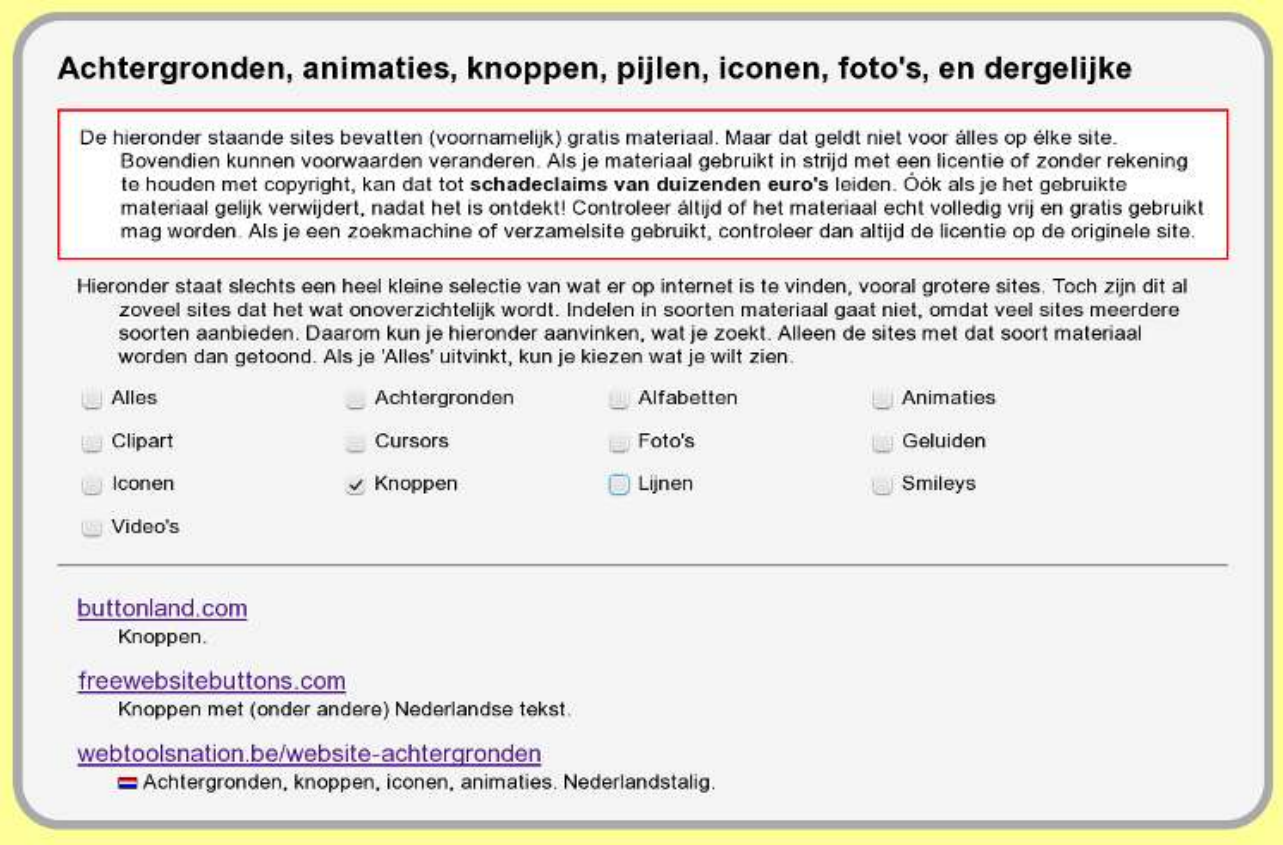
www.css-voorbeelden.nl

Met behulp van aankruisvakjes delen van een lijst verbergen of tonen



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.
* 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.
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 `rood` gekleurd. Alle niet-essentiële code is `zwart`. (In de inhoudsopgave staat alles vanwege de leesbaarheid in een gewone letter.)

Inhoudsopgave

Korte omschrijving

In een <ul> zit een hele serie links. Elke link heeft één of meer trefwoorden. Door de boven de links staande aankruisvakjes aan- of uit te vinken, worden links groepsgewijs verborgen of getoond.

Opmerkingen

De herkomst van het bij sommige links gebruikte Nederlandse vlaggetje weet ik niet meer. Maar omdat ik altijd publiek domein materiaal gebruikt, lijkt de kans dat Rutte een proces wegens plagiaat gaat aanspannen niet erg groot. Het kan dus veilig worden gebruikt. Maar als je toch twijfelt, of het is toegestaan de Nederlandse vlag te verspreiden, dan kun je veiligheidshalve natuurlijk eerst juridisch advies inwinnen.

De links in dit voorbeeld zijn nep: ze leiden allemaal naar dezelfde achterliggende pagina. Alleen de zichtbare tekst in de links is 'echt', het onzichtbare deel van de link is vervangen. Links op internet hebben nogal de neiging te veranderen, vandaar. De 'echte' links zijn, voor zover ze niet zijn gewijzigd, te vinden op de pagina met links.

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

Dit voorbeeld 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

In een gewone <ul> staat een serie links, die verwijzen naar andere sites met afbeeldingen, video's, geluiden, enz. Allerlei spullen die je bij het maken van 'n site nodig kunt hebben. Dit zijn zoveel links dat het wat onoverzichtelijk wordt, als je zoekt naar één specifiek iets. Omdat veel sites meerdere dingen aanbieden, kan het ook niet per onderwerp worden gesorteerd, want dan zouden er veel te veel links twee keer of nog vaker moeten worden vermeld.
Deze situatie deed zich op de pagina met links voor. Daar komt dit voorbeeld ook vandaan. Het is wel iets aangepast, omdat het geen onderdeel meer is van die veel grotere pagina met links.
Het zou handig zijn, als je de links met behulp van trefwoorden zou kunnen verbergen. Als je naar lijnen zoekt, zou je dan alleen de sites met lijnen zien. Of als je naar lijnen óf letters zoekt, alleen de sites met lijnen en/of letters. Enz. Met behulp van de zogenaamde 'checkbox hack' is dat mogelijk.
Eerst krijgt elke link één of meer trefwoorden. Dat kan met behulp van het data-attribuut. Als een attribuut met `data-` begint, wordt dit door de browser volledig genegeerd. Je kunt er mee doen en laten, wat je wilt. Het is te benaderen met JavaScript, maar ook met css.

De trefwoorden zitten in dit voorbeeld in het attribuut `data-soort`, maar dat had ook `data-leve-marx` of `data-gaat-henen-marx` mogen heten, als het maar met `data-` begint.
De eerste link gaat naar een site met nogal veel categorieën. De <li> waar deze link in staat, ziet er als volgt uit:

```
data-soort="achtergrond, alfabet, animatie, clipart,
     cursor, foto, lijn, smiley"
```

De tweede link gaat naar een site met alleen animaties. De <li> bij deze link ziet er zo uit:

```
<li data-soort="animatie">
```

De eerste link heeft binnen `data-soort` een hele serie trefwoorden, de tweede maar eentje. In de eerste link worden de trefwoorden gescheiden door een spatie en een komma, maar dat hoeft niet. Afhankelijk van welke selector je gebruikt, zou je ze zelfs allemaal tegen elkaar aan kunnen zetten. Met komma's en spaties vind ik het duidelijkst, dus daarom heb ik het zo gedaan.

Standaard worden alle links verborgen: bij openen van de pagina is geen enkele link zichtbaar (dat klopt niet, maar daar kom ik nog op terug. We doen even, alsof dit zo is).

Voor de links worden aankruisvakjes (<input type="checkbox">) gezet. Bij elk aankruisvakje komt een <label> met een categorie. Op die manier kun je één of meer categorieën aanvinken.
Als je nu een selector maakt die kijkt of een bepaald aankruisvakje is aangevinkt, kun je vervolgens links met een bepaald bij dat aankruisvakje horend trefwoord tonen. De volgende selector maakt de links met het trefwoord 'achtergrond' zichtbaar:

```
#toon-achtergrond:checked ~ ul
     li[data-soort*=achtergrond] {display: block;}
```

In normale mensentaal staat er: als het aankruisvakje met id='toon-achtergrond' is aangevinkt, laat dan de 's met het trefwoord 'achtergrond' in `data-soort` zien.

Mensen die de Tab-toets gebruiken om van aankruisvakje naar aankruisvakje te gaan, worden hier niet vrolijk van. Hoewel, het valt hier nog mee met dertien aankruisvakjes. Maar als je 120 aankruisvakjes hebt, moet je 120 keer die Tab-toets indrukken, voordat je al die aankruisvakjes gepasseerd bent en bij de eerste link aankomt.
Daarom worden de aankruisvakjes in eerste instantie verborgen. Er is slechts één aankruisvakje zichtbaar met in de bijbehorende label als tekst 'Alles'. Omdat dit aankruisvakje in de html het sleutelwoord `checked` heeft gekregen, is het bij openen van de pagina al aangevinkt.
Dit aankruisvakje 'Alles' zorgt ervoor dat bij openen van de pagina alle links zichtbaar zijn, en dat tegelijkertijd de aankruisvakjes voor de categorieën zijn verborgen. Pas als dit aankruisvakje voor 'Alles' wordt uitgevinkt, worden de andere aankruisvakjes zichtbaar. En worden de links verborgen. Die links kunnen weer zichtbaar worden gemaakt door het bij de betreffende categorie horende aankruisvakje is aan te vinken.
Nu hoeven gebruikers van de Tab-toets maar één keer de Tab-toets in te drukken, als ze alles willen zien. Omdat de andere aankruisvakjes met behulp van `display: none;` worden verborgen, worden die genegeerd door de Tab-toets, als 'Alles' is aangevinkt. Pas als 'Alles' wordt uitgevinkt en de de andere aankruisvakjes weer zichtbaar worden, worden ze bezocht door de Tab-toets.
Ook een schermlezer negeert met behulp van `display: none;` verborgen elementen. Daarom worden de aankruisvakjes voor de categorieën niet voorgelezen, tenzij het

aankruisvakje ’Alles’ is uitgevinkt. Als dat gebeurt worden de andere aankruisvakjes zichtbaar en worden ze niet meer genegeerd door de schermlezer.
Speciale aanpassingen voor schermlezers of toetsenbordgebruikers zijn dan ook niet nodig.

Deze constructie heeft twee nadelen, die worden veroorzaakt omdat de <input> niet binnen de bijbehorende <label> kan worden gezet. Om de benodigde selector te kunnen gebruiken, moet de <input> dezelfde ouder hebben als de <ul>, waarbinnen de ’s staan:

```
#toon-achtergrond:checked ~ ul
     li[data-soort*=achtergrond] {display: block;}
```

(Voor het geval je je nu een acute hartverzakking bent geschrokken: deze selector wordt later zorgvuldig in stukjes gehakt, waarna elk stukje uitgebreid wordt beschreven. Deze regel hoort bij de keuze ’Achtergronden’, maar het principe is bij alle aankruisvakjes hetzelfde.)
Hierin is `#toon-achtergrond` een <input>. De achter de ~ staande `ul` heeft dezelfde ouder (<main>) als de <input>. Als dat niet zo is, werkt de ~ niet, en daarmee werkt ook de hele constructie niet. Als de <input> binnen de bijbehorende <label> wordt gezet, heeft de <input> als ouder niet meer <main>, maar <label>. En werkt de constructie dus niet meer.
De <label>’s worden met behulp van het for-attribuut aan de bijbehorende <input> gekoppeld. Dat is in dit geval belangrijk, omdat schermlezers e.d. anders niet weten, welk <label> bij welke <input> hoort. Bovendien kun je nu ook de <label> aanraken of -klikken en hoef je niet precies op dat hele kleine aankruisvakje te mikken.

Dit stukje begon met die twee nadelen. Dat eerste nadeel komt dan eindelijk hier. Als de <input> binnen de <label> staat, staat de tekst van de <label> altijd op de goede hoogte ten opzichte van de <input>. Als de <input> buiten de <label> staat, is dat niet altijd het geval. Er zijn kleine afwijkingen naar boven of beneden. Dit verschilt per browser en besturingssysteem.
Dit zou je op kunnen lossen door het vakje van de eigenlijke <input> onzichtbaar te maken. In plaats daarvan toon je dan een afbeelding, die vanuit de <label> wordt getoond, of iets soortgelijks. Dat heb ik niet gedaan, omdat het voorbeeld dan wel heel ingewikkeld zou worden.

En dan hadden we nog het tweede nadeel tegoed. Als de <input>’s binnen de bijbehorende <label> staan, is het makkelijk de <label>’s horizontaal te verdelen. Als er bijvoorbeeld twee naast elkaar moeten staan, maak je ze 50% breed. Dat kan nu niet, omdat de <input>’s naast de <label> komen te staan. Als je de <label>’s 50% breed maakt, is er geen ruimte meer voor de <inputs>.
Het geven van een breedte aan de <input> werkt ook niet, omdat sommige browsers dan het hele aankruisvakje breder maken, maar andere browsers een soort marge toevoegen.
Omdat er ruimte moet overblijven voor de <input>’s, mogen de <label>’s niet de volle breedte van het browservenster vullen. Hierdoor is het niet goed mogelijk ze in alle verschillende breedtes echt netjes over de breedte van het browservenster te verdelen. Dit wordt nog bemoeilijkt, omdat de heren en dames browsermakers allemaal andere ideeën hebben over breedte e.d. van een aankruisvakje. Wat ook logisch is, want dit is natuurlijk zo’n ongelooflijk gezichtsbepalend onderdeel van de browser, dat je daar moeilijk afspraken over kunt maken met je concurrenten. Stel je voor, zeg.
Uiteindelijk werkt het het beste om de <label>’s en <input>’s naar links te floaten en de <label>’s een bepaalde breedte te geven, die afhankelijk is van de breedte van het venster van de browser. Daarnaast kan een marge bij de <input> soms ook helpen.
Overigens is dit tweede nadeel heel betrekkelijk. Omdat de lengte van de teksten in de <label>’s verschilt, is het voor het oog hoe dan ook nooit netjes over de breedte verdeeld.

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, Opera, Safari en Android browser: `-webkit-`.
(Google Chrome is van webkit overgestapt op een eigen weergave-machine: blink. Blink gaat geen voorvoegsels 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 blink.)
Internet Explorer: `-ms-`, naar de maker: Microsoft.

In dit voorbeeld worden `@keyframes` en `animation` 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-linear-gradient`. Zodra Android browser `linear-gradient` gaat herkennen, zal dit `-webkit-linear-gradient` overrulen, omdat het er later in staat. Dat ze er beide in staan, is dus geen enkel probleem.

`@-webkit-keyframes` en `-webkit-animation`:

Deze twee eigenschappen worden alleen maar gebruikt om een bug in sommige oudere op webkit gebaseerde browsers te repareren. Daarom hoeft alleen maar op browsers gelet te worden, die op webkit zijn gebaseerd. In dit geval is daarom het volgende voldoende:

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

Hetzelfde geldt voor `animation`:

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

In nieuwere browsers is deze bug niet meer aanwezig, dus de versie zonder voorvoegsel hoeft in dit geval helemaal niet gebruikt te worden.
(In het algemeen is het een bijzonder slechte gewoonte om van een eigenschap alleen één bepaalde versie te gebruiken. Dit gebeurt nogal eens voor iOS, waarmee Apple actief wordt geholpen om sites e.d. ontoegankelijk te maken voor andere browsers dan Safari. Ontwikkelaars die dit doen, werken mee aan de totstandkoming van eenzelfde wantoestand als in het verleden met het monopolie van Internet Explorer 6 heeft bestaan.
Maar in dit geval maakt het niet uit, omdat het alleen om een bug gaat in een browser die op webkit is gebaseerd. Andere browsers hebben deze css helemaal niet nodig.)

Inmiddels is de algemene mening dat ’vendor prefixes’, zoals deze voorvoegsels in het Engels heten, geen groot succes zijn. Eén van de grootste problemen: veel sitemakers gebruiken alleen de `-webkit-`variant. Daar kwamen ze in het verleden nog mee weg, omdat Apple op mobiel zo’n beetje ’n monopolie had. Inmiddels is dat niet meer zo, maar deze gewoonte bestaat nog steeds. Waardoor ’n site alleen in op webkit georiënteerde browsers goed is te bekijken.

Dit is zo'n groot probleem dat andere browsers soms de variant met `-webkit-` ook maar zijn gaan implementeren, naast de standaard. Want als 'n site het niet goed doet in 'n bepaalde browser, krijgt in de regel niet de site maar de browser de schuld.
Voorlopig zijn we echter nog niet van deze voorvoegsels af. Als je ze gebruikt, gebruik dan álle varianten, en eindig met de variant zonder voorvoegsel, zoals die uiteindelijk ooit gebruikt gaat worden. Als je alleen de `-webkit-`variant gebruikt, ben je in feite 'n onbetaalde reclamemaker voor Apple. (In dit voorbeeld ligt dat dus, zoals iets hierboven beschreven, anders, omdat de `-webkit-`variant alleen wordt gebruikt voor een bug in een aantal oudere browsers.)
Vanwege alle problemen met 'vendor prefixes' worden deze door steeds meer browsers niet meer gebruikt. Nieuwe, experimentele css-eigenschappen zitten inmiddels in bijvoorbeeld Firefox, Google Chrome en Safari achter een zogenaamde vlag: de gebruiker moet iets veranderen in de instellingen, waarna de eigenschap gebruikt (en getest) kan worden. Als alles werkt, zoals het hoort te werken, schakelt de browsermaker de vlag standaard in.

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.
  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 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.
- 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 mogelijkheden voor het simuleren van weergave op een kleiner scherm in de ontwikkelgereedschappen 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 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. Deze zijn te vinden op de site van Mozilla onder Webontwikkeling. 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, 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.
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 ikzelf op test. Ook van Internet Explorer kun je meerdere versies naast elkaar draaien. Op de pagina met links staan onder de kopjes Gereedschap → Meerdere versies van Internet Explorer draaien en Gereedschap → Weergave testen 'n aantal links die daarbij 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. 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
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.
- `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 `margin`, `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 een absolute eenheid als `px` kun je ook een relatieve eenheid gebruiken, met name `em`. Voordeel van `em` is dat een lettergrootte, regelhoogte, e.d. in `em` 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` 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.
  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.
  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.
  css kun je valideren op: jigsaw.w3.org/css-validator.

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.

Op pagina's waar dat nuttig is, wordt op deze site een skip-link gebruikt.

- 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 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 of SSI (Server Side Includes). 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.
- 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. 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.
- 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.

- Tenslotte kun je je pagina nog online op toegankelijkheid laten controleren op 'n behoorlijk aantal sites. Ik noem er hier enkele:
  lowvision.support
  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
  Voor dit voorbeeld zijn geen speciale aanpassingen nodig.
  Een skip-link is niet nodig, want als 'Alles' wordt aangevinkt, worden de aankruisvakjes voor de categorieën verborgen met `display: none;`, waardoor de Tab-toets ze negeert. Hetzelfde geldt voor schermlezers.
  Zonder css zie je (nutteloze) aankruisvakjes, waaronder gewone normale tekst en links staan. Datzelfde geldt voor de tekstbrowser Lynx, waarin de aankruisvakjes ook geen enkel nut hebben. In WebbIE daarentegen blijken ze wel te werken.

Getest in

Laatst gecontroleerd op 16 juli 2016
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 hierboven bij Toegankelijkheid en zoekmachines onder het kopje Specifiek voor dit voorbeeld.
Dit voorbeeld is getest op de volgende systemen:

- Windows 7:
  Firefox, UC Browser, Google Chrome, Internet Explorer 9 en 10, in meerdere resoluties.
- Windows 8.1:
  Bureaublad-versie: Firefox, UC Browser, Google Chrome en Internet Explorer 11, in meerdere resoluties.
  Startscherm-versie: Internet Explorer 11.
- Windows 10:
  Firefox, UC Browser, Google Chrome, Internet Explorer 11, Edge, in meerdere resoluties.
- OS X:
  Firefox, Safari en Google Chrome, in meerdere resoluties.
- Linux:
  Firefox en Google Chrome, in meerdere resoluties.
- Windows Phone 8.1 in een resolutie van 800x480 (Nokia Lumia 520):
  Internet Explorer (portret en landschap).
  UC Browser (portret en landschap).
- iPad met iOS 9.3.2 in een resolutie van 1024x768 (device-pixel-ratio: 1) (MC979NF):
  Safari, Chrome for iOS, UC Browser, Firefox (alle portret en landschap).
  Opera mini (turbo mode) portret en landschap.
- iPad Air met iOS 9.3.2 in een resolutie van 2048 x 1536 (retina, device-pixel-ratio: 2) (MD785HC/B):

Safari, Chrome for iOS, UC browser, Firefox (alle portret en landschap).
Opera mini (turbo mode) portret en landschap.

- Android 4.1.2 in een resolutie van 800x480 (Samsung Galaxy Core i8620):
  Chrome, Android browser, UC Browser en Firefox (alle portret en landschap).
  Opera mini (besparingsstand hoog) portret en landschap.
- Android 4.4.2 in een resolutie van 1280x800 (resolutie: 90-99 dpi) (Samsung Galaxy Tab 3 GT-P521):
  Android browser, UC Browser, Firefox en Chrome (alle portret en landschap).
  Opera mini (besparingsstand hoog) portret en landschap.
- Android 4.4.2 in een resolutie van 2560x1600 (hoge resolutie: 190-199 dpi) (Samsung Galaxy Tab PRO (T520):
  Android browser, UC Browser, Firefox en Chrome (alle portret en landschap).
  Opera mini (besparingsstand hoog) portret en landschap.

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.
Ik maak één uitzondering: Android browser. Omdat Android vaak niet geüpdatet kan worden, test ik ook nog in oudere versies van Android browser.
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 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 8.1 en 10 is getest met een touchscreen, met een combinatie van toetsenbord en touchpad, en met een combinatie van toetsenbord en muis.

Als dat relevant is, is op de desktop ook getest, als JavaScript uitstaat. Eventuele problemen staan hierboven bij 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, VoiceOver en ChromeVox.
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 Windows 7.
NVDA is een schermlezer, zoals die door blinden wordt gebruikt. Er is getest in combinatie met Firefox op Windows 7.
TalkBack is een in Android ingebouwde schermlezer. Er is getest in combinatie met Chrome.
VoiceOver is een in iOS en OS X ingebouwde schermlezer. Er is getest in combinatie met Safari op iOS en OS X.
ChromeVox is een schermlezer in de vorm van een extensie bij Google Chrome. Er is getest op een systeem met Kubuntu Linux.

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 hierboven bij 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.

Alle browsers

De tekst naast het aankruisvakje staat soms iets te hoog of te laag

De tekst die bij een aankruisvakje hoort, staat soms iets hoger of lager dan het aankruisvakje zelf. Deze afwijking kan bij een andere lettergrootte nog iets groter of juist iets kleiner worden. Het gaat niet om hele grote afwijkingen. Het is wat lastig om dit op te lossen, omdat verschillende browsers verschillende algoritmes gebruiken om regelhoogte e.d. bij een aankruisvakje te berekenen. Omdat dit voorbeeld zo al ingewikkeld genoeg is, is hier verder niets aan gedaan. Meer hierover is te lezen bij Deze constructie heeft twee nadelen..., waar ook suggesties voor een oplossing staan.

De kolommen staan niet helemaal gelijkmatig verdeeld over het venster

Dit heeft dezelfde oorzaak als wat bij wat gelijk hierboven staat over de tekst. Ook hierover is meer te lezen bij Deze constructie heeft twee nadelen..., waar ook suggesties voor een oplossing staan.

Validatie

Het in de css gebruikte `@-webkit-keyframes` levert een foutmelding op bij de css-validator. Omdat de reden van deze foutmelding precies bekend is, is dit verder niet van belang. De op webkit gebaseerde browsers die dit nodig hebben, accepteren

het gewoon. Andere browsers negeren het. Behalve mogelijk een verminderd gevoel van eigenwaarde heeft dit dus verder geen enkele nadelige bijwerking.

Wijzigingen

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

16 juli 2016:

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 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 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.

lijst-113-dl.html: de pagina met het voorbeeld.

lijst-113.pdf: deze uitleg (aangepast aan de inhoud van de download).

lijst-113-inhoud-download-en-licenties.txt: een kopie van de tekst onder dit kopje (Inhoud van de download en licenties).

113-css-dl:

- lijst-113-dl.css: stylesheet voor lijst-113-dl.html.
- lijst-113-hulp-dl.css: stylesheet voor lijst-113-hulp-dl.html.

113-files-dl:

- lijst-113-hulp-dl.html: de pagina achter de links.

113-pics:

- vlag.png: het Nederlandse vlaggetje. Ik mag hangen, als ik nog weet waar dat vandaan komt. Maar ik weet wel zeker dat het vrij gebruikt mag worden, dus ga je gang.

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 `rood` 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.
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 (`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.
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 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 vensters van mobiele apparaten. Op deze pagina bijvoorbeeld wordt het aantal aankruisvakjes dat naast elkaar staat, aangepast aan de breedte van het venster van de browser.
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 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 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 gebruik ik niet. De bezoeker kan zelf nog gewoon zoomen, wat belangrijk is voor mensen die wat slechter zien.

```
<link rel="stylesheet" href="113-css-dl/menu-113-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 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 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 nauwelijks nut, want er is maar één pagina. In dit geval kun je daarom de css beter in de <head> van de html-pagina zelf zetten. Voor de omvang maakt het hier nauwelijks 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 je hoeft de css slechts 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> van het voorbeeld, tussen <style> en </style>:

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

Maar zodra een iets groter 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 ik hierboven omstandig beweer dat dat in dit voorbeeld eigenlijk geen nut heeft: overzichtelijkheid. Nu kun je html en css los van elkaar bekijken.)

De hulppagina achter de links heeft een eigen stylesheet: lijst-113-hulp-dl.css. Deze wordt hier verder niet besproken, omdat hij uiterst simpel is. Op de site is deze met de stylesheet voor het voorbeeld zelf (en nog andere pagina's) gecombineerd tot één gezamenlijke stylesheet.

```
<li data-soort="achtergrond, alfabet, animatie, clipart, cursor,
    foto, lijn, smiley">
```

Elke link staat binnen een eigen <li>. Aan elk van die 's is het attribuut `data-soort` toegevoegd. Als een attribuut met `data-` begint, wordt dit door de browser volledig genegeerd. Achter het koppelteken kun je elk woord toevoegen dat je wilt. Je kunt dus ook meerdere data-attributen aan hetzelfde element toevoegen. Als ze maar met `data-` beginnen.
Na het isgelijkteken komt, tussen aanhalingstekens, de waarde van het data-attribuut te staan. Dat mogen, zoals hierboven, meerdere waarden zijn, gescheiden door een komma. (Die komma is trouwens niet nodig. Als er gewoon een spatie tussen staat, is dat voldoende. En zelfs die spatie is, afhankelijk van de gebruikte selector, niet nodig. Ik gebruik een spatie en komma, omdat ik dat duidelijker vind.)
Het data-attribuut geeft de mogelijkheid elementen aan te spreken met css, JavaScript, e.d., met behulp van eigen bedachte attribuut-namen.
In dit voorbeeld is de waarde van het data-attribuut de categorie (of categorieën), die je op de betreffende site kunt vinden. Door te kijken welke <input>'s zijn aangevinkt, kun je vervolgens de 's tonen, die in het data-attribuut de waarde hebben staan, die bij een van de aangevinkte <input>'s hoort.

```
<label for="toon-alles"> Alles</label>
```

Elke <input> heeft een bijbehorende <label>. De hierboven staande <label> hoort bij de <input> voor 'Alles'.
Omdat de <input>'s niet binnen hun bijbehorende <label> kunnen staan, is het for-attribuut hier uiterst belangrijk. Met `for="toon-alles"` wordt deze <label> gekoppeld aan de <input> met id="toon-alles". Zonder deze koppeling weten schermlezers e.d. niet, welke <label> en daarmee welke tekst bij welke <input> hoort. Bovendien kun je nu ook de <label> aanraken of -klikken om aan- of uit te vinken, zodat je niet precies op dat kleine aankruisvakje hoeft te mikken.
In deze <label> staat de tekst 'Alles'. Daarvoor staat ` ` een zogenaamde 'harde' of 'vaste' spatie ('no-break space'). Omdat deze <label>'s naar links worden gefloat, zou een gewone spatie verdwijnen. Daardoor komt de tekst in de <label> te dicht tegen het aankruisvakje te staan. Een vaste spatie blijft wel gewoon staan, waardoor je net wat afstand tussen vakje en tekst krijgt.

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 `rood` 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. Als je op internet zoekt naar 'css' en 'compress' of 'comprimeren', vind je tal van sites, waar je dat automatisch kunt laten 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.)

css voor alle vensters

```
/* lijst-113.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.

```
@-webkit-keyframes bugfix-3 {from {padding-left: 3px;} to
     {padding-left: 3px;}}
```

Bij main is een animatie toegevoegd om een bug in oudere versies van Android browser op te lossen. Die animatie wordt hier opgegeven.
Op een aantal plaatsen wordt in een selector gebruik gemaakt van het teken ~ (de 'general sibling'-selector) om een <li> te tonen of te verbergen. Genoemde browser toont of verbergt

de <li> echter niet, omdat hij problemen heeft met de ~. Door het toevoegen van een animatie wordt de <li> toch getoond of verborgen.
Als je nou denkt dat hier niets wordt uitgevoerd, omdat de padding links van 3 px in een padding links van 3 px wordt veranderd, dan heb je helemaal gelijk. Toch neutraliseert deze flauwekul de bug. De padding is 3 px, omdat dit de padding is die bij main aan alle kanten wordt opgegeven. Zou je hier een andere waarde nemen, dan zou de waarde van de padding links die andere waarde krijgen (zoals gelijk hieronder bij `@-webkit-keyframes bugfix-30` gebeurt).
Omdat er in feite helemaal niets gebeurt bij deze animatie, wordt de pagina ook niet opnieuw opgemaakt door de browser. Kennelijk werkt dit als 'n soort waakhond: áls er iets is verandert, wordt het weergegeven. Anders gebeurt er niets.
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 een op webkit gebaseerde browser. 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-.

```
@-webkit-keyframes bugfix-30 {from {padding-left: 30px;} to
    {padding-left: 30px;}}
```

Het verhaal hiervoor is precies hetzelfde als gelijk hierboven bij de andere `@-webkit-keyframes`. Alleen is de waarde van de padding-left hier geen 3 px, maar 30 px. Deze grotere padding is bedoeld voor browservensters breder dan 760 px en komt overeen met de bij main opgegeven linker-padding van 30 px.

`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.

`margin: 0; padding: 0;`

Slim om te doen vanwege verschillen tussen browsers.

`main`

Alle <main>'s. Dat is er maar eentje. De belangrijkste inhoud van de pagina staat hierin. In dit voorbeeld zijn dat de links met de erboven staande teksten, aankruisvakjes, e.d.

`-webkit-animation: bugfix-3 infinite 1s;`

In oudere versies van Android browser zit een bug, waardoor de 's niet worden verborgen of getoond bij uit- of aanvinken van een aankruisvakje. Met behulp van deze regel gebeurt dat toch. Meer hierover bij @-webkit-keyframes bugfix-3.
Het sleutelwoord `infinite` zorgt ervoor dat de animatie eindeloos doorgaat. `1s` geeft aan dat de animatie 1 seconde duurt (voordat deze weer opnieuw wordt afgespeeld.)

`background: #f5f5f5;`

Lichtgrijze achtergrondkleur.

`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 álle 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.

`display: block;`

Oudere browsers kennen <main> niet. Onbekende elementen worden standaard als inline-element weergegeven. Daarom wordt hier expliciet gezegd dat dit een blok-element is.

`margin: 20px auto;`

Omdat voor onder en links geen waarde is opgegeven, krijgen die automatisch dezelfde waarde als boven en rechts. Hier staat dus eigenlijk `20px auto 20px auto` in de volgorde boven – rechts – onder – links. Boven- en onderaan wat ruimte tussen de boven- en onderkant van het browservenster en de inhoud ervan.
Links en rechts `auto`, wat hier hetzelfde betekent als evenveel. Hierdoor staat <main> altijd horizontaal gecentreerd binnen zijn ouder <body>. <body> is een blok-element en wordt daardoor normaal genomen automatisch even breed als zijn ouder <html>, ook weer een blok-element. Omdat <html> het buitenste element is, wordt dit normaal genomen even breed als het venster van de browser. Hierdoor staat <main> altijd horizontaal gecentreerd binnen het venster van de browser, ongeacht de breedte van het venster.
In smallere vensters merk je hier niets van, omdat <main> de volle breedte van het venster vult. Maar bij main wordt voor bredere vensters een breedte van 80% opgegeven voor <main>, en dan zie je het centreren wel.
Deze manier van horizontaal centreren van een blok-element werkt alleen, als het te centreren element een breedte heeft. Omdat dat later, als daadwerkelijk gecentreerd moet worden, bij main gebeurt, is dat geregeld.

`padding: 3px;`

Kleine ruimte tussen de buitenkant en de inhoud van de <main>.

`border-top: black solid 1px;`

Klein randje aan de bovenkant.

`h1`

Alle <h1>'s. Dat is er maar eentje: de belangrijkste kop op de pagina.

`font-size: 1.5em;`

Lettergrootte iets kleiner maken dan de standaardgrootte bij een <h1>, want die is wel héél enthousiast.

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: 0;`

Van zichzelf heeft een <h1> een marge aan boven- en onderkant. Die wordt hier weggehaald. Voor browservensters breder dan 760 px wordt later bij h1 weer een marge aan boven- en onderkant aangebracht. (Bredere vensters zullen ook hoog genoeg zijn om ruimte te hebben voor die marges.)

`.let-op`

De elementen met class="let-op". Dat is er hier maar eentje, dus er had ook een id gebruikt kunnen worden. Op de pagina met links, waar dit voorbeeld vandaan komt, staan meer elementen met deze class. Het is de met de rode rand bovenaan de pagina.

`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 álle 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.

`margin: 5px 0;`

Omdat voor onder en links geen waarde is opgegeven, krijgen die automatisch dezelfde waarde als boven en rechts. Hier staat dus eigenlijk `5px 0 5px 0` in de volgorde boven – rechts – onder – links. Boven en onder kleine marge van 5px, rechts en links geen marge.

`border: solid red 2px;`

Rode rand.

`p, li`

Alle <p>'s en 's. Er zijn slechts twee <p>'s met tekst, die beide boven de aankruisvakjes staan. Elke link staat in een eigen <li>.

`text-indent: -0.7em; margin: 0.7em;`

Op de afbeelding staat links een stukje van de pagina zonder deze regel, rechts met deze regel. Links is alles links uitgelijnd, rechts steekt de eerste regel iets naar links uit.

Eerst de marge. De marge van 0,7 em zet alles binnen de <p>'s en 's 0,7 em naar links. Die marge van 0,7 em links geeft ruimte om de eerste regel iets naar links uit te laten steken. Die eerste regel komt binnen de marge van 0,7 em te staan. Met behulp van `text-indent: -0.7em;` wordt de eerste regel, en alleen de eerste regel, 0,7 em naar links teruggezet. Dit maakt de pagina 'n stuk leesbaarder.

Als eenheid wordt `em` genomen, omdat de breedte van marge en terugzetten nu mee verandert met de lettergrootte, wat mooier is: een grotere letter krijgt een grotere marge, een kleine letter een kleinere.

Je zou dit ook met behulp van een kunnen doen, maar dan moet je precies weten hoelang die eerste regel is. En dat weet je niet, want bij een grotere of kleinere letter wordt de eerste regel langer of korter. Nu regelt de browser het en wordt het automatisch aan de lettergrootte aangepast.

`input`

Alle <input>'s. Alle aankruisvakjes.

`float: left;`

Heel vaak zal een <input> binnen de bijbehorende <label> staan. In dat geval zal normaal genomen niet de <input>, maar de <label> – en dus ook de daarin zittende <input> – op de juiste plaats worden gezet. Dat kan hier niet, omdat dan de bij de <input> horende selector niet meer werkt. De <input>'s staan hier daarom buiten de <label>'s. Daarom moeten de <input>'s hier op eigen houtje op de juiste plaats worden gezet. Dit kan het best door ze naar links te floaten. Meer hierover is te vinden bij En dan hadden we nog het tweede nadeel...

`margin-top: 5px;`

Omdat de <input>'s niet binnen de bijbehorende <label> staan, ontstaan er ook kleine afwijkingen in de hoogte tussen aankruisvakje en bijbehorende tekst in de <label>. Deze staat soms iets hoger of lager dan de bijbehorende <input>. De hoogte van de <input> aanpassen gaat niet, omdat browsers hier verschillend mee omspringen. (Zelfs binnen één browser: als je in Firefox 'n hoogte aan de <input> geeft, wordt er boven en onder het aankruisvakje ruimte toegevoegd. Behalve in Firefox op Android, waar het aankruisvakje zélf hoger wordt. Zucht.)

Deze marge aan de bovenkant plaatst in alle browsers het aankruisvakje iets omlaag. In combinatie met de hier gelijk onder bij <label> opgegeven eigenschappen voor `line-height` levert dit in alle browsers een min of meer acceptabel resultaat op. Meer hierover is te lezen bij Deze constructie heeft twee nadelen..., waar ook suggesties voor een oplossing staan.

`label`

Alle <label>'s. Elke <input> heeft een <label> met daarin de bijbehorende tekst voor de <input>. Met behulp van het for-attribuut wordt de <label> aan de bijbehorende <input> gekoppeld.

`width: 30%;`

In smallere browservensters blijkt dit de beste breedte te zijn om twee <input>'s met bijbehorende <label>'s naast elkaar te krijgen. Later worden, op bredere vensters, meer <input>'s en <label>'s naast elkaar gezet. Meer hierover is te vinden bij En dan hadden we nog het tweede nadeel...

Een breedte in procenten is altijd ten opzichte van de ouder van het element. Dat is in dit geval <main>, een blok-element. Een blok-element wordt normaal genomen even breed als z'n ouder. De ouder van <main> is <body>, ook een blok-element.
Omdat <body> een blok-element is, wordt dit normaal genomen ook weer even breed als de ouder van <body>. Dat is <html>, ook een blok-element. <html> is het buitenste element en wordt daardoor normaal genomen even breed als het venster van de browser.
Uiteindelijk wordt <label> dus nooit breder dan 30% van de breedte van het browservenster.

`float: left;`

Hier gelijk boven bij <input> zijn alle <input>'s naar links gefloat. Als de bijbehorende <label>'s niet naar links worden gefloat, komen deze met z'n allen rechts van alle <input>'s te staan. Netjes op één regel, dat wel, en als het niet meer past op de volgende regel.
Door de <label>'s zelf ook naar links te floaten, komen ze naast de juiste <input> te staan. Een iets uitgebreider verhaal staat iets hierboven bij <input> onder het kopje `float: left;`.

Een <li> is een inline-element. Daardoor zijn eigenschappen als breedte niet te gebruiken. Door het naar links te floaten verandert het in 'n soort blok-element, waardoor dit soort eigenschappen wel is te gebruiken.

`line-height: 1.5em;`

Regelhoogte. In combinatie met de iets hierboven bij <input> opgegeven `margin-top` levert dit in alle browsers een min of meer acceptabel resultaat op wat betreft de plaatsing van de tekst in de <label> ten opzichte van de bijbehorende <input>. Een uitgebreider verhaal staat hier iets boven bij <input> onder het kopje `margin-top: 5px;`.

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-bottom: 10px;`

Marge aan de onderkant van de <label>'s. Dit schept wat afstand tussen boven elkaar staande <input>'s en <label>'s, wat vooral voor touchscreens van belang is. Niet iedereen heeft de lieftallige priegelvingertjes van 'n vierjarige kleuter.

`#toon-alles ~ ul li`

Voor deze elementen 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, li {text-indent: -0.7em; margin: 0.7em;}

Met dit aankruisvakje worden in één keer alle 's, en dus de daarin zittende links, verborgen of getoond. (Met latere aankruisvakjes kunnen, als alles hier is verborgen, dan toch weer één of meer categorieën worden getoond.)

`#toon-alles`: het element met id=”toon-alles”. Dit is een <input> type ’checkbox’, een aankruisvakje. Omdat dit aankruisvakje bij ’Alles’ hoort, heeft het als id ’toon-alles’: `input#toon-alles`. De aankruisvakjes staan boven de <ul> met de ’s, niet erin. Anders zou deze selector niet kunnen werken.

~: het hierachter staande element moet ergens in de html staan, na het voor de ~ staande element. Het hoeft er, anders dan bij de +, niet gelijk op te volgen, als het maar ergens na het eerste element in de html staat.
De enige voorwaarde is verder dat het voor en het na de ~ staande element dezelfde ouder hebben. Dat is hier het geval: `#toon-alles` voor de ~ en `ul` na de ~ hebben beide als ouder <main>.

`ul`: alle ’s. Dat is er hier maar eentje. Deze `ul` mag niet ontbreken in de selector, omdat de voor en na de ~ staande elementen dezelfde ouder moeten hebben. `#toon-alles` en `ul` hebben beide als ouder <main>.
De hierna volgende `li` heeft echter als ouder niet <main>, maar <ul>. `li` kan hierdoor niet rechtstreeks worden gebruikt. Maar als we ouder `ul` van de `li` ertussen zetten, wordt wel aan de eis van dezelfde ouder voldaan. Want `ul` heeft wel dezelfde ouder als `#toon-alles`.

`li`: de ’s binnen bovenstaande <ul>. Dit zijn de ’s, waarbinnen de links staan.

Alles bij elkaar:

`#toon-alles ~ ul li`: de ’s binnen de <ul> die in de html ergens na het element (aankruisvakje) met id=”toon-alles” staat.

`display: none;`

Alle ’s verbergen.
(Bij opening van de pagina zijn de ’s gewoon zichtbaar, omdat hieronder bij #toon-alles:checked ~ ul li wordt opgegeven dat ze zichtbaar moeten zijn, als `input#toon-alles` is aangevinkt. En omdat in de html bij `input#toon-alles` het sleutelwoord `checked` is opgegeven, is deze <input> bij opening van de pagina al aangevinkt. Pas als het vinkje bij de <input> voor ’Alles’ wordt weggehaald, gaat deze selector werken en worden de ’s verborgen.)

`#toon-alles:checked ~ ul li`

Voor deze elementen 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, li {text-indent: -0.7em; margin: 0.7em;}
#toon-alles ~ ul li {display: none:}

Deze selector is precies hetzelfde als die gelijk hierboven bij #toon-alles ~ ul li, maar deze is alleen geldig als `input#toon-alles`, de <input> die bij ’Alles’ hoort, is aangevinkt (`:checked`).

`display: block;`

Toon de ’s. Als deze <input> is aangevinkt, worden in één keer alle ’s getoond, ongeacht bij welke categorie ze horen.

`#toon-alles:checked ~ input, #toon-alles:checked + label ~ label`

Voor deze elementen 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.
input {float: left; margin-top: 5px;}
label {width: 30%; float: left; line-height: 1.5em; margin-bottom: 10px;}

Eerst de eerste selector, voor de komma: `#toon-alles:checked ~ input`. Met deze selector worden de aankruisvakjes voor categorieën verborgen, als het aankruisvakje voor 'Alles' is aangevinkt.

`#toon-alles:checked`: het element met id="toon-alles". Dit is een <input> type 'checkbox', een aankruisvakje. Maar alleen als deze <input> is aangevinkt (`:checked`).

~: de hierachter staande elementen moeten ergens in de html staan, na het voor de ~ staande element. Ze hoeven er, anders dan bij de +, niet gelijk op te volgen, als ze maar ergens na het eerste element in de html staan.
De enige voorwaarde is verder dat de voor en na de ~ staande elementen dezelfde ouder hebben. Dat is hier het geval: `#toon-alles` voor de ~ en `input` na de ~ hebben alle als ouder <main>.

`input`: alle <input>'s. Dit zijn de <input>'s die bij de categorieën horen.

Alles bij elkaar:

`#toon-alles:checked ~ input`: de <input>'s die in de html ergens na het element (aankruisvakje) met id="toon-alles" staan, maar alleen als `#toon-alles` is aangevinkt.

De tweede selector, na de komma: `#toon-alles:checked + label ~ label`. Deze selector zorgt voor het verbergen van de <label>'s bij de categorieën, als het aankruisvakje voor 'Alles' is aangevinkt. (Het eerste <label>, dat bij 'Alles hoort. wordt niet verborgen door deze selector.)

`#toon-alles:checked`: het element met id="toon-alles". Dit is een <input> type 'checkbox', een aankruisvakje. Maar alleen als deze <input> is aangevinkt (`:checked`).

+: het hierachter staande element moet ergens in de html gelijk na het voor de + staande element staan. Voor de + staat `#toon-alles`, achter de + staat `label`, dus het gaat om de <label> die gelijk na `input#toon-alles` in de html staat. Dit is de <label> met de tekst 'Alles', die bij `input#toon-alles` hoort.
Als 'Alles' is aangevinkt, moeten alle <label>'s voor categorieën worden verborgen. Maar niet de <label> met 'Alles', want dan zou bij het aankruisvakje voor 'Alles' geen tekst meer staan. Door expliciet de eerste <label> (met behulp van + en ~) over te slaan, wordt dit voorkomen.

`label`: omdat deze <label> achter een + staat, moet de <label> in de html gelijk volgen op het element dat voor de + staat. Dat is `input#toon-label`. Dit is dus de <label> die in de html gelijk op <input id="toon-label"> volgt. Het is de <label> met de tekst 'Alles'.

~: de hierachter staande elementen moeten ergens in de html staan, na het voor de ~ staande element. Ze hoeven er, anders dan bij de +, niet gelijk op te volgen, als ze maar ergens na het eerste element in de html staan.
De enige voorwaarde is verder dat de voor en na de ~ staande elementen dezelfde ouder hebben. Dat is hier het geval: de `label` voor de ~ en de `label` na de ~ hebben alle als ouder <main>.

`label`: alle <label>'s. Alle? Nee, één <label> houdt dapper stand. Het laatste stukje van de selector is `label ~ label`. Het gaat dus om alle <label>'s die in de html ergens op een ander <label> volgen. Dat zijn alle <label>'s, behalve de eerste die bij de keuze voor 'Alles' hoort en altijd zichtbaar moet zijn. Die eerste <label> volgt immers niet op een andere <label>, want anders zou hij niet de eerste zijn.

Alles bij elkaar:

`#toon-alles:checked + label ~ label`: alle <label>'s die in de html op een ander <label> volgen. Dat andere <label> moet in de html gelijk na het element met id="toon-checked" staan (dat is de <input> met het aankruisvakje voor 'Alles'). Maar alleen als die <input> is aangevinkt.

`display: none;`

Verbergen.

Als het aankruisvakje voor 'Alles' is aangevinkt, worden de <input>'s en <label>'s voor de categorieën verborgen.

Elementen die met behulp van `display: none;` worden verborgen, worden genegeerd door schermlezers. Als een gebruiker van een schermlezer 'Alles' selecteert, wordt deze dus niet verblijd met het nutteloos voorlezen van alle mogelijke categorieën. Zodra 'Alles' niet meer is geselecteerd, worden <input>'s en <label>'s weer gewoon zichtbaar en daardoor herkend door de schermlezer.

Hetzelfde geldt voor mensen die de Tab-toets gebruiken om links, aankruisvakjes, tekstvelden, e.d. te bezoeken. Als 'Alles' is aangevinkt, worden de andere aankruisvakjes genegeerd, omdat deze met `display: none;` zijn verborgen. Zodra 'Alles' is uitgevinkt, zijn ze er weer. En worden dan op de normale manier door de Tab-toets bezocht, zodat ze ook door gebruikers van de Tab-toets kunnen worden aan- en uitgevinkt.

`#toon-achtergrond:checked ~ ul li[data-soort*=achtergrond]`

Voor deze elementen 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, li {text-indent: -0.7em; margin: 0.7em;} #toon-alles ~ ul li {display: none:} #toon-alles:checked ~ ul li {display: block;}

Deze selector oogt wat in gewikkeld, maar als hij in stukken wordt gehakt, wordt het 'n stuk overzichtelijker.

Bij #toon-alles ~ ul li en #toon-alles:checked ~ ul li is het in één keer tonen of verbergen van álle 's (en de daarin zittende links) geregeld: als 'Alles' is aangevinkt, worden alle 's getoond. Als 'Alles' is uitgevinkt, worden alle 's verborgen.

Als 'Alles' is uitgevinkt, zijn de aankruisvakjes voor de categorieën zichtbaar. Zodra één van die aankruisvakjes is aangevinkt, moet de daarbij horende categorie worden getoond. Dat regelt deze selector. In dit geval gaat het om de categorie 'achtergronden', maar het werkt voor alle categorieën op dezelfde manier.

`#toon-achtergrond:checked`: het element met id="toon-achtergrond'. Dit is de <input> die bij 'Achtergronden' hoort. Maar alleen als deze <input> is aangevinkt (`:checked`).

~: het hierachter staande element moet ergens in de html staan, na het voor de ~ staande element. Het hoeft er, anders dan bij de +, niet gelijk op te volgen, als het maar ergens na het eerste element in de html staat.

De enige voorwaarde is verder dat het voor en na de ~ staande element dezelfde ouder hebben. Dat is hier het geval: `#toon-achtergrond` voor de ~ en `ul` na de ~ hebben beide als ouder <main>.

`ul`: alle 's. Dat is er hier maar eentje. Deze `ul` mag niet ontbreken in de selector, omdat de voor en na de ~ staande elementen dezelfde ouder moeten hebben. `#toon-alles` en `ul` hebben beide als ouder <main>.
De hierna volgende `li` heeft echter als ouder niet <main>, maar <ul>. `li` kan hierdoor niet rechtstreeks worden gebruikt. Maar als we ouder `ul` van de `li` ertussen zetten, wordt wel aan de eis van dezelfde ouder voldaan. Want `ul` heeft wel dezelfde ouder als `#toon-alles`.

`li`: alle 's. Hoewel, álle, kijk eerst even gelijk hieronder...

`[data-soort*=achtergrond]`: als je inderdaad alle 's zichtbaar zou kunnen maken met deze selector voor 'Achtergronden', is de selector nutteloos. Want dat kan ook al met het aankruisvakje voor 'Alles'. Daarom mogen alleen bepaalde 's worden getoond: alleen de 's waarin een link naar een site met achtergronden zit. Daar zorgt dit stukje van de selector voor.

`[]`: de twee teksthaken omsluiten een 'attribuut-selector': een selector die kijkt of een bepaald (deel van een) attribuut aanwezig is (of juist niet aanwezig is, maar dat gebeurt hier niet).

`data-soort`: dit is de naam van het attribuut (of de class, id, e.d.) waarop wordt getest. In dit geval is het een data-attribuut: een zelfgemaakt attribuut. Als een attribuut met `data-` begint, wordt dit door de browser volledig genegeerd. Voor het deel achter het koppelteken is hier `soort` gebruikt, maar je kunt daarvoor elke willekeurig woord gebruiken.

`*=`: in dit geval staat er een isgelijkteken (sterretje komt zo) achter `data-soort`. In dat geval wordt niet alleen gekeken of een bepaald attribuut, class, id, e.d. aanwezig is, maar wordt ook gecontroleerd op de waarde van dat attribuut. Als alleen een isgelijkteken wordt gebruikt, moet de waarde precies gelijk zijn aan wat op het isgelijkteken volgt. Maar hier staat nog een sterretje voor het isgelijkteken. In dat geval hoeft de waarde niet precies hetzelfde te zijn als wat volgt op het isgelijkteken, maar moet dat één of meer keren aanwezig zijn. Er mogen ook nog andere dingen in de waarde staan, als wat na het isgelijkteken staat maar minimaal één keer aanwezig is.
Dat is in dit geval van belang. In deze selector wordt geselecteerd op de waarde 'achtergrond'. Bij <li data-soort="achtergrond"> is de waarde van `data-soort` precies hetzelfde als 'achtergrond'. Maar bij bijvoorbeeld <li data-soort="achtergrond foto"> is die waarde niet meer precies hetzelfde, omdat er nu ' foto' bij staat. De selector voor achtergrond zou daardoor niet werken.
Door het sterretje toe te voegen werkt de selector altijd, als er maar minimaal één keer 'achtergrond' ergens in `data-soort` staat.
(Er zijn nog meer van dit soort attribuut-selectors, zoals voor het alleen aan het begin of eind voorkomen van een bepaalde waarde. Die worden hier verder niet gebruikt.)

`achtergrond`: na het isgelijkteken of, in dit geval, het sterretje met isgelijkteken, staat de waarde, waarop wordt getest.

Dit deel van de selector bij elkaar:

`[data-soort*=achtergrond]`: 'achtergrond' moet ergens in de bij `data-soort` opgegeven waarde voorkomen.

Alles bij elkaar:

`#toon-achtergrond:checked ~ ul li[data-soort*=achtergrond]`: als het aankruisvakje voor 'Achtergronden' is aangevinkt, doe dan iets met de 's die in `data-soort` het woord 'achtergrond' hebben staan.

`display: block;`

Toon de 's.

Bij #toon-alles ~ ul li is opgegeven dat alle 's verborgen moeten worden, als het aankruisvakje voor 'Alles' niet is aangevinkt. Met de selector hier worden ' weer zichtbaar gemaakt, als ze in de waarde bij `data-soort` het woord 'achtergrond' hebben staan.

```
#toon-alfabet:checked ~ ul li[data-soort*=alfabet] {display:
    block;}
```

t/m

```
#toon-video:checked ~ ul li[data-soort*=video] {display: block;}
```

Deze selectors werken precies hetzelfde als die bij #toon-achtergrond:checked ~ ul li[data-soort*="achtergrond"], maar dan voor een andere categorie. De enige verschillen: het deel van de id achter 'toon-' heeft de naam van de categorie, en bij `data-soort` wordt op de aanwezigheid van die categorie gecontroleerd.

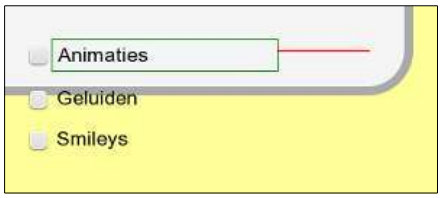
`hr`

Alle <hr>'s. Dat is er hier maar eentje: de horizontale streep tussen de aankruisvakjes en de eronder staande links.

In html 4.01 was een <hr> nog alleen een horizontale lijn, zonder enige verdere semantische betekenis. In html5 is het plotsklaps een thematische scheiding, die ook als zodanig door bijvoorbeeld schermlezers wordt gemeld. Waardoor in één klap miljoenen oudere sites plotseling zijn opgeleukt met thematische scheidingen, die nooit zo zijn bedoeld door de makers van die site. Ongetwijfeld tot dolle pret van gebruikers van schermlezers.

Waarom w3c dit in haar onmetelijke wijsheid heeft besloten, is mij een volslagen raadsel. Maar goed, het is niet anders. En in dit geval is een <hr> prima bruikbaar, want er is sprake van een thematische scheiding (tussen aankruisvakjes en links) én ik wil een simpele horizontale lijn.

`clear: both;`



Op de afbeelding is rondom de <label> met 'Animaties' een groene border gezet, zodat te zien is hoe breed deze <label> is. (Dat is breder dan de erin staande tekst.)

Daarachter staat een rode lijn: dat is de <hr>.

Normaal genomen vult een <hr> de volle breedte van de ouder van de <hr>. In dit geval is dat niet zo, omdat de <input>'s en <label>'s naar links zijn gefloat. De ruimte die achter de laatste <label> nog vrij is, wordt gevuld met een miezerig kort lijntje. Terwijl hier juist een <hr> in al zijn glorie zou moeten staan te schitteren.

Door het toevoegen van `clear: both;` wordt de <hr> niet naast, maar onder de gefloate elementen gezet. Je zou hier ook kunnen volstaan met `clear: left;`, omdat alle <input>'s en <label>'s naar links zijn gefloat. Maar het is een goede gewoonte om, als dat maar enigszins kan, altijd `clear: both;` te gebruiken. Als er ooit 'n `float: right;` zou worden ingevoegd voor de <hr>, zoek je je anders soms 'n ongeluk naar hoe het komt dat die <hr> opeens niet tot rechts doorloopt.

`margin-bottom: 30px;`

Ruimte tussen de onderkant van de <hr> en de eronder staande links.

`position: relative; top: 10px;`

Relatief positioneren, zodat de <hr> met `top` 10 px omlaag gezet kan worden voor wat afstand tussen de aankruisvakjes en de <hr>.

Een margin-top. werkt hier niet, omdat de gefloate <input>'s en <label>'s hier verstorend werken. Omdat deze elementen zijn gefloat, worden ze door een margin-top bij een eronder staand element genegeerd. Een marge aan de bovenkant wordt genomen vanaf het eerste niet-gefloate element boven de <hr>. Dat is de <p> met de tekst 'Hieronder slaat slechts...'. Daardoor zou je een hele grote marge moeten nemen. Bovendien zou die marge moeten veranderen afhankelijk van hoeveel aankruisvakjes er onder elkaar staan. Dit werkt dus niet.

Een padding werkt ook niet, omdat die bínnen de <hr> komt te staan en daardoor – afhankelijk van de browser – de lijn omtovert in een rechthoek van 10 px hoog.

`ul`

Alle 's. Dat is er hier maar eentje. Binnen deze <ul> staan de 's met de links.

`list-style-type: none;`

De gebruikelijke bolletjes e.d. binnen een <ul> zijn hier niet welkom.

`margin: 0; padding: 0;`

Omdat de standaardinstellingen van marge en padding in de verschillende browsers verschillen, kun je ze het beste zelf instellen. Nu zijn ze overal hetzelfde.

`a`

Alle <a>'s. Binnen elke <a> staat een link naar een andere site of pagina.

`word-wrap: break-word; overflow-wrap: break-word;`

Deze twee eigenschappen doen precies hetzelfde. Oorspronkelijk heette deze eigenschap `word-wrap`, tegenwoordig `overflow-wrap`. Omdat veel browsers nog alleen `word-wrap` herkennen, worden beide namen hier gebruikt.

Sommige links zijn heel erg lang. Vooral in smallere browservensters kan dat problemen opleveren, omdat ze soms niet worden afgebroken en dan een horizontale scrollbalk laten verschijnen.

Hiermee wordt afbreken toegestaan binnen de <a>, ook als dat normaal genomen niet zou gebeuren. Hierdoor krijg je soms wel afbrekingen op vreemde plaatsen, maar dat is beter dan een horizontale scrollbalk. De link blijft gewoon werken, ook als deze op vreemde plaatsen wordt afgebroken.

`.sub-link`

Voor deze elementen 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. a {word-wrap: break-word; overflow-wrap: break-word;}

De elementen met class="sub-link". Bij sommige links staat een extra link, bijvoorbeeld naar een gebruiksaanwijzing. Die links hebben deze class.

`font-size: 1em;`

Bij a wordt de lettergrootte van <a>'s in browservensters breder dan 760 px vergroot. Omdat deze sub-links in de normale tekst staan, is die grotere lettergrootte te groot voor de sub-links. Deze houden daarom hun normale grootte.

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.

`img`

Alle <img>'s. Dat zijn er hier negen: de Nederlandse vlaggetjes. Op de pagina met links zijn het er – op het moment van schrijven – 120.

`width: 16px; height: 10px;`

Normaal genomen worden breedte en hoogte van een <img> in de html opgegeven. Maar omdat het er hier zoveel zijn, wordt het de moeite waard dit in de css te doen. Dat spaart 119 keer de breedte en hoogte uit.

css voor vensters maximaal 499 px breed

`@media screen and (max-width: 499px)`

De css die hier tot nu toe staat, geldt voor alle browservensters.
De css die binnen deze media query staat, geldt alleen voor vensters die maximaal 499 px breed zijn. In deze smalle vensters komen slechts twee aankruisvakjes naast elkaar te staan.

`@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 @media-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.

`and`: er komt nog een voorwaarde, waaraan moet worden voldaan.

`(max-width: 499px)`: het venster mag maximaal 499 px breed zijn. Is het venster breder, dan wordt de css die binnen deze media-regel staat genegeerd.

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 (max-width: 499px) {
     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 `}`.

`input:nth-of-type(odd)`

Voor deze elementen 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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
input {float: left; margin-top: 5px;}
(css over tonen en verbergen van <input>'s is weggelaten, omdat dat binnen deze media query niet wordt gewijzigd en het er nogal ingewikkeld uitziet.)

`input:` alle <input>'s. Dat zijn hier de aankruisvakjes. Maar gelijk hieronder wordt een beperking aangebracht.

`nth-of-type`: alleen <input>'s die op een bepaalde plaats staan. Die plaats geef je aan tussen de haakjes. Dat kan heel simpel zijn, zoals alleen de tweede, maar het kan ook heel ingewikkeld worden. Hier blijft het redelijk simpel.

`(odd)`: het Engelse woord voor oneven. Omdat dit tussen de haakjes van `nth-of-type()` staat, geldt deze selector alleen voor de oneven <input>'s. Daarbij wordt de volgorde in de html aangehouden.

`clear: both;`

Bij input en label zijn alle <input>'s en <label>'s naar links gefloat. Daardoor komen er net zoveel naast elkaar te staan, tot de regel vol is. Dat levert de chaos op, die hiernaast op de afbeelding is te zien. Op sommige regels staan drie aankruisvakjes, op andere twee. En op sommige regels staat de tekst uit één <label>, op andere uit twee. Sommige aankruisvakjes zijn hun bijbehorende <label> kwijtgeraakt. Kortom: alom kommer en kwel.

Door elke oneven <input> met behulp van `clear: both;` op een nieuwe regel te zetten, wordt deze chaos voorkomen.

De eerste <input> is ook oneven, en die volgt niet op 'n gefloat element. Maar dat maakt niets uit: in dat geval heeft `clear` gewoon geen enkel effect.

Je zou hier ook kunnen volstaan met `clear: left;`, omdat alle <input>'s en <label>'s naar links zijn gefloat. Maar het is een goede gewoonte om, als dat maar enigszins kan, altijd `clear: both;` te gebruiken. Als er ooit 'n `float: right;` zou worden ingevoegd, zoek je je anders soms 'n ongeluk naar hoe het komt dat de <input>'s en <label>'s opeens weer op hol slaan.

`margin-left: 12px;`

Omdat de <input>'s niet binnen de <label>'s kunnen staan, moeten ze op eigen houtje worden neergezet. In combinatie met de bij label opgegeven breedte van 30% levert dit in smallere browservensters twee enigszins nette kolommen op. Meer hierover is te vinden bij En dan hadden we nog het tweede nadeel...

css voor vensters minimaal 500 px en maximaal 799 px breed

`@media screen and (min-width: 500px) and (max-width: 799px)`

De opbouw van deze regel staat beschreven bij @media screen and (max-width: 499px). Er zijn twee verschillen: het browservenster moet minimaal 500 px: breed zijn `(min-width: 500px)`. En de maximale breedte mag 799 px zijn: `(max-width: 799px)`. Er zijn hier dus twee voorwaarden: minstens 500 px breed, maar niet breder dan 799 px. In vensters van deze breedte komen drie aankruisvakjes naast elkaar te staan.

`input:nth-of-type(3n + 1)`

Voor deze elementen 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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
input {float: left; margin-top: 5px;}
(css over tonen en verbergen van <input>'s is weggelaten, omdat dat binnen deze media query niet wordt gewijzigd en het er nogal ingewikkeld uitziet.)

`input`: alle <input>'s. Dat zijn hier de aankruisvakjes. Maar gelijk hieronder wordt een beperking aangebracht.

`nth-of-type`: alleen <input>'s die op een bepaalde plaats staan. Die plaats geef je aan tussen de haakjes. Dat kan heel simpel zijn, zoals alleen de tweede, maar het kan ook heel ingewikkeld worden.

`(3n+1)`: De getallen '3' en '1' zijn gewoon getallen, daar is verder weinig geheimzinnigs aan.

De 'n' is een soort teller, die steeds met 1 wordt verhoogd. Bij de eerste keer is de 'n' 0, bij de tweede keer 0 + 1 = 1, bij de derde keer 1 + 1 = 2, enz.

`3n` betekent, net als in de wiskunde, 3 x de waarde van 'n'. De eerste keer is dat dus 3 x 0 = 0, de tweede keer 3 x 1 = 3, de derde keer 3 x 2 = 6, enz. Deze berekening

levert een reeks getallen op, beginnend met 0, die steeds 3 hoger worden: 0, 3, 6, 9, 12, …
Omdat er 13 <input>'s zijn, wordt de berekening 13 keer gemaakt. De dertiende en laatste berekening is 3 x 12 = 36. Maar alleen de uitkomsten tot en met 13 hebben nut, omdat er maar 13 <input>'s zijn.
(De laatste waarde van 'n' is geen 13 maar 12, omdat met 0 wordt begonnen. Voor mensen lastig, maar computers beginnen heel vaak te tellen met 0.)
De eerste keer is het resultaat 3 x 0 = 0. Daarachter staat nog + 1. De eerste keer is de volledige berekening dus 3 x 0 + 1 = 1. Oftewel: de eerste <input>.
De tweede keer is het resultaat 3 x 1 = 3. Daarachter staat nog + 1. De tweede keer is de volledige berekening dus 3 x 1 + 1 = 4. Oftewel: de vierde <input>.
De derde keer is het resultaat 3 x 2 = 6. Daarachter staat nog + 1. De derde keer is de volledige berekening dus 3 x 2 + 1 = 7. Oftewel: de zevende <input>.
Enz.
In dit geval gaat het dus om de eerste, vierde, zevende, tiende en dertiende <input>.
En als dat nou niet toevallig is: dat zijn precies de <inputs>'s die op een nieuwe regel moeten komen te staan! Komt dat even goed uit, nu kan ik die in één keer aanspreken.
(Als je 8000 <input>'s zou hebben, zou je met deze ene regel alle 8000 <inputs>'s op precies dezelfde manier kunnen afhandelen en alleen de eerste, vierde, zevende, tiende, dertiende, enz. <input> kunnen aanspreken. Er zijn met behulp van deze en soortgelijke pseudo-classes nog veel ingewikkelder en handiger selecties mogelijk.)

`clear: both;`

Hiervoor geldt precies hetzelfde als iets hierboven bij clear: both;. Het enige verschil: het gaat daar om de eerste, derde, vijfde enz. <input>, en hier om de eerste, vierde, zevende, enz. <input>.

`margin-left: 12px;`

Omdat de <input>'s niet binnen de <label>'s kunnen staan, moeten ze op eigen houtje worden neergezet. In combinatie met de hier gelijk onder bij `label` opgegeven breedte van 25% levert dit in deze browservensters drie enigszins nette kolommen op. Meer hierover is te vinden bij En dan hadden we nog het tweede nadeel...

`label`

Voor deze elementen 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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
label {width: 30%; float: left; line-height: 1.5em; margin-bottom: 10px;}
(css over tonen en verbergen van <label>'s is weggelaten, omdat dat binnen deze media query niet wordt gewijzigd en het er nogal ingewikkeld uitziet.)

Alle <label>'s. Elke <input> heeft een <label> met daarin de bijbehorende tekst voor de <input>. Met behulp van het for-attribuut wordt de <label> aan de bijbehorende <input> gekoppeld.

`width: 25%;`

Deze breedte is voor <label>'s het beste in deze browservensters. Meer hierover is te vinden bij En dan hadden we nog het tweede nadeel...
Een breedte in procenten is altijd ten opzichte van de ouder van het element. Dat is in dit geval <main>, een blok-element. Een blok-element wordt normaal genomen even breed als z'n ouder. De ouder van <main> is <body>, ook een blok-element.
Omdat <body> een blok-element is, wordt dit normaal genomen ook weer even breed als de ouder van <body>. Dat is <html>, ook een blok-element. <html> is het

buitenste element en wordt daardoor normaal genomen even breed als het venster van de browser.
Uiteindelijk wordt <label> dus nooit breder dan 25% van de breedte van het browservenster.

css voor vensters minimaal 760 px breed

`@media screen and (min-width: 760px)`

De opbouw van deze regel staat beschreven bij @media screen and (max-width: 499px). Er is één verschil: het browservenster moet minimaal 760 px: breed zijn `(min-width: 760px)` . In vensters van deze breedte wordt de lettergrootte iets verhoogd. Ook wordt de breedte van <main> beperkt om te lange – en daardoor slecht leesbare – regels te voorkomen. Ten slotte krijgt <main> nog een border.
(Normaal genomen zou deze @media-regel waarschijnlijk worden gecombineerd met die gelijk hieronder voor vensters van minimaal 800 px breed. Die breedte van 760 px komt nogal vaak voor op de site, en daar is dit voorbeeld ook van afkomstig. Daarom heb ik deze regel gewoon ook hier gebruikt, want dat scheelt nogal wat ombouwwerk. En vanwege het bij mij aanwezige luiheids-gen, kan ik er niets aan doen dat ik dat te veel moeite vind.)

`body`

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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
body {background: #ff9; color: black; font-family: Arial, Helvetica, sans-serif; margin: 0; padding: 0;}

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.

`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.

`main`

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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
main {-webkit-animation: bugfix-3 infinite 1s; background: #f5f5f5; color: black; display: block; margin: 20px auto; padding: 3px; border-top: black solid 1px;}

Alle <main>’s. Dat is er maar eentje. De belangrijkste inhoud van de pagina staat hierin. In dit voorbeeld zijn dat de links met de erboven staande teksten, aankruisvakjes, e.d.

`-webkit-animation: bugfix-30 infinite 1s;`

In oudere versies van Android browser zit een bug, waardoor de ’s niet worden verborgen of getoond bij uit- of aanvinken van een aankruisvakje. Met behulp van deze regel gebeurt dat toch. Meer hierover bij @-webkit-keyframes bugfix-30.
Het sleutelwoord `infinite` zorgt ervoor dat de animatie eindeloos doorgaat. `1s` geeft aan dat de animatie 1 seconde duurt (voordat deze weer opnieuw wordt afgespeeld.)

`width: 80%;`

Een breedte in procenten is altijd ten opzichte van de ouder van het element. Dat is in dit geval <body>. Omdat <body> een blok-element is, wordt dit normaal genomen

even breed als de ouder van <body>. Dat is <html>, ook een blok-element. <html> is het buitenste element en wordt daardoor normaal genomen even breed als het venster van de browser. Uiteindelijk wordt <main> hierdoor 80% van de breedte van het browservenster, ongeacht hoe breed dit venster is.
Bij main is eerder met `margin: 20px auto;` al gezorgd dat <main> altijd horizontaal in het midden van het browservenster staat.

`max-width: 1000px;`

Hier gelijk boven heeft <main> een breedte van 80% van het venster van de browser gekregen, zodat de regels niet te lang – en daardoor slecht leesbaar – worden. Maar in een venster van bijvoorbeeld 2500 px breed is dat nog steeds 2000 px, veel te lang. Daarom wordt de maximumbreedte hier beperkt tot 1000 px.
(Dat is nog steeds behoorlijk lang, maar omdat er veel korte regels aanwezig zijn en veel ruimte tussen de regels, denk ik dat het hier wel kan.)

`border: #aaa solid 8px;`

Aan alle kanten grijze border van 8 px breed.

`border-radius: 40px;`

Alle vier de hoeken rond. Omdat maar één waarde is opgegeven, worden alle hoeken horizontaal en verticaal 40 px breed en hoog.

`padding: 10px 30px 10px;`

Omdat voor links geen waarde is opgegeven, krijgt links automatisch dezelfde waarde als rechts. Hier staat dus eigenlijk `10px 30px 10px 30px` in de volgorde boven – rechts – onder – links. Aan alle kanten wat ruimte tussen de buitenkant van de <main> en de inhoud ervan.

`p, li`

Voor deze elementen 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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
p, li {text-indent: -0.7em; margin: 0.7em;}
(css over tonen en verbergen van 's is weggelaten, omdat dat binnen deze media query niet wordt gewijzigd en het er nogal ingewikkeld uitziet.)

Alle <p>'s en 's. Er zijn slechts twee <p>'s met tekst, die beide boven de aankruisvakjes staan. Elke link staat in een eigen <li>.

`text-indent: -2em;`

Eerste regel van elke <p> en <li> 2 em naar links zetten. Een uitgebreide uitleg staat bij p, li. Het enige verschil is dat het daar -0,7 em is en hier – 2 em, omdat er in bredere browservensters meer ruimte is.

`padding-left: 40px;`

Aan de linkerkant ruimte tussen tekst in en buitenkant van <p>'s en 's iets vergroten.

`h1`

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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
h1 {font-size: 1.5em; margin: 0;}

Alle <h1>'s. Dat is er maar eentje: de belangrijkste kop op de pagina.

`margin: 15px 0 20px;`

Omdat voor links geen waarde is opgegeven, krijgt links automatisch dezelfde waarde als rechts. Hier staat dus eigenlijk `15px 0 20px 0` in de volgorde boven – rechts – onder – links. In deze bredere browservensters is weer ruimte voor een marge boven en onder de <h1>.

`a`

> Voor deze elementen 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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
> a {word-wrap: break-word; overflow-wrap: break-word;}

Alle <a>'s. Binnen elke <a> staat een link naar een andere site (of pagina binnen deze site).

`font-size: 1.2em;`

De <a>'s staan op een aparte regel boven de bijbehorende tekst. Door de <a>'s iets groter te maken, ziet het er wat minder gedrongen uit allemaal, en vallen ze beter op. 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.

`line-height: 1.4em;`

Grotere regelhoogte, zodat het er minder gedrongen uitziet.
Als eenheid wordt de relatieve eenheid `em` gebruikt, omdat bij gebruik van een absolute eenheid zoals `px` niet alle browsers de regelhoogte kunnen veranderen. Zoomen kan wel altijd, ongeacht welke eenheid voor de regelhoogte wordt gebruikt.

css voor vensters minimaal 800 px breed

`@media screen and (min-width: 800px)`

De opbouw van deze regel staat beschreven bij @media screen and (max-width: 499px). Er is één verschil: het browservenster moet minimaal 800 px: breed zijn `(min-width: 800px)`. In vensters van deze breedte komen vier aankruisvakjes naast elkaar te staan.

`input:nth-of-type(4n + 1)`

> Voor deze elementen 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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
> input {float: left; margin-top: 5px;}
> (css over tonen en verbergen van <input>'s is weggelaten, omdat dat binnen deze media query niet wordt gewijzigd en het er nogal ingewikkeld uitziet.)

Hoe deze selector precies werkt is te vinden bij input:nth-of-type(3n + 1). Alleen moet hier 'n' niet worden vermenigvuldigd met 3, maar met 4. Hierdoor geldt deze selector voor de eerste, vijfde, negende, enz. <input>.

`clear: both;`

Hiervoor geldt precies hetzelfde als een eind hierboven bij clear: both;. Het enige verschil: het gaat daar om de eerste, derde, vijfde enz. <input>, en hier om de eerste, vijfde, negende, enz. <input>,

`margin-left: 20px;`

Omdat de <input>'s niet binnen de <label>'s kunnen staan, moeten ze op eigen houtje worden neergezet. In combinatie met de hier gelijk onder bij `label` opgegeven breedte van 20% levert dit in bredere browservensters vier enigszins nette kolommen op. Meer hierover is te vinden bij En dan hadden we nog het tweede nadeel...

`label`

> Voor deze elementen 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. (Alleen wat binnen deze media query geldig is, wordt binnen dit blokje herhaald.)
> label {width: 30%; float: left; line-height: 1.5em; margin-bottom: 10px;}
> (css over tonen en verbergen van <label>'s is weggelaten, omdat dat binnen deze media query niet wordt gewijzigd en het er nogal ingewikkeld uitziet.)

Alle <label>'s. Elke <input> heeft een <label> met daarin de bijbehorende tekst voor de <input>. Met behulp van het for-attribuut wordt de <label> aan de bijbehorende <input> gekoppeld.

`width: 20%;`

Deze breedte is voor <label>'s het beste in deze browservensters. Meer hierover is te vinden bij En dan hadden we nog het tweede nadeel...
Een breedte in procenten is altijd ten opzichte van de ouder van het element. Dat is in dit geval <main>, dat bij main een breedte van 80% van het venster van de browser heeft gekregen, en een maximumbreedte van 1000 px.
<label> wordt 20% van deze breedte van <main>. Als <main> de maximumbreedte van 1000 px heeft bereikt, wordt <label> 20% hiervan, dat is 200 px, breed.

Volledige code

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 `rood` 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">
     <meta name="description" content="Lijst met links die
          groepsgewijs via trefwoorden kunnen worden getoond of
          verborgen - voorbeeld. Gebruikt alleen html en css.">
     <title>Lijst met links die groepsgewijs via trefwoorden kunnen
          worden getoond of verborgen - voorbeeld</title>
     <link rel="stylesheet" href="113-css-dl/lijst-113-dl.css">
</head>
<body>
<main>
     <h1>Achtergronden, animaties, knoppen, pijlen, iconen, foto's,
          e.d.</h1>
     <div class="let-op">
     <p>De hieronder staande sites bevatten (voornamelijk) gratis
          materiaal. Maar dat geldt niet voor álles op élke site.
          Bovendien kunnen voorwaarden veranderen. Als je materiaal
```

```
        gebruikt in strijd met een licentie of zonder rekening te
        houden met copyright, kan dat tot <strong>schadeclaims
        van duizenden euro's</strong> leiden. Óók als je het
        gebruikte materiaal gelijk verwijdert, nadat het is
        ontdekt! Controleer áltijd of het materiaal echt volledig
        vrij en gratis gebruikt mag worden. Als je een
        zoekmachine of verzamelsite gebruikt, controleer dan
        altijd de licentie op de originele site.</p>
    </div>
    <p>Hieronder staat slechts een heel kleine selectie van wat er
        op internet is te vinden, vooral grotere sites. Toch zijn
        dit al zoveel sites dat het wat onoverzichtelijk wordt.
        Indelen in soorten materiaal gaat niet, omdat veel sites
        meerdere soorten aanbieden. Daarom kun je hieronder
        aanvinken, wat je zoekt. Alleen de sites met dat soort
        materiaal worden dan getoond. Als je 'Alles' uitvinkt,
        kun je kiezen wat je wilt zien.</p>
    <input type="checkbox" id="toon-alles" checked>
    <label for="toon-alles"> Alles</label>
    <input type="checkbox" id="toon-achtergrond">
    <label for="toon-achtergrond"> Achtergronden</label>
    <input type="checkbox" id="toon-alfabet">
    <label for="toon-alfabet"> Alfabetten</label>
    <input type="checkbox" id="toon-animatie">
    <label for="toon-animatie"> Animaties</label>
    <input type="checkbox" id="toon-clipart">
    <label for="toon-clipart"> Clipart</label>
    <input type="checkbox" id="toon-cursor">
    <label for="toon-cursor"> Cursors</label>
    <input type="checkbox" id="toon-foto">
    <label for="toon-foto"> Foto's</label>
    <input type="checkbox" id="toon-geluid">
    <label for="toon-geluid"> Geluiden</label>
    <input type="checkbox" id="toon-icon">
    <label for="toon-icon"> Iconen</label>
    <input type="checkbox" id="toon-knop">
    <label for="toon-knop"> Knoppen</label>
    <input type="checkbox" id="toon-lijn">
    <label for="toon-lijn"> Lijnen</label>
    <input type="checkbox" id="toon-smiley">
    <label for="toon-smiley"> Smileys</label>
    <input type="checkbox" id="toon-video">
    <label for="toon-video"> Video's</label>
    <hr>
    <ul>
```

```
            <li data-soort="achtergrond, alfabet, animatie, clipart,
                cursor, foto, lijn, smiley"><a href="113-files-
                dl/lijst-113-hulp-dl.html">animaatjes.nl</a><br><img
                src="113-pics/vlag.png" alt=""> Achtergronden,
                foto's, tekeningen, animaties, lijnen, alfabetten,
                cursors, smileys, clipart, enz.
                Nederlandstalig.</li>
            <li data-soort="animatie"><a href="113-files-dl/lijst-
            113-hulp-dl.html">animatedgif.net</a><br>Animaties.</li>
```
(…) t/m (...)
```
            <li data-soort="clipart"><a lang="en"
                href="113-files-dl/lijst-113-hulp-
                dl.html">widgetworx.com/spritelib</a><br>Zip met
                aantal sprites, toegespitst op spelletjes. Met
                behulp van sommige sprites kan beweging worden
                gesimuleerd.</li>
            <li data-soort="clipart"><a lang="en"
                href="113-files-dl/lijst-113-hulp-
                dl.html">wpclipart.com/browse</a><br>Publiek domein
                clipart.</li>
        </ul>
</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 `rood` gekleurd. Alle niet-essentiële code is `zwart`. (In de inhoudsopgave staat alles in een gewone letter vanwege de leesbaarheid.)

```
/* lijst-113.css */

@-webkit-keyframes bugfix-3 {from {padding-left: 3px;} to
    {padding-left: 3px;}}
@-webkit-keyframes bugfix-30 {from {padding-left: 30px;} to
    {padding-left: 30px;}}

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

```
main {
     -webkit-animation: bugfix-3 infinite 1s;
     background: #f5f5f5;
     color: black;
     display: block;
     margin: 20px auto;
     padding: 3px;
     border-top: black solid 1px;
}

h1 {
     font-size: 1.5em;
     margin: 0;
}

.let-op {
     background: white;
     color: black;
     margin: 5px 0;
     border: solid red 2px;
}

p, li {
     text-indent: -0.7em;
     margin: 0.7em;
}

input {
     float: left;
     margin-top: 5px;
}

label {
     width: 30%;
     float: left;
     line-height: 1.5em;
     margin-bottom: 10px;
}

#toon-alles ~ ul li {display: none;}

#toon-alles:checked ~ ul li {display: block;}

#toon-alles:checked ~ input, #toon-alles:checked + label ~ label
     {display: none;}
```

```
#toon-achtergrond:checked ~ ul li[data-soort*=achtergrond]
     {display: block;}

#toon-alfabet:checked ~ ul li[data-soort*=alfabet] {display:
     block;}

#toon-animatie:checked ~ ul li[data-soort*=animatie] {display:
     block;}

#toon-clipart:checked ~ ul li[data-soort*=clipart] {display:
     block;}

#toon-cursor:checked ~ ul li[data-soort*=cursor] {display: block;}

#toon-foto:checked ~ ul li[data-soort*=foto] {display: block;}

#toon-geluid:checked ~ ul li[data-soort*=geluid] {display: block;}

#toon-icon:checked ~ ul li[data-soort*=icon] {display: block;}

#toon-knop:checked ~ ul li[data-soort*=knop] {display: block;}

#toon-lijn:checked ~ ul li[data-soort*=lijn] {display: block;}

#toon-smiley:checked ~ ul li[data-soort*=smiley] {display: block;}

#toon-video:checked ~ ul li[data-soort*=video] {display: block;}

hr {
     clear: both;
     margin-bottom: 30px;
     position: relative;
     top: 10px;
}

ul {
     list-style-type: none;
     margin: 0;
     padding: 0;
}

a {
     word-wrap: break-word;
     overflow-wrap: break-word;
}
```

```
.sub-link {font-size: 1em;}

img {
     width: 16px;
     height: 10px;
}

@media screen and (max-width: 499px) {
     input:nth-of-type(odd) {
          clear: both;
          margin-left: 12px;
     }
}

@media screen and (min-width: 500px) and (max-width: 799px) {
     input:nth-of-type(3n + 1) {
          clear: both;
          margin-left: 12px;
     }
     label {width: 25%;}
}

@media screen and (min-width: 760px) {
     body {font-size: 110%;}
     main {
          -webkit-animation: bugfix-30 infinite 1s;
          width: 80%;
          max-width: 1000px;
          border: #aaa solid 8px;
          border-radius: 40px;
          padding: 10px 30px 10px;
     }
     p, li {
          text-indent: -2em;
          padding-left: 40px;
     }
     h1 {margin: 15px 0 20px;}
     a {
          font-size: 1.2em;
          line-height: 1.4em;
     }
}

@media screen and (min-width: 800px) {
     input:nth-of-type(4n + 1) {
```

```
        clear: both;
        margin-left: 20px;
    }
    label {width: 20%;}
}
```