kies voor 'Install WebbIE 4 Web Browser Now'. Dat is – op het moment van schrijven – de een na bovenste knop. Als je voor de bovenste download kiest, krijg je 'n hele berg hulpprogramma's erbij, waar je voor het testen niets aan hebt.)
- Ten slotte kun je je pagina nog online op toegankelijkheid laten controleren op 'n behoorlijk aantal sites, zoals:

[lowvision.support](#): laat zien hoe een kleurenblinde de site ziet. Engelstalig.  
[wave.webaim.org](#): deze laat grafisch zien, hoe de toegankelijkheid is. Engelstalig. Deze tester is ook als extensie in Firefox en Google Chrome te installeren.  
Op de pagina met [links](#) kun je onder Toegankelijkheid links naar meer tests e.d. vinden.

## Getest in

Laatst gecontroleerd op 23 april 2019.

Onder dit kopje staat alleen maar, hoe en waarin is getest. Alle eventuele problemen, ook die met betrekking tot zoomen, lettergroottes, toegankelijkheid, uitstaan van JavaScript en/of css, enz. staan iets 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!

Dit voorbeeld is getest op de volgende systemen:

### DESKTOPCOMPUTERS

Windows 7 (1280 x 1024 px, resolution: 96 dpi):

Firefox, UC Browser, Google Chrome, Opera en Internet Explorer 11, in grotere en kleinere browservensters.

OS X 10.11.6 ('El Capitan') (1680 x 1050 px, resolution: 96: dpi, device-pixel-ratio: 1):

Firefox, Safari, Opera en Google Chrome, in grotere en kleinere browservensters.

Linux (Kubuntu 18.04 LTS, 'Bionic Beaver Tahr') (2560 x 1080 px, resolution: 96 dpi):

Firefox, Opera en Google Chrome, in grotere en kleinere browservensters.

### LAPTOPS

Windows 8.1 (1366 x 768 px, resolution: 96 dpi):

Bureaublad-versie: Firefox, UC Browser, Google Chrome, Opera en Internet Explorer 11, in grotere en kleinere browservensters.

Startscherm-versie: Internet Explorer 11.

Windows 10 (1600 x 900 px, resolution: 96 dpi):

Firefox, UC Browser, Google Chrome, Opera, Internet Explorer 11 en Edge, in grotere en kleinere browservensters.

### TABLETS

iPad met iOS 9.3.5 (1024 x 768 px, device-pixel-ratio: 1):

Safari, Chrome for iOS, UC Browser, Firefox (alle portret en landschap).

Opera Mini (Opera Turbo) portret en landschap.

iPad met iOS 12.2 (2048 x 1536 px, device-pixel-ratio: 2 ('retina')):

Safari, Chrome for iOS, Firefox, Microsoft Edge (alle portret en landschap).

Opera Mini (Opera Turbo) portret en landschap.

Android 4.4.2 ('Kitkat') (1280 x 800 px, resolution: 96 dpi):

UC Browser, Firefox en Chrome (alle portret en landschap).

Android 4.4.2 ('Kitkat') (2560 x 1600 px, resolution: 192 dpi):  
Android browser, UC Browser, Firefox en Chrome (alle portret en landschap).

Android 6.0 ('Marshmallow') (1920 x 1200 px, resolution: 144 dpi):  
Dolphin, Samsung Internet, UC Browser, Firefox en Chrome (alle portret en landschap).  
Opera Mini (besparingen uitgeschakeld) portret en landschap.

Android 8.1.0 ('Oreo') (1920 x 1200 px, resolution: 144 dpi):  
Dolphin, Samsung Internet, UC Browser, Firefox, Microsoft Edge en Chrome (alle portret en landschap).  
Opera Mini (besparingen uitgeschakeld) portret en landschap.

#### SMARTPHONES

Windows 10 Mobile (1280 x 720 px, resolution: 192 dpi):  
Edge en UC browser (portret en landschap).

Android 4.1.2 ('Jelly Bean') (800 x 480 px, resolution: 144 dpi):  
Chrome, UC Browser en Firefox (alle portret en landschap).  
Opera Mini (besparingen uitgeschakeld) portret en landschap.

Android 8.1.0 ('Oreo') (1280 x 720 px, resolution: 192 dpi):  
Dolphin, Samsung Internet, UC Browser, Firefox, Microsoft Edge en Chrome (alle portret en landschap).  
Opera Mini (besparingen uitgeschakeld) portret en landschap.

Er is op de aan het begin van dit hoofdstukje genoemde controledatum getest in de meest recente versie van de browser, die op het betreffende besturingssysteem kon draaien. Het aantal geteste browsers en systemen is al tamelijk fors, en als ook nog rekening gehouden moet worden met (zwaar) verouderde browsers, is het gewoon niet meer te doen. Surfen met een verouderde browser is trouwens vragen om ellende, want updates van browsers hebben heel vaak met beveiligingsproblemen te maken.

In- en uitzoomen en – voor zover de browser dat kan – een kleinere en grotere letter zijn ook getest. Er is ingezoomd en vergroot tot zover de browser kan, maar niet verder dan 200%.

Er is getest met behulp van muis en toetsenbord, behalve op iOS, Android en Windows 10 Mobile, waar een touchscreen is gebruikt. Op Windows 8.1 en 10 is getest met touchscreen, touchpad, toetsenbord, muis, en – waar dat zinvol was – op een combinatie daarvan.

Als JavaScript is gebruikt, is op de desktop ook getest zonder JavaScript. (Op iOS, Android en Windows 10 Mobile 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.) Ook is getest zonder css en – als afbeeldingen worden gebruikt – zonder afbeeldingen.

#### SCHERMLEZERS E.D.

Naast deze 'gewone' browsers is ook getest in Lynx, WebbIE, NVDA, TalkBack, VoiceOver, ChromeVox en Verteller.

[Lynx](#) is een browser die alleen tekst laat zien en geen css gebruikt. Er is getest op Linux.

[WebbIE](#). is een browser die gericht is op mensen met een handicap. Er is getest op Windows7. (LET OP: kies voor 'Install WebbIE 4 Web Browser Now'. Dat is – op het moment van schrijven – de een na bovenste knop. Als je voor de bovenste download kiest, krijg je 'n hele berg hulpprogramma's erbij, waar je voor het testen niets aan hebt.)

[NVDA](#) is een schermlezer, zoals die door blinden wordt gebruikt. Er is getest op Windows 7 en Windows 10 in combinatie met Firefox.

TalkBack is een in Android ingebouwde schermlezer. Er is getest in combinatie met Chrome op Android 4.4.2, 6.0, 7.0 en 8.1.0.

VoiceOver is een in iOS en OS X ingebouwde schermlezer. Er is getest in combinatie met Safari op iOS (9.3.5 en 12.1.) en OS X 10.11.6.

ChromeVox is een schermlezer in de vorm van een extensie bij Google Chrome. Er is getest op een systeem met Kubuntu Linux 14.04.

Verteller (Narrator) is een in Windows 10 ingebouwde schermlezer. Er is getest in combinatie met Edge.

(Voor de bovenstaande programma's zijn links naar sites met uitleg e.d. te vinden op de pagina met [links](#) onder Toegankelijkheid → Blinden en slechtzienden.)

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 problemen in schermlezers (en eventuele aanpassingen om die te voorkomen) staan iets hieronder bij [Bekende problemen \(en oplossingen\)](#).

Alleen op de hierboven genoemde systemen en browsers is getest. Er is dus niet getest op bijvoorbeeld 'n Blackberry. Er is een kans dat dit voorbeeld niet (volledig) werkt op niet-geteste systemen en apparaten. Om het wel (volledig) werkend te krijgen, zul je soms (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 Samsung Internet op Android 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 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 worden pas getest, als ze uit het bèta-stadium zijn. Anders is er 'n redelijke kans dat je tegen 'n bug zit te vechten, 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 kunnen niet worden beantwoord, en het melden van fouten in niet-geteste browsers heeft ook geen enkel nut. (Melden van fouten, problemen, enz. in wel geteste browsers: graag! Dat kan op het [forum](#).)

## 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.

Bij toegankelijkheid is er vaak geen goed onderscheid te maken tussen oplossing en probleem. Zonder (heel simpele) aanpassingen heb je vaak 'n probleem, en omgekeerd. Daarom staan wat betreft toegankelijkheid aanpassingen en problemen hier bij elkaar in dit hoofdstukje.

Voor zover van toepassing wordt eerst het ontbreken van JavaScript, css en/of afbeeldingen besproken. Vervolgens problemen en aanpassingen met betrekking tot toegankelijkheid voor specifieke groepen bezoekers, zoals zoomen en andere lettergrootte, Tab-toets, tekstbrowsers en schermlezers. Als laatste volgen algemene problemen in alle of in specifieke browsers.

Als in een onderdeel één of meer problemen worden besproken, staat in een breed rood kadertje een korte samenvatting daarvan. Bij een onderwerp over toegankelijkheid zijn er soms geen problemen, maar alleen aanpassingen. In dat geval staat in een smal groen kadertje 'Geen problemen'.

### ZONDER JAVASCRIPT

Probleem: zonder JavaScript werkt de speler niet in Safari.

Niet in iOS en niet op de desktop. In de andere geteste browsers op iOS kan JavaScript niet worden uitgezet, dus dit speelt alleen in Safari.

Mensen die JavaScript hebben uitstaan, zullen dit vaker tegenkomen en al weten, dus een echt probleem is het niet.

### ZONDER CSS

Geen problemen.

De speler werkt gewoon. Alleen staat alles niet meer netjes gecentreerd en zo.

### GEBRUIKERS TAB-TOETS

Geen problemen.

Er is geen enkele invloed op de volgorde van de Tab-toetsen of zoiets.

### TEKSTBROWSERS

Probleem: geluid wordt niet afgespeeld.

Eh, tja, dat is nou juist de bedoeling van een tekstbrowser: dat je alleen tekst ziet, en geen afbeeldingen, video's, of wat dan ook.

Lynx geeft netjes de tekst weer, die bedoeld is voor als de browser het niet kan afspelen, WebIE niet. Beide browsers geven de mogelijkheid de geluidsbestanden te downloaden.

### SCHERMLEZERS

Geen problemen, er is geen enkele invloed op schermlezers.

In alle geteste schermlezers kan de speler met het toetsenbord worden bediend.

### ZOOMEN EN LETTERGROOTTES

Geen problemen.

## ANDROID BROWSER EN DOLPHIN OP ANDROID, UC BROWSER OP SOMMIGE VERSIES VAN ANDROID, OPERA MINI OP iOS 12.2

Probleem: de speler werkt niet.

In deze browsers kan het geluid niet worden afgespeeld, omdat de speler dat stomweg vertikt.

Dolphin kan de bestanden in geen enkele van de geteste versies van Android afspelen. Wel geeft de speler de mogelijkheid de bestanden te downloaden, waarna je ze alsnog kunt afspelen.

In Android browser kan het af te spelen bestand ook niet worden gedownload. Omdat deze browser na Android 4.3 niet meer wordt geleverd, is dit probleem snel aan het uitsterven.

UC browser werkt in alle geteste versies van Android, behalve in de tablet met Android 4.4.2 met gewone resolutie (1280 x 800 px).

Opera Mini op iOS 12.2 doet gewoon helemaal niets. Op iOS 9.3.5 werkt het gewoon, net als op Android, dus het gaat hier waarschijnlijk om een of andere bug, die hopelijk snel wordt gerepareerd.

### Wijzigingen

Alleen grotere wijzigingen worden hier vermeld, geen dingen als een link die is geüpdatet.  
23 april 2019:

Nieuw opgenomen.

### Inhoud van de download en licenties

De inhoud van deze download kan vrij worden gebruikt, met drie beperkingen:

- \* Sommige onderdelen die van 'n andere site of zo afkomstig zijn, vallen mogelijk onder een of andere licentie. Dat is hieronder bij het betreffende onderdeel te vinden.

- \* Je gebruikt het materiaal uit deze download volledig op eigen risico. Het kan prima zijn dat er fouten in de hier verstrekte code e.d. zitten. Voor eventuele schade die door gebruik van materiaal uit deze download ontstaat, in welke vorm dan ook, zijn [www.css-voorbeelden.nl](http://www.css-voorbeelden.nl) en medewerkers daarvan op geen enkele manier verantwoordelijk.

- \* Dit voorbeeld (en de bijbehorende uitleg e.d.) wordt regelmatig bijgewerkt. Het is daarom niet toegestaan dit voorbeeld (en de bijbehorende uitleg e.d.) op welke manier dan ook te verspreiden, zonder daarbij duidelijk te vermelden dat voorbeeld, uitleg, e.d. afkomstig zijn van [www.css-voorbeelden.nl](http://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](http://www.css-voorbeelden.nl) wordt trouwens altijd op prijs gesteld.

afbeelding-126-dl.html: de pagina met het voorbeeld.

afbeelding-126.pdf: deze uitleg (aangepast aan de inhoud van de download).

afbeelding-126-inhoud-download-en-licenties.txt: een kopie van de tekst onder dit kopje  
(Inhoud van de download en licenties).

126-css-dl:

afbeelding-126-dl.css: stylesheet voor afbeelding-126-dl.html.

126-files-dl:

'Amazing Grace' van Jason Shaw in ogg- en mp3-formaat.

De muziek is afkomstig van afkomstig van [audionautix.com](http://audionautix.com) en wordt verspreid onder de Creative Commons-licentie Attribution 3.0 Unported (CC BY 3.0), die is te vinden op

[creativecommons.org/licenses/by/3.0/deed.en\\_US](http://creativecommons.org/licenses/by/3.0/deed.en_US).



## HTML

De code is geschreven in een afwijkende lettersoort. De code die te maken heeft met de basis van dit voorbeeld (essentiële code), is in de hele uitleg blauw gekleurd. Alle niet-essentiële code is zwart. (In de inhoudsopgave staat alles in een gewone letter vanwege de leesbaarheid.)

In de html hieronder wordt alleen de html besproken, waarover iets meer is te vertellen. Een <h1> bijvoorbeeld wordt in de regel niet genoemd, omdat daarover weinig interessants valt te melden. (Als bijvoorbeeld het uiterlijk van de <h1> wordt aangepast met behulp van css, staat dat verderop bij de bespreking van de [css](#).)

Zaken als een doctype en charset hebben soms wat voor veel mensen onbekende effecten, dus daarover wordt hieronder wel een en ander geschreven.

<!doctype html>

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](http://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. Dit kan zonder enig probleem worden gebruikt: het werkt zelfs in Internet Explorer 6.

<html lang="nl">

De toevoeging lang="nl" bij <html> geeft aan dat de pagina in het Nederlands is. De taal is van belang voor schermlezers, automatisch afbreken, automatisch genereren van aanhalingstekens, juist gebruik van decimale punt of komma, e.d.

<meta charset="utf-8">

Zorgt dat de browser letters met accenten e.d. goed kan weergeven.

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 (&auml; 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.

In html5 hoeft deze regel niet langer te zijn, dan wat hier staat.

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

Mobiele apparaten variëren enorm in grootte. En dat is een probleem. Sites waren, in ieder geval tot enkele jaren geleden, 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 alle 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 vensters van mobiele apparaten. In dit voorbeeld bijvoorbeeld wordt de speler nooit breder dan het venster.

Maar die stomme mobiele browser weet dat niet, dus die gaat ervan uit dat ook deze pagina 980 px breed is, en verkleint die dan. Dat is ongeveer even behulpzaam als de gediensig 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 portretstand 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 portretstand dus 768 px.

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. Er is ook een instructie om zoomen helemaal onmogelijk te maken, maar die wordt niet gebruikt. De bezoeker kan zelf nog gewoon zoomen, wat belangrijk is voor mensen die wat slechter zien.

`<link rel="stylesheet" href="126-css-dl/afbeelding-126-dl.css">`

Dit is een koppeling naar een externe 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, waar je eigen stylesheet staat.

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 hoeft 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 externe 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 stylesheet eigenlijk 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 hoeft 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 een extern stylesheet is, terwijl hierboven omstandig wordt beweerd dat dat in dit voorbeeld eigenlijk geen nut heeft: overzichtelijkheid. Nu kun je html en css los van elkaar bekijken.)

## Het <audio>-element

```
<audio controls preload="metadata">
  <source src="126-files-dl/AmazingGrace.ogg" type="audio/ogg">
  <source src="126-files-dl/AmazingGrace.mp3" type="audio/mpeg">
  Je browser ondersteunt het afspelen van geluid niet. Je kunt
    hieronder nog wel de muziek downloaden.
</audio>
```

Het <audio>-element is onderdeel van html5. In dit voorbeeld worden slechts twee attributen gebruikt.

Deze site beperkt zich grotendeels tot html en css, maar <audio> kan ook met JavaScript worden aangestuurd. Als je op internet zoekt naar iets als 'custom audio element', vind je tal van handleidingen.

Hieronder staat een korte omschrijving van de onderdelen, verderop staat de uitgebreide beschrijving van alle onderdelen.

<audio> en </audio>: begin- en eindtag. In de begintag staan enkele attributen die aangeven, hoe het element zich moet gedragen. Er zijn meer attributen, die worden verderop besproken.

<source>: binnen elke <source> staat een geluidsbestand. Zodra de browser een bestand vindt dat kan worden afgespeeld, wordt niet verder gekeken naar volgende <source>'s. Bij dit vinden helpt het opgeven MIME-formaat (type) van het geluidsbestand.

Tekst onderaan: als een browser <audio> niet ondersteunt, wordt deze tekst getoond.

```
<audio controls preload="metadata">
```

<audio>: gewoon de openingstag. Helemaal achteraan, na een aantal elementen dat binnen <audio> zit, wordt het element afgesloten met </audio>.

controls: geeft aan dat knoppen moeten worden weergegeven, waarmee de speler is te bedienen. De maker van de browser heeft voor deze knoppen gezorgd, dus ze verschillen behoorlijk per browser.

Als je controls weglaat, worden de bedieningsknoppen niet getoond. Anders dan bij <video> zie je nu helemaal niets meer en kan de speler niet worden bediend, ook niet als je het scherm aanraakt of rechtsklikt om een contextueel menu te openen.

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.

preload="metadata": dit regelt of het geluidsbestand al wordt gedownload, voordat het wordt afgespeeld, of dat pas wordt gedownload, als wordt afgespeeld. Dit is niet meer dan een hint voor de browser, de browser kan afwijken van wat hier wordt opgegeven.

Als preload niet wordt gebruikt, downloaden sommige browsers het volledige geluidsbestand direct, als de pagina wordt geopend.

Mogelijke waarden:

`none`: in eerste instantie niets downloaden, maar wachten tot wordt afgespeeld.

`metadata`: alleen metagegevens zoals tijdsduur, artiest, en dergelijke ophalen. Dit wordt in het voorbeeld gebruikt. Pas als het geluid daadwerkelijk wordt afgespeeld, wordt het volledige bestand gedownload.

`auto`: bestand gelijk downloaden, al voordat wordt gekozen voor afspelen. Als je niets invult bij `preload`, is dit de standaardwaarde.

`autoplay`: wordt niet gebruikt in dit voorbeeld. Zorgt ervoor dat het geluid 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 het om een opname gaat van het voltallige Nederlandse elftal, dat luidkeels het Wilhelmus zingt. Na de revolutie komt hier drie maanden zonnepanelen aanbrengen op een verlaten olieplatform op te staan.

Serius: dit is een prima manier om bezoekers weg te jagen. Laat mensen zelf kiezen of ze wel of niet je geluid willen afspelen. Het wordt nog extra sadistisch als je `controls` weglaat, want dan ontbreekt in de meeste browsers de mogelijkheid om het geluid uit te zetten.

`crossorigin`: wordt niet gebruikt in dit voorbeeld.

Een pagina kan normaal genomen zonder problemen van een andere plek op internet opvragen. Het opgevraagde bestand wordt alleen weergegeven, verder kan er weinig mee worden gedaan.

Als je echter met JavaScript een bestand van elders opvraagt, ligt dat anders. Daarom kan een script normaal genomen uit veiligheidsoverwegingen geen bestanden van buiten de eigen site opvragen. Zou dat niet zo zijn, dan zou je hele leuke dingen met de bankrekening van Bill Gates kunnen doen. (Die Gates zelf mogelijk iets minder leuk vindt.)

Met dit attribuut kun je aangeven dat een script een bestand toch van elders moet ophalen, en op wat voor manier dat moet gebeuren. De server waar het vandaan komt, moet dat uiteraard nog wel goed vinden. Voor het eigenlijke ophalen heb je JavaScript nodig.

Als je op internet zoekt naar 'crossorigin cors' vind je tal van tutorials.

`loop`: wordt niet gebruikt in dit voorbeeld. Als het geluidsbestand is uitgespeeld, wordt weer van voren af aan begonnen.

`muted`: wordt niet gebruikt in dit voorbeeld. Zet het geluid uit. Je kunt nu dus het geluidsbestand afspelen, zonder dat je dat hoort.

Eerlijk gezegd is schrijver dezes niet helemaal duidelijk, wat hiervan het nut zou zijn. Ik bedoel: als je 'n geluidsbestand afspeelt, wil je het toch horen? Bij `<video>` heeft het 'n duidelijke functie: 'n filmpje afspelen zonder geluid.

Maar mogelijk mis ik iets en is het bijvoorbeeld wel nuttig, als `<audio>` met behulp van JavaScript wordt aangestuurd of zoiets.

`src`: hierin komt de url van het af te spelen geluidsbestand. Dit wordt alleen gebruikt, als er

maar één bestand is opgegeven. Zodra er – zoals hier in dit voorbeeld – meerdere geluidsbestanden zijn, waaruit de browser moet kiezen, komt elk bestand in een aparte <source> te staan binnen <audio>.

Als het wel wordt gebruikt ziet dit er zo uit:

```
<video src="buurman-doucht.ogg">
```

```
<source src="126-files-dl/AmazingGrace.ogg" type="audio/ogg">
```

Binnen <audio> kunnen meerdere <source>'s staan. Elke <source> verwijst naar een bepaald geluidsbestand. De browser bekijkt van boven naar beneden alle <source>'s, tot een bestand wordt gevonden dat kan worden afgespeeld. Zodra dat het geval is, worden lagere <source>'s niet meer bekeken.

<source: gewoon de opening van de tag. De tag wordt aan het eind van de regel afgesloten met een >.

src="(..)": de url van het af te spelen geluidsbestand. Hier is dat 126-files-dl/AmazingGrace.ogg.

type="(..)": om het MIME-type van de video op te geven. Hier is dat audio/ogg: een geluidsbestand in een container van de soort ogg.

Als geen MIME-type wordt opgegeven, moet de browser van elk geluidsbestand een klein stukje downloaden om te kijken, of het bestand kan worden afgespeeld, net zolang tot een bestand is gevonden dat kan worden afgespeeld.

codecs="(..)": wordt niet gebruikt in dit voorbeeld. Hier kun je voor de meer exotische formaten ook nog de exacte codec opgeven die is gebruikt. Normaal genomen is dit niet nodig.

```
<source src="126-files-dl/AmazingGrace.mp3" type="audio/mpeg">
```

Deze <source> werkt hetzelfde als [<source src="126-files-dl/AmazingGrace.ogg" type="audio/ogg">](#) hierboven. De enige verschillen zijn de naam van het geluidsbestand en het MIME-type. Bij de eerste <source> is het MIME-type audio/ogg, hier is het audio/mpeg (en niet audio/mp3, zoals vaak wordt gedacht). Browsers die ogg kunnen afspelen, hebben het eerdere geluidsbestand gekozen en komen niet eens aan bij deze regel.

In dit voorbeeld is ogg boven mp3 gezet, omdat het bestand veel kleiner is: 2,5 MB tegen 6,1 MB, bij gelijke kwaliteit. Omdat de meeste browsers met ogg uit de voeten kunnen, zullen ze dit kleinere bestand gebruiken.

Omdat op mp3 inmiddels geen patent meer rust, kunnen vrijwel alle browsers dit inmiddels ook afspelen. Maar gezien het verschil in bestandsgrootte blijft ogg belangrijk.

Je browser ondersteunt het afspelen van geluid niet. Je kunt hieronder nog wel de muziek downloaden.

Browsers van voor het Stenen tijdperk ondersteunen <audio> niet. In dat geval wordt de hier weergegeven tekst op het scherm gezet. Deze browsers zullen nog nauwelijks worden gebruikt, maar het is een kleine moeite zo'n tekst toe te voegen.

Bij [Bekende problemen \(en oplossingen\)](#) worden een paar browsers genoemd, die geen geluid kunnen afspelen (sommige geven wel de mogelijkheid het bestand te downloaden).

Bij geen van deze browsers wordt deze tekst getoond.

**De download-links** `<p>Muziek downloaden? Kies <a href="..." download="..." title="...">`  
of ...

Onder de speler staan links om het geluidsbestand te kunnen downloaden. Deze links zijn aangebracht, voor het geval de browser het geluid niet af kan spelen, zoals Android browser. Je kunt dan in ieder geval het geluidsbestand downloaden en op 'n andere manier afspelen. De volledige code voor het downloaden van de geluidsbestanden:

```
<p>Muziek downloaden? Kies <a  
    href="126-files-dl/AmazingGrace.ogg"  
    download="Amazing Grace van Jason Shaw.ogg"  
    title="Download muziek in ogg-formaat">ogg</a> (2,5  
    <abbr title="megabyte">MB</abbr>) of <a href="126-  
    files-dl/AmazingGrace.mp3" download="Amazing Grace  
    van Jason Shaw.mp3" title="Download muziek in mp3-  
    formaat">mp3</a> (6,1 <abbr>MB</abbr>).</p>
```

Hier staan twee links: eentje voor het geluid in ogg-formaat, en eentje voor het geluid in mp3-formaat. De bezoeker kan hierdoor kiezen, welk formaat wordt gedownload.

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

```
download="Amazing Grace van Jason Shaw.ogg"
```

Normaal genomen zou als naam voor de download 'AmazingGrace.ogg' worden gebruikt. Maar in browsers die download herkennen, wordt de naam nu veranderd in het veel mensvriendelijker 'Amazing Grace van Jason Shaw.ogg'.

De extensie 'ogg' 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. Het is niet nodig dit veiligheidsrisico naar andere systemen te exporteren, dus daarom wordt achter download een extensie gebruikt. (Voor de extensie 'mp3' geldt uiteraard hetzelfde.)

Achter de download-link staat de grootte van het bestand in MB. Omdat MB een afkorting is, staat deze in een `<abbr>`. De afkorting MB komt voor de eerste keer achter de eerste download-link, 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 geluidsbestand van 300 gigabyte begint te downloaden. Daarom is achter de link de grootte van de download aangegeven.

Als de browser het attribuut download ondersteunt, wordt bij klikken op de link het bestand in principe gelijk gedownload. Als de browser het niet ondersteunt, wordt het geluidsbestand afgespeeld en moet je rechtsklikken of iets langer aanraken om te kunnen downloaden.



Als je wilt uitproberen, hoe deze download-link in de diverse browsers op de diverse systemen werkt, kun je dat het best op een server uitproberen. Firefox bijvoorbeeld speelde tot voor kort het geluid gewoon af, als je op je lokale computer op de link klikte. Maar als je het probeerde op 'n server, werd het geluidsbestand in datzelfde Firefox wel gedownload. (Inmiddels doet Firefox dat ook op de lokale computer.)

## CSS

De code is geschreven in een afwijkende lettersoort. De code die te maken heeft met de basis van dit voorbeeld (essentiële code) is in de hele uitleg **blauw** gekleurd. Alle niet-essentiële code is zwart. (In de inhoudsopgave staat alles in een gewone letter vanwege de leesbaarheid.) Omdat deze site nou eenmaal (voornamelijk) op css is gericht, wordt hieronder álle css besproken.

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 p
{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 p {
    margin-left: 16px;
    height: 120px;
    text-align: center;
}
```

Hiernaast is het heel belangrijk voldoende commentaar (uitleg) in de stylesheet te schrijven. Op dit moment 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 één 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. De uitgebreide versie 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. Op de pagina met [links](#) kun je onder het kopje Gereedschap → Snelheid, testen, gzip, comprimeren (inclusief theorie) links naar sites vinden, waar je bestanden kunt comprimeren. (Stylesheets op deze site zijn niet gecomprimeerd. Omdat het vaak juist om de css gaat, kunnen mensen dan zonder al te veel moeite de css bekijken.)

```
/* afbeelding-126-dl.css */
```

Om vergissingen te voorkomen is het een goede gewoonte bovenaan het stijlbestand even de naam neer te zetten. Voor je het weet, zit je anders in het verkeerde bestand te werken.

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;
```

Achtergrondkleurtje.

```
color: black;
```

Voorgrondkleur zwart. Dit is o.a. de kleur van de tekst.

Hoewel dit de standaardkleur is, wordt deze toch specifiek opgegeven. Hierboven is een achtergrondkleur opgegeven. Sommige mensen hebben zelf de voorgrond- en/of achtergrondkleur veranderd, bijvoorbeeld omdat ze slecht kleuren kunnen onderscheiden. Als nu de achtergrondkleur wordt veranderd, maar niet de voorgrondkleur, loop je het risico dat tekstkleur en achtergrondkleur te veel op elkaar gaan lijken.

Door beide op te geven, is redelijk zeker dat achtergrond- en tekstkleur genoeg van elkaar blijven verschillen. Als de gebruiker `!important` heeft gebruikt in een eigen stylesheet, is er nog niets aan de hand, want dan veranderen achtergrond- en voorgrondkleur geen van beide.

```
font-family: Arial, Helvetica, sans-serif;
```

Als Arial is geïnstalleerd op de machine van de bezoeker, wordt deze gebruikt, anders Helvetica. Als die ook niet wordt gevonden, wordt in ieder geval een schreefloze letter (zonder dwarsstreepjes) gebruikt.

```
font-size: 110%;
```

Iets groter dan standaard. 't Zal de leeftijd zijn, maar ik vind de standaardgrootte wat te klein.

Als eenheid wordt de relatieve eenheid % gebruikt, omdat bij gebruik van een absolute eenheid zoals px niet alle browsers de lettergrootte kunnen veranderen.

Zoomen kan wel altijd, ongeacht welke eenheid voor de lettergrootte wordt gebruikt.

```
margin: 0; padding: 0;
```

Slim om te doen vanwege verschillen tussen browsers.

(Mogelijk is padding niet meer nodig, maar in het verleden verschilde de standaard-padding tussen browsers. Het is simpeler om dit gewoon te blijven gebruiken dan om een uitgebreide test uit te gaan voeren om te kijken, of dit nog wel nodig is.)

main

Alle `<main>`'s. Dat is er maar één: de belangrijkste inhoud van de pagina staat erin. (Hier is dat alleen de speler met de eronder staande links.)

`display: block;`

Oudere browsers kennen `<main>` niet. Een onbekend element is standaard een inline-element, waardoor eigenschappen als breedte niet kunnen worden gebruikt. Nu weten alle browsers dat dit een blok-element is.

`width: 640px;`

Breedte.

`max-width: 94%;`

Hier gelijk boven is een breedte van 640 px opgegeven. Dat levert in browservensters smaller dan 640 px een horizontale scrollbar op. Daarom wordt hier een maximumbreedte opgegeven.

Een breedte in procenten is normaal genomen ten opzichte van de ouder van het element. Dat is hier `<body>`, een blok-element. Een blok-element wordt normaal genomen automatisch even breed als z'n ouder. Dat is `<html>`, het buitenste element. Omdat `<html>` het buitenste element is, wordt dit normaal genomen even breed als het venster van de browser. Uiteindelijk wordt `<main>` hierdoor nooit breder dan 94% van de breedte van het venster, ongeacht hoe breed dit is.

`margin: 20px auto 0;`

Omdat voor links geen waarde is opgegeven, krijgt dit automatisch dezelfde waarde als rechts. Hier staat dus eigenlijk `20px auto 0 auto` in de volgorde boven – rechts – onder – links.

Aan de bovenkant wat afstand tot de bovenkant van het browservenster. Aan de onderkant geen marge. Links en rechts `auto`, wat hier hetzelfde betekent als evenveel. `<main>` staat hierdoor altijd horizontaal gecentreerd binnen z'n ouder `<body>`.

Omdat `<body>` een blok-element is, wordt dit normaal genomen automatisch even breed als z'n ouder `<html>`. Omdat `<html>` het buitenste element is, wordt dit normaal genomen even breed als het browservenster. Hierdoor staat `<main>` altijd horizontaal gecentreerd binnen het venster, ongeacht hoe breed het venster is.

audio

Het element dat zorgt voor het weergeven van het geluid en het tonen van de knoppen van de speler. Dit element is alleen zichtbaar, als het in de html het attribuut `controls` heeft gekregen. Als het onzichtbaar is, kan het geluidsbestand alleen worden aangestuurd met behulp van JavaScript.

Waarom hier zo weinig css voor het `<audio>`-element staat, is te vinden bij [Opmerkingen](#).

`display: block;`

`<audio>` is een inline-element, net zoals tekst. Weliswaar een wat eigenaardig inline-element met aparte eigenschappen, maar het blijft een inline-element.

Aan de onderkant van tekst is een kleine extra ruimte om letters als de g en de j weer te kunnen geven. Omdat `<audio>` een inline-element is, is deze ruimte ook aanwezig onder de knoppen. Dit levert een lelijke kier tussen de knoppen en de eronder zittende tekst op.

De simpelste manier om dit op te lossen is het veranderen van `<audio>` in een blok-element.

`width: 100%;`

Een breedte in procenten is normaal genomen opzichte van de ouder van het element, dat is hier `<main>`.

Hier iets boven heeft `<main>` een breedte en een maximumbreedte gekregen. `<audio>`, en daarmee de knoppen, vullen altijd de volle breedte van `<main>`, ongeacht de breedte van `<main>`. En omdat `<main>` hierboven ook horizontaal is gecentreerd binnen het venster van de browser, staat ook `<audio>` horizontaal gecentreerd binnen het venster.

`#tekst`

Het element met `id="tekst"`. Hier zit de tekst onder de knoppen in.

`background: white;`

Witte achtergrond.

`color: black;`

Voorgrondkleur zwart. Dit is o.a. de kleur van de tekst.

Hoewel dit de standaardkleur is, wordt deze toch specifiek opgegeven. Hierboven is een achtergrondkleur opgegeven. Sommige mensen hebben zelf de voorgrond- en/of achtergrondkleur veranderd, bijvoorbeeld omdat ze slecht kleuren kunnen onderscheiden. Als nu de achtergrondkleur wordt veranderd, maar niet de voorgrondkleur, loop je het risico dat tekstkleur en achtergrondkleur te veel op elkaar gaan lijken.

Door beide op te geven, is redelijk zeker dat achtergrond- en tekstkleur genoeg van elkaar blijven verschillen. Als de gebruiker `!important` heeft gebruikt in een eigen stylesheet, is er nog niets aan de hand, want dan veranderen achtergrond- en voorgrondkleur geen van beide.

Dit is ook al bij `<body>` opgegeven, maar sommige mensen hebben bij alle elementen de kleuren veranderd. Het heeft immers weinig zin, als ze dat alleen bij de body doen, terwijl de sitebouwer de kleuren ook bij bijvoorbeeld de paragrafen heeft aangepast.

`text-align: center;`

Alle tekst horizontaal centreren.

`border: black solid 1px;`

Zwart randje.

`padding: 3px;`

Kleine afstand tussen buitenkant van en tekst in de `<div>`.

`p`

Alle `<p>`'s.

`margin: 0;`

Een `<p>` heeft van zichzelf een marge aan boven- en onderkant. Die wordt hier weggehaald.

p + p

Voor dit element is eerder css opgegeven. Deze wordt binnen dit blokje herhaald in de volgorde, waarin deze in de stylesheet staat, zodat alles hier overzichtelijk bij elkaar staat.

p {margin: 0;}

p: alle <p>'s.

+: het element achter de + moet in de html direct volgen op het element voor de +. In dit geval gaat het om een <p> die gelijk op een andere <p> volgt. Beide elementen moeten ook nog dezelfde ouder hebben.

p: alle <p>'s.

De hele selector in normale mensentaal: doe iets met elke <p> die in de html direct op een andere <p> volgt. In dit voorbeeld zijn slechts twee <p>'s aanwezig: de tekst onder de speler staat erin.

font-size: 0.8em;

Iets kleinere lettergrootte. Als eenheid wordt de relatieve eenheid em gebruikt, omdat bij gebruik van een absolute eenheid zoals px niet alle browsers de lettergrootte kunnen veranderen. Zoomen kan wel altijd, ongeacht welke eenheid voor de lettergrootte wordt gebruikt.

margin-top: 6px;

Kleine afstand tot de <p> erboven.

## HTML

De code is geschreven in een afwijkende lettersoort. De code die te maken heeft met de basis van dit voorbeeld (essentiële code) is in de hele uitleg blauw gekleurd. Alle niet-essentiële code is zwart. (In de inhoudsopgave staat alles in een gewone letter vanwege de leesbaarheid.)

<!DOCTYPE html>

<html lang="nl">

<head>

<meta charset="utf-8">

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

<link rel="stylesheet"  
href="126-css-dl/afbeelding-126-dl.css">

<meta name="description" content="Geluid afspelen met behulp  
van het <audio>-element - voorbeeld. Gebruikt  
alleen html en css.">

<title>Geluid afspelen met behulp van het  
<audio>-element - voorbeeld</title>

</head>

<body>

<main>

<audio controls preload="metadata">

<source src="126-files-dl/AmazingGrace.ogg">

```

        type="audio/ogg">
        <source src="126-files-dl/AmazingGrace.mp3"
        type="audio/mpeg">
        Je browser ondersteunt het afspelen van geluid niet. Je
        kunt hieronder nog wel de muziek downloaden.
</audio>
<div id="tekst">
<p><a href="https://audionautix.com/"><span lang="en">Amazing
    Grace</span> van <span lang="en">Jason
    Shaw</span></a></p>
<p>Muziek downloaden? Kies <a
    href="126-files-dl/AmazingGrace.ogg" download="Amazing
    Grace van Jason Shaw.ogg" title="Download muziek in ogg-
    formaat">ogg</a> (2,5 <abbr title="megabyte">MB</abbr>)
    of <a href="126-files-dl/AmazingGrace.mp3"
    download="Amazing Grace van Jason Shaw.mp3"
    title="Download muziek in mp3-formaat">mp3</a> (6,1
    <abbr>MB</abbr>).</p>
</div>
</main>
</body>
</html>

```

## CSS

De code is geschreven in een afwijkende lettersoort. De code die te maken heeft met de basis van dit voorbeeld (essentiële code) is in de hele uitleg **blauw** gekleurd. Alle niet-essentiële code is zwart. (In de inhoudsopgave staat alles in een gewone letter vanwege de leesbaarheid.)

```

/* afbeelding-126-dl.css */
body {
    background: #ff9;
    color: black;
    font-family: Arial, Helvetica, sans-serif;
    font-size: 110%;
    margin: 0;
    padding: 0;
}

main {
    display: block;
    width: 640px;
    max-width: 94%;

```



```
        margin: 20px auto 0;
    }

    audio {
        display: block;
        width: 100%;
    }

    #tekst {
        background: white;
        color: black;
        text-align: center;
        border: black solid 1px;
        padding: 3px;
    }

    p {margin: 0;}

    p + p {
        font-size: 0.8em;
        margin-top: 6px;
    }
```