

## Drie tellers en markeringen geven het aantal goed, foutief en nog niet ingevulde provinciehoofdsteden weer

Smaller venster

Vul de provinciehoofdsteden in:  
Goed: 2. Fout: 1. Nog niet ingevuld: 9.

Gr: groningen    Fr: Leeuwarden

Naar de uitslag

Breder venster

Vul de provinciehoofdsteden in:  
Goed: 2. Fout: 1. Nog niet ingevuld: 9.

Naar de uitslag

### BELANGRIJKE INFORMATIE

Alles op deze site kan vrij worden gebruikt, met drie beperkingen:

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

\* Deze uitleg wordt min of meer regelmatig bijgewerkt. Het is daarom niet toegestaan deze uitleg op welke manier dan ook te verspreiden, zonder daarbij duidelijk te vermelden dat de uitleg afkomstig is van [www.css-voorbeelden.nl](http://www.css-voorbeelden.nl) en dat daar altijd de nieuwste versie is te vinden. Dit is om te voorkomen dat er verouderde versies worden verspreid.

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

\* Het kan zijn dat materiaal is gebruikt dat van anderen afkomstig is. Dat materiaal kan onder een bepaalde licentie vallen, waardoor het mogelijk niet onbeperkt gebruikt mag worden. Als dat zo is, wordt dat vermeld onder [Inhoud van de download en licenties](#).

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

## Inhoudsopgave

[Korte omschrijving](#)

[Opmerkingen](#)

[Achterliggend idee](#)

[Semantische elementen en WAI-ARIA](#)

[Semantische elementen](#)

[WAI-ARIA-codes](#)

[Toetsenbordnavigatie en tabindex](#)

[Muis, toetsenbord, touchpad en aanraakscherm](#)

[:hover](#)

[:focus](#)

[:active](#)

[De code aanpassen aan je eigen ontwerp](#)

[Toegankelijkheid en zoekmachines](#)

[Getest in](#)

[Bekende problemen \(en oplossingen\)](#)

[Zonder css](#)

[Zonder afbeeldingen](#)

[Toetsenbordnavigatie](#)

[Tekstbrowsers](#)

[Schermlezers](#)

[Zoomen en andere lettergrootte](#)

[Overige problemen](#)

[Wijzigingen](#)

[Inhoud van de download en licenties](#)

Uitleg code:

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

[<!doctype html>](#)

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

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

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

[<link rel="stylesheet" href="009-css-dl/tekst-009-dl.css">](#)

[<h1 id="boven" tabindex="-1">Vul de provincie&shy;hoofdsteden in:</h1>](#)

[](#)

[<label for="gr">Gronin&shy;gen<span class="sr-fout">Deze stad is fout.</span><span class="sr-goed">Deze stad is goed.</span><span class="fout" aria-hidden="true">Fout!</span></label>](#)

[<input id="gr" type="text" required pattern="\\*\[Gg\]roningen \\*" placeholder="">](#)

[<span class="afk" aria-hidden="true">Gr:</span>](#)

[<div id="uitslag" tabindex="-1">](#)

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

css voor alle browsers en vensters:

```
main {counter-reset: goed fout leeg 12;}
#wrapper {position: relative;}
#kaart {max-width: 100%;}
label {font-size: 0.8em; text-align: center; position: absolute;}
input {width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em; position: absolute;}
# groningen label {width: 3em; bottom: 80%; left: 74%;}
# groningen input {margin-bottom: 10px; margin-left: calc(3.5em + 30px); bottom: calc(80% - 2em); left: calc(74% - 3.1em);}
# friesland label {width: 3em; right: 28%; bottom: 75%;}
t/m
# limburg input {top: calc(80% + 1.6em); left: calc(56% - 2.6em);}
span {display: none;}
.sr-fout, .sr-goed {position: absolute; top: -1000px; left: -20000px;}
.fout {background: rgb(255 255 255 / 0.7); color: black; width: 4ch; font-size: 0.7em; margin: 0 auto; border: red solid 1px; padding: 2px; position: absolute; top: 1em; right: 0; left: 0;}
.afk {box-sizing: border-box; height: 1.2rem; font-size: 0.8em; padding: 1px 20px 1px 2px; position: absolute; left: 0;}
div:nth-of-type(even) .afk {left: 54%;}
#wrapper div:has(:focus) label {background: rgb(255 255 255 / 0.8); color: black; font-weight: bold; z-index: 20;}
#wrapper input:focus {width: 9em; margin: 0; z-index: 20;}
#wrapper input:user-valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0; left: 1.8em; z-index: 10;}
#wrapper div:nth-of-type(even) input:user-valid {right: 0; left: auto;}
#wrapper div:has(:user-valid) label {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}
input:user-valid ~ .afk {display: block;}
# groningen input:user-valid, # groningen input:user-valid ~ .afk,
# friesland input:user-valid, # friesland input:user-valid ~ .afk {bottom: 0;}
# drenthe input:user-valid, # drenthe input:user-valid ~ .afk {top: calc(100% + 0.4em);}
# overijssel input:user-valid {bottom: -2em; left: calc(54% + 1.9em);}
# overijssel input:user-valid ~ .afk {bottom: -2em;}
t/m
# limburg input:user-valid ~ .afk {bottom: -9.9em;}
#wrapper div:has(:user-invalid):not(:has(:focus)) .fout {display: block;}
#wrapper div:has(:user-invalid) .sr-fout {display: block;}
#wrapper div:has(:user-valid) .sr-goed {display: block;}
#naar-uitslag, #naar-boven {display: none; position: relative; top: 12.4em;}
#naar-boven {display: block;}
#uitslag {display: none;}
```

css alleen voor browsers die :has() ondersteunen:

@supports selector(:has(\*))

h1 {padding-bottom: 2.2em;}

#uitslag {display: block; scroll-margin-top: 3.5em; position: absolute; top: 3.8em;}

#alles-goed {background: lightgreen; color: black; display: none; margin: 0;}

#naar-uitslag {display: block;}

#naar-boven {display: none;}

css voor alle browsers en vensters:

#goed, #fout, #leeg {display: inline;}

input:user-invalid {counter-increment: fout leeg -1;}

input:user-valid {counter-increment: goed leeg -1;}

#goed::after {content: counter(goed);}

css alleen voor browsers die :user-valid niet ondersteunen:

@supports not selector(:user-valid)

#wrapper input:valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0; left: 1.8em; z-index: 10;}

#wrapper div:nth-of-type(even) input:valid {right: 0; left: auto;}

#wrapper div:has(:valid) label {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}

input:valid ~ .afk {display: block;}

# groningen input:valid, # groningen input:valid ~ .afk, # friesland input:valid, # friesland input:valid ~ .afk {bottom: 0;}

#drenthe input:valid, #drenthe input:valid ~ .afk {top: calc(100% + 0.4em);}

#overijssel input:valid {bottom: -2em; left: calc(54% + 1.9em);}

#overijssel input:valid ~ .afk {bottom: -2em;}

#flevoland input:valid, #flevoland input:valid ~ .afk {bottom: -3.8em;}

t/m

# limburg input:valid ~ .afk {bottom: -9.9em;}

input:valid {counter-increment: goed leeg -1;}

#fout, #leeg {display: none;}

#fout-of-leeg {display: inline;}

#fout-of-leeg::after {content: counter(leeg);}

css voor alle browsers en vensters:

#fout::after {content: counter(fout);}

#leeg::after {content: counter(leeg);}

css voor vensters minimaal 500 px breed:

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

# groningen label {bottom: 84%; left: 76%;}

# groningen input {bottom: calc(84% - 2em); left: calc(76% - 3.1em);}

# friesland label {right: 32%; bottom: 79%;}

t/m

# limburg input {top: calc(83% + 1.6em); left: calc(60% - 2.8em);}

#wrapper div:has(:focus) label {font-size: 1em; font-weight: normal;}

css voor vensters minimaal 760 px breed:

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

#wrapper div {display: flex; flex-wrap: wrap; align-content: center; justify-content: center; position: absolute;}

# groningen {width: 20%; height: 20%; top: 4%; left: 72%;}

```

#friesland {width: 19%; height: 22%; top: 9%; left: 53%;}
t/m
#limburg {width: 11%; height: 32%; top: 62%; left: 58%;}
#wrapper label {width: auto; font-size: 1em; position: static;}
#wrapper div:has(:focus) label {background: transparent; font-size:
1.5em;}
.fout {position: static;}
#wrapper input, #wrapper input:valid {background: rgb(255 255 255 /
0.8); color: black; width: 9em; height: auto; margin: 0;
position: static;}
#wrapper .afk {display: none;}
#naar-uitslag, #naar-boven {top: 6px;}
css voor alle browsers en vensters:
#wrapper:not(:has(:placeholder-shown)):not(:has(:invalid)) + #uitslag
#met-fout {display: none;}
#wrapper:not(:has(:placeholder-shown)):not(:has(:invalid)) + #uitslag
#alles-goed {display: block;}

```

Volledige code:

[HTML](#)  
[CSS](#)

### Korte omschrijving

De twaalf provinciehoofdsteden moeten worden ingevuld. Tellers houden bij hoeveel er goed, fout of nog niet zijn ingevuld. Fout ingevulde krijgen een melding, goed ingevulde kleuren groen.

### Opmerkingen

De plattegrond is afkomstig van

[https://commons.wikimedia.org/wiki/File:Netherlands\\_location\\_map.svg](https://commons.wikimedia.org/wiki/File:Netherlands_location_map.svg) en valt onder de [Creative Commons Attribution-Share Alike 3.0 Unported](#) licentie.

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

Dit voorbeeld is gemaakt op een systeem met Linux ([KDE neon](#)). Daarbij is vooral gebruik gemaakt van [Visual Studio Code](#), [GIMP](#) en [Firefox](#) met extensies. De pdf-bestanden zijn gemaakt met [LibreOffice](#).

Vragen of opmerkingen? Fout gevonden? Ga naar het [forum](#).

### Achterliggend idee

Op een kaart van Nederland met de provinciegrenzen staat in elke provincie de naam van de provincie. Onder elke naam staat een tekstveld, waarin de naam van de hoofdstad van de provincie moet worden ingevuld. Tekstvelden worden meestal in een formulier (<form>) gebruikt, maar dat hoeft niet.

In smallere browservensters staat het tekstveld soms niet onder, maar iets naast de naam van de provincie. De namen en tekstvelden zijn daarin zo geplaatst, dat de letters tot 200% vergroot kunnen worden, zonder dat provincienamen en tekstvelden elkaar te veel overlappen.

In bredere browservensters is genoeg ruimte om de volledige naam van de hoofdstad weer te kunnen geven. In smallere vensters is die ruimte er niet, daarin is het tekstveld veel smaller dan de naam van de hoofdstad. Om die naam toch in te kunnen vullen, wordt het tekstveld

bij [focus](#) breder gemaakt. Maar het venster wordt daarmee helaas niet breder, dus er is nog steeds onvoldoende ruimte om de volledige naam van de hoofdstad weer te geven. Zeker als alle twaalf de namen zouden worden getoond: ze komen dan voor een groot deel over elkaar heen te staan.

Daarom wordt in smallere browservensters, als een hoofdstad goed is ingevuld, deze onder de kaart van Nederland neergezet. Voor de hoofdstad komen twee letters te staan, die de bijbehorende provincie aangeven.

De bedoeling is dat de juiste hoofdsteden worden ingevuld. Daarbij mag aan het begin een hoofd- of kleine letter worden gebruikt. Ook mogen er één of meer spaties voor of na de hoofdstad staan. Den Haag en Den Bosch mogen ook als 's-Gravenhage of 's-Hertogenbosch worden ingevuld. Met behulp van het `pattern`-attribuut wordt gecontroleerd op de hoofdstad goed is ingevuld.

Als een tekstveld binnen een formulier wordt gebruikt, wordt de inhoud normaal genomen gecontroleerd, als op de knop voor verzenden wordt gedrukt. Hier staan de tekstvelden echter niet binnen een `<form>`, dus die controle werkt niet. Je zou ook met behulp van JavaScript kunnen controleren op juistheid, maar de lol is nou juist om zo weinig mogelijk JavaScript te gebruiken.

Daarom wordt hier gecontroleerd met behulp van `:user-valid` en `:user-invalid`. Deze twee pseudo-classes maken het mogelijk gelijk na het invoeren van een hoofdstad te controleren op de juistheid ervan. Als de naam juist is, wordt de achtergrond van de bijbehorende provincienaam groen (en in smallere browservensters wordt de naam van de hoofdstad onder de kaart van Nederland neergezet). Als de naam niet juist is, verschijnt bij de provincienaam de melding 'Fout!'. (En in smallere vensters wordt het tekstveld weer smaller, omdat het anders over andere elementen heen zou komen te staan.)

`:user-valid` en `:user-invalid` voeren hun controle pas uit, nadat er iets is gewijzigd in het tekstveld. Voor oudere browsers die `:user-valid` en `:user-invalid` niet ondersteunen, worden `:valid` en `:invalid` gebruikt. Deze twee pseudo-classes voeren hun controle echter gelijk bij het openen van de pagina al uit. Hierdoor zou, terwijl er nog helemaal niets is ingevuld, twaalf keer de melding 'Fout!' verschijnen. Mensen hebben van minder faalangst opgelopen. Daarom wordt bij die oudere browsers de melding 'Fout!' weggelaten. Alleen als iets goed is ingevuld, wordt dat gemeld. Er is nog een andere bijwerking: er wordt geen onderscheid gemaakt tussen nog lege en verkeerd ingevulde velden. Deze worden in de uitslag gecombineerd weergegeven.

Er bestaat – zonder JavaScript – geen mogelijkheid om te kijken of een tekstveld leeg is. Soms wordt gedacht dat je hier `:empty` voor kunt gebruiken, maar deze pseudo-class kijkt, of het element geen kinderen heeft. Je kunt hier echter wel een trucje voor gebruiken. Met behulp van het `placeholder`-attribuut kun je in een veld een hint geven, wat er moet worden ingevuld. Bijvoorbeeld 'Adres' in een veld voor het adres. Deze hint verdwijnt, zodra er iets wordt ingevuld.

Soms wordt zo'n hint gebruikt als vervanging voor een `<label>`. Dat is een onjuist gebruik, want zodra er iets is ingevuld, verdwijnt de hint. Terwijl een `<label>` gewoon blijft staan. Daarnaast lezen niet alle [schermlezers](#) de hint voor, waardoor soms volkomen onduidelijk is, wat er moet worden ingevuld.

Hier wordt bij elk tekstveld `placeholder=""` opgegeven. De hint is volkomen leeg, er zit zelfs geen spatie in. Daardoor zie je de hint niet en wordt deze niet voorgelezen door schermlezers, want er is niets om te tonen of om voor te lezen. Maar evenzogoed verdwijnt de hint, zodra er iets is ingevuld. En daar kun je met de pseudo-class `:placeholder-shown` op controleren: zodra de hint is verdwenen, dus zodra er iets is ingevuld, krijgt het tekstveld deze pseudo-class. Dat er eigenlijk helemaal niets is verdwenen, maakt voor de

computer niets uit. Er was een hint en die is nu weg. Of die hint nu een volledig onzichtbare geest was of een betonnen muur, maakt verder niets uit. De lenigheid van geest die ook iemand als minister Faber uitbundig toont: het is gitzwart of helderwit en alles ertussen bestaat niet. Met als verschil dat dat bij Faber tot weerzinwekkende onmenselijkheid leidt en hier juist nuttig is.

Drie afzonderlijke tellers houden het aantal goed ingevulde hoofdsteden, het aantal verkeerd ingevulde hoofdsteden en het aantal nog niet ingevulde tekstvelden bij. Niet in alle browsers kunnen alle drie deze tellers worden weergegeven. Dat is afhankelijk van de ondersteuning van `:has()`, `:user-valid` en `:user-invalid`.

De drie tellers worden bij `<main>` aangemaakt, zodat ze binnen de hele pagina zijn te gebruiken. De tellers voor goed en fout krijgen de standaardwaarde '0', want er zijn nog geen goed of fout ingevulde tekstvelden. De teller voor leeg krijgt de waarde '12', want alle twaalf tekstvelden zijn nog leeg.

Zodra een hoofdstad goed is ingevuld – waarbij 'goed' afhankelijk is van het voor die hoofdstad in `pattern` opgegeven patroon –, wordt de teller voor goed met behulp van `:user-valid` met 1 verhoogd en de teller voor leeg met 1 verlaagd.

Zodra iets is ingevuld, maar niet voldoet aan het voor die hoofdstad opgegeven patroon in `pattern`, wordt de teller voor fout met behulp van `:user-invalid` met 1 verhoogd en de teller voor leeg met 1 verlaagd. Zelfs als alleen maar een spatie is ingevuld, is er iets ingevuld en geldt ook die spatie als een foutieve invoer.

Deze drie tellers worden boven de kaart van Nederland weergegeven.

Alle twaalf de tekstvelden staan in een `div#wrapper`. Met behulp van `:has()`, `:placeholder-shown`, `:user-invalid` en `:invalid` wordt gekeken, of er in `div#wrapper` alleen maar juiste hoofdsteden zijn ingevuld. Als alle twaalf hoofdsteden goed zijn ingevuld, worden geen tellers meer weergegeven, maar alleen een melding dat alles goed is.

Bovenstaande verhaal geldt voor browsers die `:user-valid` en `:user-invalid` ondersteunen.

Voor browsers die `:user-valid` en `:user-invalid` niet ondersteunen, worden de pseudo-classes `:valid` en `:invalid` gebruikt. De CSS is voor beide variaties grotendeels hetzelfde, maar moet wel apart worden opgegeven.

Bij gebruik van `:invalid` wordt bij openen van de pagina gelijk een validatie uitgevoerd, waardoor al gelijk bij het openen van de pagina twaalf foutmeldingen zouden verschijnen. Daarom wordt voor deze browsers geen foutmelding gegeven. (Bij juist invullen van een hoofdstad verandert de achtergrondkleur van de bijbehorende provincie wel.)

De teller voor het aantal juist ingevulde hoofdsteden werkt gewoon. Met behulp van `:placeholder-shown` kan worden gecontroleerd, of er iets is ingevuld. Hierdoor is dus het totale aantal ingevulde én het aantal goed ingevulde tekstvelden bekend. Daardoor kan voor deze browser het aantal goed ingevulde hoofdsteden worden getoond, en het gecombineerde aantal lege of verkeerd ingevulde tekstvelden.

Een overzicht van browsers die `:user-valid` en `:user-invalid` niet ondersteunen is te vinden bij [Overige problemen](#).

Als een browser `:has()` niet ondersteunt, vindt geen enkele controle plaats op de juistheid van de ingevulde hoofdsteden. Ook zijn er geen tellers aanwezig. Je kunt nog steeds de hoofdsteden invullen, maar als je in plaats daarvan de namen van alle gebeurtenissen wilt invullen, waarbij Mark Rutte plotseling last van z'n geheugen had, kan dat ook: de browser zal niet klagen.

Een overzicht van browsers die :has () niet ondersteunen is te vinden bij [Overige problemen](#).

De drie tellers moeten onder de twaalf tekstvelden staan, omdat ze anders niet worden bijgewerkt. Daarom zijn ze in de html onder div#wrapper neergezet. Dat is echter een behoorlijk onlogische plaats, omdat ze in iets lagere browservensters nu niet zijn te zien. Daarom worden ze met behulp van css boven de kaart van Nederland neergezet. Onder de kaart staat een link naar de tellers, zodat ook in lagere vensters de tellers snel zijn te vinden.

Voor [schermlezers](#) is dit allemaal niet ideaal. Een schermlezer heeft niets aan een groene achtergrond als teken dat iets goed is ingevuld. En ook de melding 'Fout!', die verschijnt als iets fout is ingevuld, is nutteloos, want die verschijnt pas nadat het tekstveld is verlaten.

Voor schermlezers bestaan speciale codes. Een van deze codes is aria-live, waarmee je op een heel specifiek moment tekst kunt laten voorlezen. Dit blijkt hier echter niet goed te werken en alleen maar voor verwarring te zorgen. Meer daarover is te vinden bij [Bekende problemen \(en oplossingen\) → Schermlezers](#).

Voor schermlezers wordt de melding 'Fout!' met behulp van [aria-hidden](#) verborgen. In plaats daarvan worden de teksten 'Deze stad is goed' en 'Deze stad is fout' gebruikt. Deze teksten staan onzichtbaar buiten het scherm, maar worden wel gewoon voorgelezen.

Schermlezers krijgen nu nog steeds geen directe goed- of foutmelding. Maar als de tellers fouten aangeven, kunnen ze nogmaals door de tekstvelden lopen, waarbij de teksten 'Deze stad is goed' of 'Deze stad is fout' op het juiste moment worden voorgelezen.

## Semantische elementen en WAI-ARIA

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

### Semantische elementen

De meeste elementen die in html worden gebruikt, hebben een semantische betekenis. Dat wil zeggen dat je aan de gebruikte tag al (enigszins) kunt zien, wat voor soort inhoud er in het element staat. In een <h1> staat een belangrijke kop. In een <h2> staat een iets minder belangrijke kop. In een <p> staat een alinea. In een <table> staat een tabel (en geen lay-out, als het goed is!). Enz.

Door het op de goede manier gebruiken van semantische elementen, kunnen zoekmachines, schermlezers, enz. de structuur van een pagina begrijpen. De spider van een zoekmachine is redelijk te vergelijken met een blinde. Het is dus ook in je eigen belang om semantische elementen zo goed mogelijk te gebruiken. Een site die toegankelijk is voor mensen met een handicap, is in de regel ook goed te verwerken door een zoekmachine en maakt dus een grotere kans gevonden en bezocht te worden.

Als het goed is, wordt het uiterlijk van de pagina bepaald met behulp van css. Het uiterlijk staat hierdoor (vrijwel) los van de semantische inhoud van de pagina. Met behulp van css kun je een <h1> heel klein weergeven en een <h6> heel groot, terwijl schermlezers, zoekmachines, e.d. nog steeds weten dat de <h1> een belangrijke kop is.

Slechts enkele elementen, zoals <div> en <span>, hebben geen semantische betekenis. Daardoor zijn deze elementen uitstekend geschikt om met behulp van css het uiterlijk van de pagina aan te passen: de semantische betekenis verandert niet, maar het uiterlijk wel. Voor een schermlezer of zoekmachine verandert er (vrijwel) niets, voor de gemiddelde bezoeker krijgt het door de css een heel ander uiterlijk. (De derde laag, naast html voor de inhoud en css voor het uiterlijk, is JavaScript. Die zorgt voor de interactie tussen site en bezoeker. De min of meer strikte scheiding tussen css en html aan de ene kant en JavaScript aan de andere kant is met de komst

van css3 en html5 veel vager geworden. Je kunt nu bijvoorbeeld ook met css dingen langzaam verplaatsen en met html de invoer in formulieren controleren.)

Html5 heeft een aantal nieuwe elementen, die speciaal zijn bedoeld om de opbouw van een pagina aan te geven. Deze elementen gedragen zich als een gewone <div>, maar dan een <div> met een semantische betekenis. Hierdoor kunnen schermlezers, zoekmachines, e.d. beter zien, hoe de pagina is samengesteld. De meeste schermlezers kunnen dit soort elementen ook gebruiken om snel door de pagina te navigeren. In dit voorbeeld wordt hiervan alleen <main> gebruikt.

<MAIN>

Hierbinnen staat de belangrijkste inhoud van de pagina (in dit voorbeeld is dat de hele pagina).

## WAI-ARIA-codes

WAI-ARIA wordt vaak ingekort tot ARIA. Voluit betekent het Web Accessibility Initiative – Accessible Rich Internet Applications.

Er wordt in dit voorbeeld één WAI-ARIA-code gebruikt: `aria-hidden`.

### ARIA-HIDDEN

Met behulp van `aria-hidden="true"` kan een deel van de code worden verborgen voor schermlezers e.d., zodat dit niet wordt voorgelezen. Op de normale weergave op het scherm heeft dit verder geen enkele invloed.

Elke provincienaam zit in een <label>. In die <label> zit o.a. ook een <span> met daarin het woordje 'Fout!'. Dat woordje wordt pas zichtbaar, als een tekstveld wordt verlaten, waarin de hoofdstad niet goed is ingevuld. Als je 'Fout!' kunt zien, is de betekenis daarvan glashelder.

Met behulp van de WAI-ARIA-code `aria-live` moet je zo'n melding kunnen voorlezen, op het moment dat deze verschijnt. Dan zou het ook voor schermlezers duidelijk zijn: gelijk na de foutieve invoer zouden ze dan 'Fout' horen. Maar `aria-live` blijkt hier in de meeste schermlezers niet goed te werken en alleen maar voor verwarring te zorgen. Wat er precies misgaat is te vinden bij [Bekende problemen \(en oplossingen\) → Schermlezers](#).

Omdat `aria-live` hier onbruikbaar blijkt te zijn, wordt de <span> met 'Fout!' pas voorgelezen, als het tekstveld nogmaals wordt bezocht. En dan is het heel verwarrend, omdat de volgorde van voorlezen voor bijvoorbeeld Groningen dan is: 'Groningen Fout! Groningen'. 'Groningen' zit in de <label> bij het tekstveld, de <span> met 'Fout!' zit daarachter in de <label>, en 'Groningen' zit in het tekstveld en hoort 'Groningen' te zijn.

Daarom wordt de <span> met 'Fout!' voor schermlezers met `aria-hidden="true"` verborgen:

```
<span class="fout"
      aria-hidden="true">Fout!</span>
```

Voor schermlezers wordt een eigen tekst gebruikt, waarover meer is te vinden bij [<label for="gr">Gronin&shy;gen<span class="sr-fout">Deze...](#)

(Omdat een juist ingevulde hoofdstad alleen met een kleur en een kadertje wordt aangegeven, hebben schermlezers ook hier niets aan. Ook hierover is meer te vinden bij de link gelijk hierboven.)

In smallere browservensters worden goed ingevulde provinciehoofdsteden vanwege ruimtegebrek onder de kaart van Nederland gezet. De bijbehorende provincie wordt met twee letters aangegeven. Ook daar hebben schermlezers niets aan, omdat in de html die twee letters na de naam van de provincie en de

hoofdstad komen. Bij Groningen bijvoorbeeld zou dit worden voorgelezen als 'Groningen Groningen gr'. Dit geeft de houding van 'Den Haag' tegenover Groningen aardig weer, maar schermlezers hebben er niets aan. Die 'gr' is voor schermlezers ook helemaal niet nodig, want bij het voorlezen wordt gewoon de volgorde van de html aangehouden, en die is bij bijvoorbeeld Groningen nog steeds <label> (met provincie 'Groningen') en <input type="text"> met (de hoofdstad 'Groningen'). Daarom worden die tweeletterige afkortingen voor schermlezers verborgen:

```
<span class="afk" aria-hidden="true">Gr:</span>
```

## Toetsenbordnavigatie en tabindex

Op een desktopcomputer gebruiken de meeste mensen een muis om over een pagina te navigeren, links te volgen, e.d. Soms gebruiken mensen hier echter een toetsenbord voor, omdat ze geen muis kunnen of willen gebruiken. Navigeren, links volgen, e.d. gebeurt dan met behulp van de Tab-toets, Enter, Spatiebalk, pijltjestoetsen, e.d. (Overigens wordt ook bij aanraakschermen soms een toetsenbord en/of muis gebruikt.)

Als je alleen simpele html zou gebruiken, werkt deze toetsenbordnavigatie fantastisch. Maar soms kan een bepaalde constructie in de html of in de css de werking ervan verstoren. Bij het maken van een website moet je er vooral op letten dat alles bereikbaar is met de Tab-toets. Voor andere toetsen zijn dan meestal geen aanpassingen nodig.

Links, invoervelden in formulieren, e.d. kunnen met behulp van de Tab-toets één voor één worden bezocht, in de volgorde waarin ze in de html voorkomen. Shift+Tab-toets keert de volgorde van de Tab-toets om. Dit is een belangrijk hulpmiddel voor mensen die om een of andere reden de muis niet kunnen of willen gebruiken. (En het is vaak ook veel sneller dan de muis, vooral in formulieren.)

In sommige browsers en/of besturingssystemen is dit vreemd genoeg standaard uitgeschakeld en is een zoektocht in de instellingen nodig om dit aan te zetten. Maar gebruikers van de Tab-toets zullen dit al hebben gedaan.

Als je met behulp van de Tab-toets een element hebt bereikt, heeft dit [focus](#): als het een link is en je drukt op Enter, wordt de link gevolgd. Bij een tekstveld kun je tekst gaan invoeren. Enz.

De Tab-toets volgt normaal genomen de volgorde van de elementen in de html. Het maakt niet uit, in welke volgorde ze op het scherm staan. Als je met behulp van css de elementen van plaats verwisselt op het scherm, wordt toch gewoon de volgorde in de html gevolgd.

De volgorde van de Tab-toets kan worden veranderd met behulp van het `tabindex`-attribuut: <div `tabindex="3"`>. Deze <div> zal nu als derde worden bezocht, ook al krijgt een simpele <div> normaal genomen nooit bezoek van de Tab-toets.

Normaal genomen is het gebruik van een `tabindex` niet nodig. Het is zeker niet bedoeld om de bezoeker als een kangoeroe op een hindernisbaan van onder via links over rechts naar boven te laten springen. Maar soms kan het handig zijn voor kleinere correcties, als de normale volgorde in de html niet optimaal is. Of om een element bereikbaar te maken voor de Tab-toets, zoals de hierboven genoemde <div>.

Schermlezers blijven altijd de volgorde van de html volgen, dus als de `tabindex` sterk afwijkt van de volgorde in de html, kan dat behoorlijk verwarrend zijn.

De `tabindex` kan drie verschillende waarden hebben: -1, 0 of een positief getal.

In principe is de volgorde bij gebruik van de Tab-toets als volgt: eerst worden alle positieve getallen in volgorde afgewerkt. Als twee `tabindex`en dezelfde waarde hebben, wordt de volgorde in de html aangehouden. Daarna worden elementen die automatisch al de [focus](#)

kunnen krijgen (zoals een link en een knop) en elementen met een `tabindex="0"` afgewerkt in de volgorde, waarin ze in de html staan.

#### **TABINDEX="-1"**

Een negatieve waarde van -1 zorgt ervoor dat een element, dat normaal genomen door het gebruik van de Tab-toets de [focus](#) kan krijgen, volledig wordt genegeerd door de Tab-toets. Zelfs een link met een negatieve `tabindex` wordt volledig genegeerd.

Ook kan aan een element met een negatieve `tabindex` dat normaal genomen geen focus kan krijgen, toch de focus worden gegeven met behulp van JavaScript, een klik of een aanraking. Waarbij gebruikers van de Tab-toets daar geen last van hebben, omdat `tabindex="-1"` wordt genegeerd door de Tab-toets.

Hier wordt om nog een andere reden op twee plaatsen `tabindex="-1"` gebruikt:

```
<h1 id="boven" tabindex="-1">Vul de  
    provincie&shy;hoofdsteden in:</h1>  
<div id="uitslag" tabindex="-1">
```

Afhankelijk van wat de browser ondersteunt, staat onder de kaart van Nederland een link naar een van deze twee ankers. Deze links blijken in [schermlezers](#) NVDA en VoiceOver niet te werken, tenzij je aan de `<h1>` en de `<div>` een negatieve `tabindex` geeft. Waarom die negatieve `tabindex` het wel laat werken is onduidelijk, maar dat is verder niet zo belangrijk: de links werken nu.

#### **TABINDEX="0"**

Als je `tabindex="0"` gebruikt, kan een element [focus](#) krijgen met behulp van de Tab-toets, door een aanraking of door een klik. Ook als dat element normaal genomen geen focus kan krijgen.

`tabindex="0"` wordt hier niet gebruikt.

#### **TABINDEX="..."**

Op de plaats van de puntjes moet een positief getal worden ingevuld: het volgnummer. Een element met een positieve `tabindex` wordt altijd bezocht bij gebruik van de Tab-toets, ook als dit element normaal genomen zou worden genegeerd. Elementen met een `tabindex` met een positief volgnummer worden altijd als eerste bezocht, voor elementen als een link of tekstveld zonder `tabindex`, en ook voor elementen met `tabindex="0"`.

Een positieve `tabindex` wordt hier niet gebruikt.

### **Muis, toetsenbord, touchpad en aanraakscherm**

Vroeger, toen het leven nog mooi was en alles beter, waren er alleen monitors. Omdat kinderen daar niet af konden blijven met hun tengels, besloten fabrikanten dan maar aanraakschermen te gaan maken, omdat je die mag aanraken. Het bleek makkelijker te zijn om volwassenen ook te leren hoe je 'n scherm vies maakt, dan om kinderen te leren met hun vingers van de monitor af te blijven.

Zo ontstonden aanraakschermen en kreeg het begrip ellende een geheel nieuwe lading. In de perfecte wereld van vroeger, waarin alleen desktops bestonden, werkten dingen als hoveren, klikken en slepen gewoon. Zonder dat je eerst 'n cursus hogere magie op Zweinstein hoefde te volgen. Zelfs in JavaScript was het nog wel te behappen, ook voor mensen die toevallig niet Einstein heten.

Op dit moment kun je computerschermen ruwweg in twee soorten indelen: schermen die worden aangeraakt, en schermen die worden bediend met hulpmiddelen als een toetsenbord,

muis of touchpad. Omdat ook computerschermen zich kennelijk vermengen, bestaan er inmiddels ook schermen die zowel van aanraken als van muizen houden. Hieronder staat een lijstje met dingen die zijn aangepast voor de verschillende soorten schermen, zodat dit voorbeeld overal werkt. Een touchpad werkt ongeveer hetzelfde als een muis. Als hieronder iets over een muis staat, geldt dat ook voor een touchpad.

#### **:HOVER**

Omdat `:hover` mogelijk niet werkt, als css uitstaat, ontbreekt of onvolledig is geïmplementeerd, moet je nooit belangrijke informatie alleen met behulp van `:hover` tonen.

Je hoovert over een element, als je met behulp van muis of touchpad de cursor boven dat element brengt. Hoveren kan over alle elementen. Het wordt veel gebruikt om iets van uiterlijk te laten veranderen, een pop-up te laten verschijnen, e.d.

Op een aanraakscherm wordt hoveren vaak hetzelfde afgehandeld als een aanraking. Bij gebruik van een muis is er een verschil tussen hoveren en klikken, maar op een aanraakscherm is dat verschil er niet: je raakt een aanraakscherm aan of niet. Dat levert vooral soms problemen op, als bij een element `:hover` én klikken worden gebruikt, zoals bij een link die bij hoveren erover verkleurt. Omdat deze combinatie niet wordt gebruikt in dit voorbeeld, spelen deze problemen niet. Een aanraking op een aanraakscherm werkt in dit geval hetzelfde als hoveren met een muis.

#### **:FOCUS**

Omdat `:focus` mogelijk niet werkt, als css uitstaat, ontbreekt of onvolledig is geïmplementeerd, moet je nooit belangrijke informatie alleen met behulp van `:focus` tonen.

Daar wordt in dit voorbeeld fors van afgeweken: het werkt stomweg helemaal niet zonder css. Dat het helemaal niet werkt, is echter direct duidelijk. Mensen kunnen dan css inschakelen, de pagina opnieuw laden, geen tekstbrowser gebruiken, of wat dan ook. Hierboven wordt meer bedoeld, dat bijvoorbeeld links niet allemaal herkenbaar zijn. En dat valt mogelijk helemaal niet op.

De meeste mensen gaan met een muis naar een link, invoerveld, e.d. Waarna ze vervolgens klikken om de link te volgen, tekst in te voeren, of wat ze ook maar willen doen. (Dit geldt uiteraard niet voor aanraakschermen.)

Er is echter 'n tweede manier om naar links, invoervelden, e.d. te gaan: met behulp van de Tab-toets kun je naar links, invoervelden, e.d. 'springen'. (Over het gebruik van de Tab-toets staat meer bij [Toetsenbordnavigatie en tabindex](#).)

Waar je staat, wordt door alle browsers aangegeven met een of ander kadertje. De link e.d. met het kadertje eromheen 'heeft focus'. Dat wil zeggen dat je die link volgt, als je op de Enter-toets drukt, of dat je in dat veld tekst kunt gaan invoeren, enz.

Het kadertje dat de focus aangeeft, moet nooit zonder meer worden weggehaald. Gebruikers van de Tab-toets hebben dan geen idee meer, waar ze zijn.

Bij gebruik van de muis of een aanraakscherm is het kadertje niet nodig, want je mag toch hopen dat iemand weet, waar iemand iets heeft aangeraakt of -geklikt. Als je dat niet meer weet, valt te vrezen dat een kadertje ook geen redding meer biedt.

In het verleden was dat vaak een probleem, omdat dat kadertje nou niet bepaald erg mooi is. En dus eigenlijk alleen bij gebruik van de Tab-toets nodig is. Dit is opgelost met de pseudo-class `:focus-within`: de focus wordt alleen nog aangegeven, als een toetsenbord wordt gebruikt. Bij een klik of een aanraking zie je het kadertje niet meer.

Inmiddels is `:focus-within` in alle iets nieuwere browsers de standaard, dus eigenlijk hoef je `:focus` en/of `:focus-within` alleen nog te gebruiken, als je bijvoorbeeld het uiterlijk van het kadertje wilt aanpassen.

In dit voorbeeld is het effect van `:focus` afhankelijk van de breedte van het browservenster.

In browservensters smaller dan 500 px wordt de naam van de provincie vet, als de bijbehorende `<input>` de focus heeft. Het tekstveld bij de `<input>` wordt breder, zodat er voldoende ruimte is om de naam van de hoofdstad in te vullen.

In browservensters met een breedte van 500 tot en met 759 px is iets meer ruimte. Daar wordt de naam van de provincie bij focus niet vet, maar iets groter.

In browservensters met een breedte van 760 px of meer wordt de naam van de provincie nog sterker vergroot.

#### **:ACTIVE**

Omdat `:active` mogelijk niet werkt, als css uitstaat, ontbreekt of onvolledig is geïmplementeerd, moet je nooit belangrijke informatie alleen met behulp van `:active` tonen.

Een element is actief, als de muis wordt ingedrukt boven dat element. Op sommige aanraakschermen is het element actief, als het wordt aangeraakt.

`:active` wordt niet gebruikt in dit voorbeeld.

### **De code aanpassen aan je eigen ontwerp**

- Als je dit voorbeeld gaat aanpassen voor je eigen site, houd het dan in eerste instantie zo eenvoudig mogelijk. Ga vooral geen details invullen.
- Gebruik geen FrontPage, Publisher of Word (alle drie van Microsoft). Publisher en Word zijn niet bedoeld om websites mee te maken. FrontPage is zwaar verouderd en wordt al jaren niet meer onderhouden door Microsoft. Ook OpenOffice en LibreOffice leveren een uiterst beroerd soort html af. Tekstverwerkers met al hun toeters en bellen zijn gewoon niet geschikt om websites mee te bouwen. Je kunt beter een goed (gratis) programma gebruiken. Links naar dat soort programma's vind je op de pagina met [links](#) onder Gereedschap → wysiwyg-editor en Gereedschap → Editors en IDE's. 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 aanraakscherm. Inmiddels hebben veel browsers in de ontwikkelgereedschappen mogelijkheden voor het simuleren van weergave op een kleiner scherm ingebouwd. Ook dit blijft een simulatie, maar geeft vaak wel een beter beeld dan zoomen.
- Als je 'n site maakt in Firefox, Opera, Safari, Google Chrome, Vivaldi of Edge, is er 'n hele grote kans dat die in alle browsers werkt. Hier wordt de voorkeur gegeven aan Firefox, omdat het een van de weinige browsers is, die niet bij een bedrijf hoort dat vooral op je centen of je data uit is. Google Chrome wordt ook door veel mensen gebruikt, maar slaat ondertussen wel je hele surfgedrag, je schoenmaat en de kleur van je onderbroek op. Daarom wordt Google Chrome alleen gebruikt om in te testen.

- Het allereerste dat je moet invoeren, is het doctype, vóór welke andere code dan ook. Een lay-out met een missend of onvolledig doctype ziet er totaal anders uit dan een lay-out met een geldig doctype. Wát er anders is, verschilt ook nog 'ns tussen de diverse browsers. Als je klaar bent en dan nog 'ns 'n doctype gaat invoeren, weet je vrijwel zeker dat je van voren af aan kunt beginnen met de lay-out.  
Geldige doctypes vind je op [www.w3.org/QA/2002/04/valid-dtd-list](http://www.w3.org/QA/2002/04/valid-dtd-list).  
Gebruik het volledige doctype, inclusief de eventuele url, anders werkt het niet goed.
- Gebruik het doctype voor html5. Dat is bedoeld voor nieuwe sites.  
Het oudere transitional doctype staat talloze tags toe, die in html5 zijn verboden. Deze tags worden al zo'n tien jaar afgeraden. Het transitional doctype is alleen bedoeld om de puinhoop van vroeger, toen niet volgens standaarden werd gewerkt, enigszins te herstellen. Het (ook al oudere) 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.
- De eerste regel binnen de `<head>` moet de charset zijn. Dit vertelt de browser, welke tekenset er gebruikt moet worden, zodat letters met accenten e.d. overal goed worden weergegeven. Het beste kun je utf-8 nemen. Als je later van charset verandert, loop je 'n grote kans dat je alle aparte tekens als letters met accenten weer opnieuw moet gaan invoeren. In html5 is het simpele `<meta charset="utf-8">` voldoende.
- Test vanaf het allereerste begin in zoveel mogelijk verschillende browsers in 'n aantal resoluties (schermgroottes). Onder het kopje [Getest in](#) kun je in deze uitleg vinden, waar dit voorbeeld in is getest.
- Voor alle voorbeelden geldt: breng veranderingen stapsgewijs aan. Als je bijvoorbeeld foto's wilt laten weergeven, begin dan met het alleen veranderen van de namen van de foto's, zodat je eigen foto's worden weergegeven. Maakt niet uit als de maten niet kloppen en de teksten fout zijn. Als dat werkt, ga dan bijvoorbeeld de maten aanpassen. Dan de teksten. En controleer steeds, of alles nog goed werkt.
- Als het om een lay-out of iets dergelijks gaat: zorg eerst dat header, kolommen, footer, menu, en dergelijke staan en bewegen, zoals je wilt. Ga daarna pas details binnen die blokken invullen. In eerste instantie gebruik je dus bijvoorbeeld 'n leeg blok op de plaats, waar uiteindelijk het menu komt te staan.  
Als je begint met allerlei details, is er 'n heel grote kans dat die de werking van de blokken gaan verstoren. Bouw eerst het huis, en ga dan pas de kamers inrichten. Zorg eerst dat de blokken werken, zoals je wilt. Dan zul je het daarna gelijk merken, als 'n toegevoegd detail als tekst of 'n afbeelding iets gaat verstoren. Daarvoor moet je natuurlijk wel regelmatig controleren in verschillende browsers, of alles nog wel goed werkt.  
Je kunt de blokken tijdens het aanpassen opvullen met bijvoorbeeld `<br>1<br>2<br>3` enz., tot ze de juiste hoogte hebben. Het is handig om aan het einde even iets toe te voegen als 'laatste', zodat je zeker weet dat er niet ongemerkt drie regels onderaan naar 't virtuele walhalla zijn verhuisd.  
Om de breedte te vullen, kun je het best 'n kort woord als 'huis' duizend keer of zo herhalen. Ook hier is het handig om aan 't eind (en hier ook aan 't begin) 'n herkenningsteken te maken, zodat je zeker weet dat je de hele tekst ziet.
- Zolang je in grotere dingen zoals 'n lay-out aan 't wijzigen bent, kan het helpen de verschillende delen een achtergrondkleur of een outline te geven. Je ziet dan goed, waar 'n deel precies staat. Een achtergrondkleur en een outline hebben – anders dan bijvoorbeeld een border – verder geen invloed op de lay-out, dus die zijn hier heel geschikt voor.

- Als je eigenschappen verandert in de css, verander er dan maar één, hooguit twee tegelijk. Als je er zeventien tegelijk verandert, is de kans groot dat je niet meer weet, wat je hebt gedaan. En dat je 't dus niet meer terug kunt draaien.
  - `margin`, `padding` en `border` worden bij de breedte en hoogte 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 px aan beide kanten), en 4 px hoger. Zoiets kan je hele lay-out verstoren, omdat iets net te breed of te hoog wordt. Je moet dan elders iets 4 px kleiner maken. Dat zal vaak zo zijn: als je één maat verandert, zul je vaak ook 'n andere moeten aanpassen. Css geeft de mogelijkheid om met behulp van `box-sizing` de padding en border binnen de breedte en hoogte van de inhoud te zetten, als je dat handiger vindt. Met nieuwere css-eigenschappen als grid en flexbox, die speciaal zijn gemaakt om een lay-out mee te maken, spelen dit soort problemen veel minder. In alle browsers waarop hier nog wordt getest, werken flexbox en grid prima. Maar als je oudere browsers moet ondersteunen, kan dat wel problemen opleveren en moet je ook in die oudere browsers testen.
  - In plaats van een absolute eenheid als px kun je ook een relatieve eenheid gebruiken, met name `em` en `rem`. Voordeel van `em` en `rem` is dat een lettergrootte, regelhoogte, e.d. in `em` en `rem` in alle browsers kan worden veranderd. Nadeel is dat het de lay-out sneller kan verstoren dan bijvoorbeeld px. Dit moet je gewoon van geval tot geval bekijken. Voor weergave in mobiele apparaten zijn relatieve eenheden als `em` en `rem` vrijwel altijd beter dan absolute eenheden als px.  
(De minder bekende `rem` is ongeveer hetzelfde als de `em`. Alleen is de lettergrootte bij `rem` gebaseerd op de lettergrootte van het `<html>`-element, waardoor de `rem` overal op de pagina precies even groot is. Bij de `em` kan de lettergrootte worden beïnvloed door de voorouders van het element, bij de `rem` niet.)  
Zoomen kan trouwens altijd, ongeacht welke eenheid je gebruikt.
  - Valideren, valideren, valideren en dan voor 't slapen gaan nog 'ns valideren. Valiwié???
- Valideren is het controleren van je html en css op 'n hele serie fouten. Computers zijn daar vaak veel beter in dan mensen. Als je 300 keer `<h2>` hebt gebruikt en 299 keer `</h2>` vindt 'n computer die ene missende `</h2>` zonder enig probleem. Jij ook wel, maar daarna ben je misschien wel aan vakantie toe.
- Valideren kan helpen om gekmakende fouten te vinden. Valide code garandeert ook dat de weergave in verschillende browsers (vrijwel) hetzelfde is. En valide code is over twintig jaar ook nog te bekijken.
- Valideren moet trouwens ook niet worden overdreven. Het is een hulpmiddel om echte fouten te vinden, meer niet. Het gaat erom dat je site goed werkt, niet dat je het braafste kind van de klas bent. Als de code niet valideert, maar daar is een goede reden voor, is daar niets op tegen. Zeker met nieuwere html en css wil de validator nog wel eens achterlopen, terwijl dat al prima is te gebruiken.
- Op deze site is alle css en html gevalideerd. Als de code niet helemaal valide is (wat regelmatig voorkomt), staat daar onder [Bekende problemen \(en oplossingen\)](#) de reden van. Je kunt je css en html valideren als 't online staat, maar ook als het nog in je computer staat. Html kun je valideren op: [validator.w3.org/nu](http://validator.w3.org/nu).  
Css kun je valideren op: [jigsaw.w3.org/css-validator](http://jigsaw.w3.org/css-validator).

## Toegankelijkheid en zoekmachines

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

Toegankelijkheid (in het Engels 'accessibility') is belangrijk voor bijvoorbeeld blinden die een schermlezer gebruiken, of voor motorisch gehandicapte mensen die moeite hebben met het bedienen van een muis. Een spider van een zoekmachine (dat is het programmaatje dat de site indexeert voor de zoekmachine) is te vergelijken met een blinde. Als je je site goed toegankelijk maakt voor gehandicapten, is dat gelijk goed voor een hogere plaats in een zoekmachine. Dus als je 't niet uit sociale motieven wilt doen, kun je 't uit egoïstische motieven doen.

(Op die plaats in de zoekmachine heb je maar beperkt invloed. De toegankelijkheid van je site is maar één van de factoren, maar zeker niet onbelangrijk.)

Als je bij het maken van je site al rekening houdt met toegankelijkheid, is dat nauwelijks extra werk. 't Is ongeveer te vergelijken met inbraakbescherming: doe dat bij 'n nieuw huis en 't is nauwelijks extra werk, doe 't bij 'n bestaand huis en 't is al snel 'n enorme klus.

Enkele tips die helpen bij toegankelijkheid:

- Gebruik altijd een alt-beschrijving bij een afbeelding. De alt-tekst wordt gebruikt door schermlezers en 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.

- Gebruik in links een tekst die duidelijk aangeeft, waar de link naartoe gaat. Een tekst als 'pagina met externe links' is waarschijnlijk duidelijk genoeg, een tekst als alleen 'links' waarschijnlijk niet. Een duidelijke zwart-witregel is niet te geven, omdat dit ook van tekst e.d. in de omgeving van de link afhangt.

ScherMLEzers kunnen een lijst van alle links in de pagina weergeven, en een duidelijke tekst is daarbij belangrijk. Alleen 'volgende' zegt niets, als dat in 'n lijst met alleen links staat.

- Accesskeys (sneltoetsen) kun je beter niet gebruiken, deze geven te veel problemen, omdat ze vaak dubbelop zijn met sneltoetsen voor de browser of andere al gebruikte sneltoetsen. Bovendien is voor de gebruiker meestal niet duidelijk, welke toetsen het zijn.

Op zichzelf zijn accesskeys een heel goed idee. Maar helaas zijn ze ook in html5 volstrekt onvoldoende gedefinieerd. Er is nog steeds geen standaard voor de meest gebruikelijke accesskeys, zoals Zoek of Home.

Er is nog steeds niet vastgelegd, hoe accesskeys zichtbaar gemaakt kunnen worden. Voor de makers van browsers zou dit 'n relatief kleine moeite zijn, voor de makers van 'n site is het bergen extra werk.

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

- Met behulp van de Tab-toets (of op 'n soortgelijke manier) kun je door links, invoervelden, e.d. lopen. Elke tab brengt je één link, invoerveld, e.d. verder, Shift+Tab één plaats terug. Met behulp van het attribuut `tabindex` kun je de volgorde aangeven, waarin de Tab-toets werkt. Zonder `tabindex` wordt de volgorde van de html aangehouden bij gebruik van de Tab-toets, maar soms is een andere volgorde logischer.

In principe is het beter, als `tabindex` niet nodig is, maar gewoon de volgorde van de html wordt aangehouden. Bij verkeerd gebruik kan `tabindex` heel verwarrend zijn. Het is niet bedoeld om van de pagina een hindernisbaan voor kangoeroes te maken, waarop van

beneden via links over rechts naar boven wordt gesprongen. (Meer over de Tab-toets is te vinden [Toetsenbordnavigatie en tabindex](#).)

- Als, zoals hierboven beschreven, een gebruiker van de Tab-toets bij een link, invoerveld, e.d. is aangekomen, heeft dit element 'focus'. Dit wordt aangegeven door de link, invoerveld, e.d. extra te markeren met een kadertje. Dat kadertje mag je alleen weghalen, als op een andere manier wordt duidelijk gemaakt, welk element focus heeft. Een gebruiker van de Tab-toets kan anders niet zien, waar die zit, en welk element gaat reageren op bijvoorbeeld een Enter. (Meer over focus is te vinden bij [focus](#).)
- 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. Schermlezers houden namelijk altijd de volgorde van de html aan en niet een eventueel afwijkende volgorde op het scherm.
- Een zogenaamde skip-link is vaak nog wel zinvol. Dat is een link die je buiten het scherm parkeert met behulp van css, zodat die normaal genomen niet te zien is. Zo'n link is wel gewoon zichtbaar in speciale programma's zoals tekstbrowsers en schermlezers, want die kijken gewoon naar wat er in de broncode staat.  
(Alleen in de schermlezer TalkBack op oudere versies van Android werkt zo'n buiten het scherm geplaatste link niet. TalkBack leest zo'n link wel voor, maar de link kan niet worden gevolgd, als deze buiten het scherm staat. Met ingang van versie 8.1 van Android is dit eindelijk opgelost en werkt een skip-link ook fatsoenlijk in TalkBack.)  
Een skip-link staat bovenaan de pagina, nog boven menu, header, e.d., en linkt naar de eigenlijke inhoud van de pagina. Hierdoor kunnen mensen met één toetsaanslag naar de eigenlijke inhoud van de pagina gaan.  
Een skip-link is vooral nuttig voor gebruikers van de Tab-toets. Zodra de normaal genomen onzichtbare link door het indrukken van de Tab-toets focus krijgt, kun je de link op het scherm plaatsen, waardoor deze zichtbaar wordt. Bij een volgende tab wordt de link dan weer buiten het scherm geplaatst en is dus niet meer zichtbaar, zodat de lay-out niet wordt verstoord.  
Op pagina's en in voorbeelden waar dat nuttig is, wordt op deze site een skip-link gebruikt.
- Van oorsprong is html een taal om wetenschappelijke documenten weer te geven, pas later is deze gebruikt voor lay-out. Maar daar is de taal 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 extern stijlbestand staan of, als deze alleen voor één bepaalde pagina van toepassing is, in de <head> van die pagina.
- Breng een logische structuur aan in je document. Gebruik een <h1> voor de belangrijkste kop, een <h2> voor een subkop, enz. Schermlezers e.d. kunnen van kop naar kop springen. En een zoekmachine gaat ervan uit dat <h1> belangrijke tekst bevat.  
Dit geldt voor al dit soort structuurbepalende tags.  
Als een <h1> te grote letters geeft, maak daar dan met behulp van je css 'n kleinere letter van, maar blijf die <h1> gewoon gebruiken. Op dezelfde manier kun je al dit soort dingen oplossen.
- <table> is fantastisch, maar alleen als die wordt gebruikt om een echte tabel weer te geven, niet als <table> 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 papieren krant van links naar rechts: niet per kolom, maar per regel. Dat gaat dus alleen maar goed bij een echte tabel, zoals een spreadsheet. In alle

andere gevallen garandeert 'n tabel volstreekte ontoegankelijkheid voor schermlezers e.d. en als extra bonus vaak 'n lagere plaats in een zoekmachine.

- Frames horen bij een volstrekt verouderde techniek, die heel veel nadelen met zich meebrengt. <iframe>'s hebben voor een deel dezelfde nadelen. Eén van die nadelen is dat de verschillende frames voor zoekmachines, schermlezers, e.d. als los zand aan elkaar hangen, omdat ze los van elkaar worden weergegeven. Ze staan wel naast elkaar op het scherm, maar er zit intern geen verband tussen.

Als je 'n stuk code vaker wilt gebruiken, zoals 'n menu dat op elke pagina hetzelfde is, voeg dat dan in met PHP. Dan wordt de pagina niet pas in de browser, maar al op de server samengesteld. Hierdoor zien zoekmachines, schermlezers, e.d. één pagina, net zoals wanneer je maar één pagina met html zou hebben geschreven.

(Je kunt ook invoegen met behulp van SSI (Server Side Includes). Maar tegenwoordig kun je beter PHP dan SSI gebruiken, omdat SSI min of meer aan het uitsterven is en PHP veel meer mogelijkheden heeft.)

- Geef de taal van het document aan, en bij woorden en dergelijke die afwijken van die taal de afwijkende taal met behulp van `lang="..."`. Op deze site gebeurt dat lang niet overal, 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.

Op deze site wordt de lijst op [woordenlijst.org](http://woordenlijst.org) gebruikt om te bepalen, of een woord inmiddels 'Nederlands' is. Als het woord in deze lijst voorkomt, wordt geen `lang`-attribuut gebruikt, ook niet als het woord oorspronkelijk uit een andere taal komt.

- Gebruik de tag <abbr> bij afkortingen. Doe dat de eerste keer op een pagina samen met de title-eigenschap: <abbr title="ten opzichte van">t.o.v.</abbr>. Daarna kun je op dezelfde pagina volstaan met <abbr>t.o.v.</abbr>. Doe je dit niet, dan is er 'n grote kans dat 'n schermlezer 't.o.v.' uit gaat spreken als 'tof', en 'n zoekmachine kan er ook geen chocola van maken.

- Geef een verandering niet alleen door kleur aan. Een grote groep mensen heeft moeite met het onderscheiden van kleuren en/of het herkennen van kleuren. Verander bijvoorbeeld een ronde rode knop niet in een ronde groene knop, maar in een vierkante groene knop. Door ook de vorm te veranderen, is het herkennen van de verandering niet alleen van een kleur afhankelijk.

- Zorg voor voldoende contrast tussen achtergrond- en voorgrondkleur, tussen `background-color` en `color`. Soms zie je heel lichtgrijze tekst op een donkergrijze achtergrond, en dan ook nog in een mini-formaat. Dat is dus voor heel veel mensen stomweg volledig onleesbaar. Op de pagina met [links](#) staat onder het kopje Toegankelijkheid → Contrast en kleurenblindheid een hele serie sites, waar je kunt controleren of het contrast groot genoeg is.

- 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.

Op Linux kun je met Lynx kijken, hoe je pagina eruitziet zonder plaatjes e.d., als echt alleen de tekst overblijft. Een installatie-programma voor Lynx op Windows is te vinden op [invisible-island.net/lynx](http://invisible-island.net/lynx).

Ook kun je in Windows het gratis programma WebbIE installeren. WebbIE laat de pagina zien, zoals een tekstbrowser e.d. deze ziet. WebbIE is te downloaden vanaf [www.webbie.org.uk](http://www.webbie.org.uk).

- Ten slotte kun je je pagina nog online op toegankelijkheid laten controleren op 'n behoorlijk aantal sites, zoals:

[lowvision.support](#): laat zien hoe een kleurenblinde de site ziet. Engelstalig.

[wave.webaim.org](#): deze laat grafisch zien, hoe de toegankelijkheid is. Engelstalig. Deze tester is ook als extensie in Firefox en Google Chrome te installeren.

Op de pagina met [links](#) kun je onder Toegankelijkheid links naar meer tests e.d. vinden.

## Getest in

Laatst gecontroleerd op 26 maart 2025.

Onder dit kopje staat alleen maar, hoe en waarin is getest. Alle eventuele problemen, ook die met betrekking tot zoomen, lettergroottes, toegankelijkheid, uitstaan van JavaScript en/of css, enz. staan iets hieronder bij [Bekende problemen \(en oplossingen\)](#). Het is belangrijk dat deel te lezen, want uit een test kan ook prima blijken dat iets totaal niet werkt!

Dit voorbeeld is getest op de volgende systemen:

### DESKTOPCOMPUTERS

Linux (KDE neon 6.3) (2560 x 1080 px, resolutie: 96 ppi):  
Firefox, Google Chrome en Vivaldi, in grotere en kleinere browservensters.  
In Vivaldi is ook ruimtelijke navigatie ('Spatial Navigation') getest.

### LAPTOPS

Windows 10 (1600 x 900 px, resolutie: 106 ppi):  
Firefox, Google Chrome en Edge, in grotere en kleinere browservensters.

OS X 11.7.10 ('Big Sur') (1440 x 900 px, resolutie: 96 ppi):  
Firefox, Safari, Google Chrome en Microsoft Edge, in grotere en kleinere browservensters.

OS X 15.4.1 ('Sequoia') (2500 x 1600 px, 227 ppi):  
Firefox, Safari, Google Chrome en Microsoft Edge, in grotere en kleinere browservensters.

### TABLETS

iPad met iOS 12.5.7 (2048 x 1536 px, device-pixel-ratio: 2):  
Safari, Chrome, Firefox, Microsoft Edge (alle portret en landschap).

iPad met iPadOS 18.4.1 (2160 x 1620 px, resolutie: 264 ppi):  
Safari, Chrome, Firefox, Microsoft Edge (alle portret en landschap).

Android 6.0 ('Marshmallow') (1920 x 1200 px, resolutie: 224 ppi):  
Samsung Internet, Firefox en Chrome (alle portret en landschap).

Android 8.1 ('Oreo') (1920 x 1200 px, resolutie: 218 ppi):  
Samsung Internet, Firefox, Microsoft Edge en Chrome (alle portret en landschap).

Android 13 ('Tiramisu') (2000 x 1200 px, resolutie: 225 ppi):  
Samsung Internet, Firefox, Microsoft Edge en Chrome (alle portret en landschap).

### SMARTPHONES

iPhone met iOS 15.8.4 (1334 x 750, resolutie: 326 ppi):  
Safari, Chrome, Firefox en Microsoft Edge (alle portret en landschap).

iPhone met iOS 18.4.1 (1334x750, resolutie: 326 ppi):  
Safari, Chrome, Firefox en Microsoft Edge (alle portret en landschap).

Android 7.0 ('Nougat') (1280 x 720 px, resolutie: 294 ppi):  
Samsung Internet, Firefox, Microsoft Edge en Chrome (alle portret en landschap).

Android 9.0 ('Pie') (1920 x 1080 px, resolutie: 424 ppi):  
Samsung Internet, Firefox, Microsoft Edge en Chrome (alle portret en landschap).

Android 14 ('Upside Down Cake') (2408 x 1080 px, resolutie: 401 ppi):  
Samsung Internet, Firefox, Microsoft Edge en Chrome (alle portret en landschap).

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

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

Er is getest met behulp van muis en toetsenbord, behalve op iOS, iPadOS en Android, waar een aanraakscherm is gebruikt. Op Windows 10 is getest met aanraakscherm, touchpad, toetsenbord, muis, en – waar dat zinvol was – op een combinatie daarvan. Op OS X 11.7.10 en 15.4.1 is getest met (een combinatie van) toetsenbord, touchpad en muis.

Als in een voorbeeld JavaScript is gebruikt, is ook getest of het werkt zonder JavaScript. Dat is alleen gedaan in de browsers, waarin in de instellingen JavaScript kan worden uitgeschakeld.

Ook is getest zonder css en – als afbeeldingen worden gebruikt – zonder afbeeldingen.

#### **SCHERMLEZERS E.D.**

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

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

[WebbIE](#) is een browser die gericht is op mensen met een handicap. Er is getest op Windows 10.

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

TalkBack is een in Android ingebouwde schermlezer. Er is getest in combinatie met Chrome op Android 6.0, 7.0, 8.1, 9, 13 en 14.

VoiceOver is een in iOS en OS X ingebouwde schermlezer. Er is getest in combinatie met Safari op iOS 12.5.7, 15.8.4 en 18.4.1, iPadOS 18.4.1, OS X 11.7.10 en 15.4.1.

Orca is een schermlezer voor Linux. Er is getest in combinatie met Firefox op KDE neon 6.3.

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

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

Als het voorbeeld in deze programma's toegankelijk is, zou het in principe toegankelijk moeten zijn in alle aangepaste browsers en dergelijke. En dus ook voor zoekmachines, want een zoekmachine is redelijk vergelijkbaar met een blinde.

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

Waar dat zinvol was, is ook nog getest op combinaties als een grote letter in een schermlezer met toetsenbordbediening.

Alleen op de hierboven genoemde systemen en browsers is getest. Er is dus niet getest op bijvoorbeeld 'n Blackberry. Er is een kans dat dit voorbeeld niet (volledig) werkt op niet-geteste systemen en apparaten. Om het wel (volledig) werkend te krijgen, zul je soms (kleine) wijzigingen en/of (kleine) aanvullingen moeten aanbrengen, bijvoorbeeld met JavaScript.

Er is ook geen enkele garantie dat iets werkt in een andere tablet of smartphone dan hierboven genoemd, omdat fabrikanten in principe de software kunnen veranderen. Dit is anders dan op de desktop, waar browsers altijd (vrijwel) hetzelfde werken, zelfs op verschillende besturingssystemen. Iets wat in Samsung Internet op Android werkt, zal in de regel overal werken in die browser, maar een garantie is er niet. De enige garantie is het daadwerkelijk testen op een fysiek apparaat. En aangezien er duizenden mobiele apparaten zijn, is daar geen beginnen aan.

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

Nieuwe browsers worden pas getest, als ze uit het bèta-stadium zijn. Anders is er 'n redelijke kans dat je tegen 'n bug zit te vechten, die voor de uiteindelijke versie nog gerepareerd wordt.

Dit voorbeeld is alleen getest in de hierboven met name genoemde browsers. Vragen over niet-geteste browsers kunnen niet worden beantwoord, en het melden van fouten in niet-geteste browsers heeft ook geen enkel nut. (Melden van fouten, problemen, enz. in wel geteste browsers: graag! Dat kan op het [forum](#).)

### **Bekende problemen (en oplossingen)**

Waarop en hoe is getest, kun je gelijk hierboven vinden bij [Getest in](#).

Als je hieronder geen oplossing vindt voor een probleem dat met dit voorbeeld te maken heeft, kun je op het [forum](#) proberen een oplossing te vinden voor je probleem. Om forumspam te voorkomen, moet je je helaas wel registreren, voordat je op het forum een probleem kunt aankaarten.

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

Voor zover van toepassing wordt eerst het ontbreken van JavaScript, css en/of afbeeldingen besproken. Vervolgens problemen en aanpassingen met betrekking tot toegankelijkheid voor specifieke groepen bezoekers, toetsenbordnavigatie, tekstbrowsers, schermlezers en zoomen en andere lettergrootte. Als laatste volgen de overige problemen in één of meer specifieke browsers.

Als in een onderdeel geen problemen aanwezig zijn, staat in een smal groen kadertje 'Geen problemen'. Bij een onderwerp over toegankelijkheid zijn er soms geen problemen, maar alleen aanpassingen. Ook in dat geval staat bovenaan in een smal groen kadertje 'Geen problemen'. Daaronder staan dan de aanpassingen.

Als in een onderdeel één of meer problemen worden besproken, staat van elk probleem in een breed rood kadertje een korte samenvatting daarvan.

Als bij het probleem een oplossing wordt gegeven, staat de samenvatting in een rode stippellijn. Bij een onderwerp over toegankelijkheid zijn er soms, naast de opgeloste problemen, ook aanpassingen. In dat geval staan die aanpassingen boven de kadertjes met opgeloste problemen.

Als bij het probleem geen oplossing is gevonden, staat de samenvatting in een rode ononderbroken lijn. Bij een onderwerp over toegankelijkheid zijn er soms, naast de problemen, ook aanpassingen. In dat geval staan die aanpassingen boven de kadertjes met problemen.

## ZONDER CSS

Probleem: zonder css werkt dit voorbeeld niet.



Groningen Deze stad is fout. Deze stad is goed. Fout!  Gr:  
Friesland Deze stad is fout. Deze stad is goed. Fout!  Fr:

Zonder css is dit voorbeeld volledig onbruikbaar.

Op de afbeelding is een klein stukje van de weergave in een smal browservenster te zien: de kaart van Nederland met daaronder in een grote lijst alle teksten en invoervelden. Inclusief goed- en foutmeldingen, maar zonder enige daadwerkelijk controle op een juiste invoer. Ook de tellers met het resultaat ontbreken.

## ZONDER AFBEELDINGEN

Probleem: zonder afbeeldingen werkt dit voorbeeld niet.

Invullen van provinciehoofdsteden met markering en aantal goede en foute

Vul de provinciehoofdsteden in:

Goed: 0. Fout: 0. Nog niet ingevuld: 12.

Kaart van Nederland met provinciehoofdsteden

Naar de uitslag

De grootte van de afbeelding bepaalt de grootte van `div#wrapper`. Vrijwel alle elementen worden gepositioneerd ten opzichte van `div#wrapper`. Als er geen afbeelding is, heeft `div#wrapper` nauwelijks een hoogte. Omdat alle `<labels>`'s en `<input>`'s nu ten opzichte van die kleine hoogte worden neergezet, staan die allemaal over elkaar heen. Heel gezellig als je van feestjes houdt, maar zoals dat op feestjes vaker voorkomt: wie bij wie hoort is volstrekt onduidelijk en helemaal helder zie je het ook niet meer.

## TOETSEN BORDNAVICATIE

Geen problemen.

Met de [Tab-toets](#) kan van tekstveld naar tekstveld worden gegaan. In lagere browservensters kan de uitslag bovenaan het venster daardoor boven het venster

komen te staan, waardoor de uitslag niet meer is te zien. Daarom is helemaal onderaan de pagina een link neergezet, waarmee je weer terug naar boven kunt gaan.

## TEKSTBROWSERS

Probleem: in tekstbrowsers werkt dit voorbeeld niet.

Dit voorbeeld is afhankelijk van css en van een afbeelding. Beide worden nou juist (vrijwel) volledig weggelaten door tekstbrowsers, waardoor dit voorbeeld niet werkt in een tekstbrowser.

In Lynx kun je de tekstvelden invullen, maar er is geen enkele controle op de juistheid van de hoofdsteden. Bovendien staan alle teksten achter elkaar, ongeveer zoals iets hierboven bij [Zonder css](#) is afgebeeld.

Voor Webbie geldt hetzelfde, maar hierin kun je zelfs de tekstvelden niet invullen.

## SCHERMLEZERS

Probleem: schermlezers lezen niet gelijk na verkeerd invullen een foutmelding voor.

Als een hoofdstad goed wordt ingevuld, krijgt gelijk na het verlaten van het tekstveld de provincienaam een groene achtergrond en een kadertje. Als een hoofdstad fout wordt ingevuld, verschijnt gelijk na het verlaten van het tekstveld de melding 'Fout!'.

Schermlezers hebben hier niets aan. Een groene achtergrond en een kadertje worden hoe dan ook niet voorgelezen, maar ook de melding 'Fout!' wordt niet voorgelezen. (Ook niet als die op een andere plaats in de html wordt neergezet, bijvoorbeeld in een `<span>` na de `<input>`, in plaats van zoals nu in een `<span>` binnen de `<label>`.)

Het woordje 'Fout!' is ook eigenlijk niet erg geschikt om voor te lezen, want het is niet echt duidelijk waar dat bij hoort. Als je terug naar boven gaat, is volstrekt onduidelijk bij welke hoofdstad 'Fout!' hoort. Op het scherm is dat duidelijk te zien, bij voorlezen niet.

Daarom wordt met behulp van [aria-hidden](#) de `<span>` met het woordje 'Fout!' voor schermlezers verborgen. De groene achtergrond en het kadertje bij juist invullen hoeven niet verborgen te worden, want die worden toch niet voorgelezen.

Voor schermlezers is geprobeerd om, gelijk na het verlaten van het tekstveld, bij verkeerd invullen van een hoofdstad de tekst 'De vorige stad is fout' voor te laten lezen. Dit zou moeten kunnen met behulp van `aria-live="assertive"`. Als een element met bijvoorbeeld `display: none;` is verborgen, zou `aria-live="assertive"` de tekst hierin onmiddellijk moeten voorlezen, zodra het element zichtbaar wordt gemaakt. Dit blijkt hier echter niet goed te werken.

Verteller negeert dit vrijwel volledig. Heel af en toe wordt het voorgelezen als je teruggaat naar het vorige veld en dan weer naar het verkeerd ingevulde veld. Waarom dit af en toe wel en af en toe niet wordt voorgelezen, is volstrekt onduidelijk.

NVDA zegt het op het juiste moment, maar als je teruggaat naar het verkeerd ingevulde veld wordt het zowel voor als na het verkeerd ingevulde veld voorgelezen.

Orca negeert het gewoon volledig, hoe vaak je ook heen en weer gaat in de pagina. TalkBack op Android 6 negeert de melding volledig.

TalkBack op Android 7 en Android 8.1 negeren het. Pas als je de tweede keer door de pagina gaat, wordt het voorgelezen.

TalkBack op Android 9 en Android 13 maken er helemaal een creatieve vertoning van. Na het verlaten van het verkeerd ingevulde veld wordt de volgende provincienaam voorgelezen, en pas daarna volgt de foutmelding. Als je voor de tweede keer door de pagina gaat, wordt het wel correct voorgelezen.

TalkBack op Android 14 doet het goed, maar bij de tweede keer door de pagina gaan wordt de melding voor én na het verkeerd ingevulde veld voorgelezen.  
VoiceOver op iOS 12.5.7. negeert de melding volledig.  
VoiceOver op iPadOS 18.3.1 is de enige die alles goed doet: de melding wordt gelijk na het verlaten van een verkeerd ingevuld veld voorgelezen.  
VoiceOver op OS X 11.7.10 en OS X 15.4.1 negeren de melding volledig.  
Het maakt hierbij niet uit, op welke plaats in de html de span met `aria-live="assertive"` staat. Deze chaos levert alleen maar verwarring op (als het al wordt voorgelezen), daarom wordt `aria-live` niet gebruikt.

Als de pagina voor de eerste keer wordt doorlopen, krijgen schermlezers nu geen enkele melding. Als de pagina voor de tweede keer wordt doorlopen, lezen schermlezers tussen de provincie en de provinciehoofdstad 'Deze stad is goed' of 'Deze stad is fout'. Maar dit gebeurt dus pas als de pagina voor de tweede keer wordt voorgelezen.  
Niet ideaal, maar wel duidelijk.

**Probleem: TalkBack op Android 6 leest de goed- en foutmeldingen niet voor.**

Zoals gelijk hierboven beschreven, moeten schermlezers bij het de tweede keer doorlopen van de pagina gelijk voor elke hoofdstad 'Deze stad is goed' of 'Deze stad is fout' voorlezen. TalkBack op Android 6 leest deze teksten niet voor.

#### **ZOOMEN EN ANDERE LETTERGROOTTE**

In browservensters smaller dan 760 px zijn de provincienamen en tekstvelden zo gepositioneerd, dat ze elkaar ook bij een lettergrootte van 200% niet of nauwelijks overlappen. Daardoor is de positionering bij een standaardlettergrootte iets minder ideaal, maar echt veel maakt het niet uit.

**Probleem: in kleinere browservensters zijn de namen van de provinciehoofdsteden die onder de Nederland worden neergezet niet meer volledig zichtbaar.**



Als de letters worden vergroot tot ongeveer 135%, zijn in kleinere browservensters de namen van de provinciehoofdsteden niet meer volledig zichtbaar.

In een browservenster van 320 px breed, speelt dit probleem al bij een vergroting tot 110%. In een venster van 360 px breed, gaat het goed bij een vergroting tot 110%, maar begint dit te spelen bij een vergroting tot 120%.

Hoe breder het browservenster wordt, hoe minder dit probleem speelt. Vanaf een breedte van ongeveer 620 px zijn alle hoofdsteden bij een lettergrootte van 200% weer volledig zichtbaar.

Op de afbeelding zijn de letters vergroot tot 200% in een browservenster met een breedte van 400 px. De groene achtergrondkleur van de provincienaam bij het juist invullen van de provinciehoofdstad werkt wel gewoon, dus er is wel te zien dat goed is ingevuld.

Als het apparaat in landschapmode wordt

gehouden, zijn de provinciehoofdsteden wel volledig zichtbaar.

## OVERIGE PROBLEMEN

Probleem: Samsung Internet op Android 6 geeft geen tellers en goed- en foutmeldingen weer. De tekstvelden staan niet op de goede plaats.

Samsung Internet op Android 6 wordt niet meer geüpdatet. Hierdoor wordt `:has()` niet ondersteund. `:has()` wordt hier gebruikt om de tellers en de goed- en foutmeldingen te regelen. Daardoor worden in deze browser geen tellers en goed- en foutmeldingen weergegeven. De provinciehoofdsteden kunnen gewoon worden ingevuld, maar er wordt niets gecontroleerd.

Ook zijn de tekstvelden te klein en staan ze niet onder, maar achter de provincienaam.

Omdat deze versie van Android nauwelijks nog wordt gebruikt, en omdat dit eigenlijk niet is op te lossen, is dit zo gelaten.

Probleem: Safari, iOS en iPadOS ouder dan versie 15.4 geven geen tellers en goed- en foutmeldingen weer.

Safari, iOS en iPadOS ondersteunen pas vanaf versie 15.4 `:has()`. `:has()` wordt hier gebruikt om de tellers en de goed- en foutmeldingen te regelen. Daardoor worden in deze browser en systemen geen tellers en goed- en foutmeldingen weergegeven. De provinciehoofdsteden kunnen gewoon worden ingevuld, maar er wordt niets gecontroleerd.

Omdat op iOS en iPadOS alle browsers de weergave-machine van Safari gebruiken, geldt dit op iOS en iPadOS ook voor Google Chrome, Edge en Firefox.

Probleem: Safari, iOS en iPadOS ouder dan versie 16.5, Google Chrome op Android 6 en Samsung Internet op Android 7 geven lege en verkeerd ingevulde velden in een gecombineerde teller weer. De meldingen 'Fout!' en 'Bravo! Alle twaalf goed!' worden niet weergegeven.

Google Chrome op Android 6 en Samsung Internet op Android 7 worden niet meer geüpdatet. Hierdoor worden `:user-valid` en `:user-invalid` niet ondersteund. Safari, iOS en iPadOS ondersteunen dit pas in versie 16.5.

Daarom wordt in deze browsers en systemen hiervoor in de plaats `:valid` en `:invalid` gebruikt. Deze pseudo-classes controleren echter de invoer gelijk bij openen van de pagina, voordat er ook maar iets is ingevoerd. Daardoor zie je gelijk bij openen van de pagina twaalf keer 'Fout!' verschijnen. Ook de melding 'Bravo! Alle twaalf goed' wordt niet weergegeven. Als een provinciehoofdstad goed wordt ingevuld, werkt alles wel zoals het hoort te werken.

Om dit te voorkomen wordt de melding 'Fout!' niet weergegeven in deze browsers en systemen. Omdat op iOS en iPadOS alle browsers de weergave-machine van Safari gebruiken, geldt dit op iOS en iPadOS ook voor Google Chrome, Edge en Firefox.

Probleem: de validator geeft een foutmelding voor `@supports selector()`

Bij valideren van de css op de [css-validator](#) van W3C krijg je een hele serie foutmeldingen, omdat `@supports selector()` onbekend is. Deze fout kan genegeerd worden. Dit is relatief nieuw en de validator loopt soms wat achter.

## Wijzigingen

Alleen grotere wijzigingen worden hier vermeld, geen dingen als een link die is geüpdatet.  
29 maart 2025:  
Nieuw opgenomen.

## Inhoud van de download en licenties

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

- \* Sommige onderdelen die van 'n andere site of zo afkomstig zijn, vallen mogelijk onder een of andere licentie. Dat is hieronder bij het betreffende onderdeel te vinden.
- \* Je gebruikt het materiaal uit deze download volledig op eigen risico. Het kan prima zijn dat er fouten in de hier verstrekte code e.d. zitten. Voor eventuele schade die door gebruik van materiaal uit deze download ontstaat, in welke vorm dan ook, zijn [www.css-voorbeelden.nl](http://www.css-voorbeelden.nl) en medewerkers daarvan op geen enkele manier verantwoordelijk.
- \* Dit voorbeeld (en de bijbehorende uitleg e.d.) wordt min of meer regelmatig bijgewerkt. Het is daarom niet toegestaan dit voorbeeld (en de bijbehorende uitleg e.d.) op welke manier dan ook te verspreiden, zonder daarbij duidelijk te vermelden dat voorbeeld, uitleg, e.d. afkomstig zijn van [www.css-voorbeelden.nl](http://www.css-voorbeelden.nl) en dat daar altijd de nieuwste versie is te vinden. Dit is om te voorkomen dat er verouderde versies worden verspreid. Een link naar [www.css-voorbeelden.nl](http://www.css-voorbeelden.nl) wordt trouwens altijd op prijs gesteld.

tekst-009-dl.html: de pagina met het voorbeeld.

tekst-009.pdf: deze uitleg (aangepast aan de inhoud van de download).

tekst-009-inhoud-download-en-licenties.txt: een kopie van de tekst onder dit kopje  
(Inhoud van de download en licenties).

009-css-dl:

tekst-009-dl.css: stijlbestand voor tekst-009-dl.html.

009-pics:

provincies-nederland.jpg in drie groottes.

De plattegrond is afkomstig van

[commons.wikimedia.org/wiki/File:Netherlands\\_location\\_map.svg](https://commons.wikimedia.org/wiki/File:Netherlands_location_map.svg) en valt onder de Creative Commons Attribution-Share Alike 3.0 Unported licentie. ([creativecommons.org/licenses/by-sa/3.0/deed.en](https://creativecommons.org/licenses/by-sa/3.0/deed.en))

## 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 **vet blauw**. 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](http://www.css-voorbeelden.nl).)

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

### <!doctype html>

Een document moet met een doctype beginnen om weergaveverschillen tussen browsers te voorkomen. Zonder doctype is de kans op verschillende (en soms volkomen verkeerde) weergave tussen verschillende browsers heel erg groot.

Geldige doctypes vind je op [www.w3.org/QA/2002-04/valid-dtd-list](http://www.w3.org/QA/2002-04/valid-dtd-list).

Gebruik het volledige doctype, inclusief de eventuele url, anders werkt het niet goed.

Het hier gebruikte doctype is dat van html5. Dit kan zonder enig probleem worden gebruikt: het werkt zelfs in Internet Explorer 6.

**<html lang="nl">**

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

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

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

utf-8 is de beste charset (tekenset), omdat deze alle talen van de wereld (en nog heel veel andere extra tekens) bestrijkt, maar toch niet meer ruimte inneemt voor de code, dan nodig is. Als je utf-8 gebruikt, hoef je veel minder entiteiten (`&auml`; e.d.) te gebruiken, maar kun je bijvoorbeeld gewoon `ä` gebruiken.

Deze regel moet zo hoog mogelijk komen te staan, als eerste regel binnen de `<head>`, omdat de regel anders door sommige browsers niet wordt gelezen.

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

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

Mobiele apparaten variëren enorm in grootte. En dat is een probleem. Sites waren, in ieder geval tot enkele jaren geleden, gemaakt voor desktopbrowsers. En die hebben, in vergelijking met bijvoorbeeld een smartphone, heel brede browservensters. Hoe moet je op 'n smartphone een pagina weergeven, die is gemaakt voor de breedte van een desktop? Je kunt natuurlijk wachten tot alle sites zijn omgebouwd voor smartphones, tablets, enz., maar dan moet je waarschijnlijk heel erg lang wachten.

Mobiele browsers gokken erop dat een pagina een bepaalde breedte heeft. Safari voor mobiel bijvoorbeeld gaat ervan uit dat een pagina 980 px breed is. De pagina wordt vervolgens zoveel versmald dat deze 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 deze wil zien.

Dit betekent ook dat bij het openen van de pagina de tekst meestal heel erg klein wordt weergegeven. (Meestal, want niet alle browsers en apparaten doen het op dezelfde manier.)

Niet erg fraai, maar bedenk maar 'ns 'n betere oplossing voor bestaande sites.

Nieuwe sites of pagina's kunnen echter wel rekening houden met de veel kleinere vensters van mobiele apparaten. In dit voorbeeld bijvoorbeeld worden in smallere vensters de positie van de provincienamen en de positie en de grootte van de tekstvelden aangepast, en goed ingevulde provinciehoofdsteden worden onder de kaart van Nederland neergezet.

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 gediensstige 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. 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. Als een iPad in portretstand bijvoorbeeld 768 px breed is, wordt de pagina ook 768 px breed.

Er staat nog een tweede deel in de tag: `initial-scale=1`. Sommige mobiele apparaten zoomen een pagina gelijk in of uit. Ook weer in een poging behulpzaam te zijn. Ook dat is hier niet nodig. Er is ook een instructie om zoomen helemaal onmogelijk te maken, maar die

wordt niet gebruikt. De bezoeker kan zelf nog gewoon zoomen, wat belangrijk is voor mensen die wat slechter zien.

**<link rel="stylesheet" href="009-css-dl/tekst-009-dl.css">**

Dit is een koppeling naar een extern 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 het stijlbestand aanpassen aan de naam en plaats, waar je eigen stijlbestand staat.

Voordeel van een extern stijlbestand is o.a. dat deze geldig is voor alle pagina's, waaraan deze is gelinkt. 'n Verandering in de lay-out hoeft je dan maar in één enkel stijlbestand aan te brengen, in plaats van in elke pagina apart. Op een grotere site kan dit ontzettend veel werk schelen. Bovendien hoeft de browser zo'n extern stijlbestand 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 stijlbestand eigenlijk geen nut, omdat er maar één pagina is die dit stijlbestand gebruikt. In dit geval kun je de css beter in de `<head>` van de html-pagina zelf zetten. Voor de omvang maakt het hier niets uit, want de css wordt hoe dan ook altijd precies één keer gedownload, en nooit vaker. Voor het onderhoud maakt het ook geen verschil, want ook hier hoeft je de css maar op één plaats te wijzigen. Maar het scheelt wel een extra aanroep naar de server, omdat geen apart stijlbestand hoeft te worden gedownload. Dat opnemen in de `<head>` gaat heel simpel: je kopieert gewoon het hele stijlbestand en zet dat bovenin de `<head>`, tussen `<style>` en `</style>`:

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

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

(De reden dat er toch een extern stijlbestand is, terwijl hierboven omstandig wordt beweerd dat dat in dit voorbeeld eigenlijk geen nut heeft: overzichtelijkheid. Nu kun je html en css los van elkaar bekijken.)

**<h1 id="boven" tabindex="-1">Vul de provincie&shy;hoofdsteden in:</h1>**

De `<h1>` is de belangrijkste titel van de pagina.

Het attribuut `tabindex="-1"` is nodig voor [schermlezers](#) VoiceOver en NVDA. Vanaf de onderkant van de pagina wordt gelinkt naar de `<h1>`:

```
<a id="naar boven" href="#boven">Terug naar boven</a>
```

Omdat de `<h1>` een id heeft, zou een link naar de `<h1>` gewoon horen te werken. Maar in VoiceOver en NVDA blijkt deze link alleen te werken, als de `<h1>` een `tabindex="-1"` heeft. Meer over deze `tabindex` is te vinden bij [tabindex="-1"](#).

(Als de browser `:has()` ondersteunt, staat er in plaats van deze link een andere link naar `div#uitslag`. Ook deze `<div>` is `tabindex="-1"` nodig.)

```


Dit ziet er mogelijk wat ingewikkeld uit, maar als je het in stukjes hakt, valt het hopelijk mee. Deze code voorkomt dat in een klein mobieltje een hele grote afbeelding wordt gedownload, die vervolgens moet worden verkleind. Dat kost alleen maar bandbreedte en tijd, terwijl het geen enkel voordeel heeft.

`<img`: gewoon de normale openingstag van een afbeelding.

`srcset`: binnen `srcset` worden, gescheiden door een komma, één of meer afbeeldingen opgegeven, waaruit de browser een keuze kan maken. De browser kan alleen maar een goede keuze maken, als de breedte van de afbeeldingen bekend is. Daarom wordt die in pixels achter de afbeelding gezet. Alleen wordt niet de afkorting 'px' gebruikt, maar 'w':

```
009-pics/provincies-nederland-320.jpg 320w,
009-pics/provincies-nederland-500.jpg 500w,
009-pics/provincies-nederland-700.jpg 700w,
```

Deze drie afbeeldingen zitten in de map '009-pics'. Aan het eind van de naam staat de breedte van de afbeelding. Voor de browsers maakt dat niets uit, want je mag een afbeelding elke naam geven die je maar wilt. Maar op deze manier is ook voor mensen gelijk duidelijk, hoe breed de afbeelding is: 320px, 500 px en 700px.

Achter elke afbeelding staat een spatie met daarachter de breedte van de afbeelding, met daarachter de letter 'w'. De reden dat de letter 'w' wordt gebruikt in plaats van 'px': er waren eerst plannen om desgewenst ook de hoogte van de afbeelding aan te geven met een 'h'. De 'w' staat voor 'width': breedte. Misschien dat de 'h' ooit nog gebruikt gaat worden, vandaar dat de 'w' in gebruik is gebleven. Maar 320w betekent dus precies hetzelfde als 320px.

De browser kan uit een van deze drie afbeeldingen kiezen. De kleinste afbeelding is 12,3 KiB groot, de middelste 22,7 KiB en de grootste 35,1 KiB. Dat scheelt dus nogal. Voor één afbeelding valt het nog mee, maar als je meerdere afbeeldingen op een pagina hebt, kan het verschil heel groot worden.

`sizes`=: de afbeelding kan op drie verschillende breedtes worden weergegeven: 320 px, 500 px en 700 px, afhankelijk van de breedte van het browservenster. Dit geef je op ongeveer dezelfde manier aan als bij een media query, alleen zijn er hier wat minder mogelijkheden dan bij een echte media query, en het heet hier 'media conditie' (in het Engels 'media condition').

```
"(max-width: 320px) 320px, (max-width: 500px) 500px,
  700px"
```

Net als bij een media query staat het geheel achter `sizes` tussen aanhalingstekens. De verschillende voorwaarden waaraan het scherm en/of het browservenster moeten voldoen worden, met de bijbehorende weergavebreedte, gescheiden door een komma.

(max-width: 320px) 320px: tussen de haakjes staat de voorwaarde, waaraan het browservenster moet voldoen: maximaal 320 px breed. Achter de haakjes staat de breedte, waarop de afbeelding moet worden weergegeven: 320 px.

(max-width: 500px) 500px: de tweede mogelijkheid. Tussen de haakjes staat weer de voorwaarde, waaraan het browservenster moet voldoen: maximaal 500 px breed. Bij de eerste media conditie zijn de vensters met een maximumbreedte van 320 px er al uitgehaald, dus deze media conditie geldt voor vensters met een breedte van 321 t/m 500 px.

700px: de derde en laatste breedte. Omdat hier geen voorwaarde voor scherm of browservenster staat, geldt deze breedte voor alle vensters die niet aan de eerdere condities voldoen (hier: maximaal 320 of 500 px breed). Oftewel: in vensters breder dan 500 px wordt de afbeelding op een breedte van 700 px weergegeven.

Dit zou problemen op kunnen leveren, want in een browservenster van bijvoorbeeld 550 px breed wordt de afbeelding met een breedte van 700 px breed gebruikt.

Daardoor zou je heel vaak moeten scrollen om de hele afbeelding te zien. Dit wordt voorkomen door aan de <img> bij [#kaart](#) een maximumbreedte van 100% te geven, waardoor de afbeelding nooit breder wordt dan ouder div#wrapper. Bij [#wrapper](#) wordt gezorgd dat div#wrapper nooit breder wordt dan het browservenster, dus de afbeelding wordt dat ook nooit. Als dat nodig is, wordt de afbeelding verkleind weergegeven, net zoals dat bij een gewone <img> met één afbeelding gebeurt.

src="009-pics/provincies-nederland-500.jpg": deze regel is precies hetzelfde als bij een gewone <img>. Oudere browsers ondersteunen srcset en sizes niet. Die negeren deze twee attributen en gebruiken deze regel.

alt="Kaart van Nederland met provinciegrenzen": het alt-attribuut zoals dat bij afbeeldingen hoort te staan, net zoals bij een gewone <img> >: geeft het eind van de <img> aan.

Wat hierboven is opgegeven is niet meer dan een hint voor de browser. De browser kan hiervan afwijken. Als bijvoorbeeld een verbinding heel traag is, kan toch in een breder browservenster een kleinere afbeelding worden gebruikt.

Het omgekeerde is ook waar: de browser kan voor een grotere afbeelding kiezen dan hierboven wordt gesuggereerd. De hierboven opgegeven breedtes zijn opgegeven in 'css-pixels': de eenheid die van oudsher in css, html, e.d. wordt gebruikt. Een css-pixel heeft een vaste grootte, gebaseerd op oudere beeldschermen. Er passen 96 css-pixels in 1 inch. Tegenwoordig hebben de meeste schermen een (veel) hogere resolutie: meer pixels per inch. In deze hogeresolutieschermen staan de (veel) kleinere pixels (veel) dichter op elkaar, waardoor een (veel) betere weergave mogelijk is. Die fysiek in het scherm aanwezige pixels heten 'schermpixels' (in het Engels 'device-pixels').

Als een hogeresolutiescherm een resolutiedichtheid heeft van twee keer die van een ouder beeldscherm en 1000 css-pixels breed is, zijn er 2000 schermpixels. Bij een resolutiedichtheid die vier keer zo hoog is, zitten er 4000 schermpixels in 1000 css-pixels. Op zo'n hogeresolutiescherm kan de browser toch kiezen voor een grotere afbeelding. Omdat die grotere afbeelding is opgebouwd uit meer beeldpunten, kan die op een hogeresolutiescherm (veel) duidelijker worden weergegeven.

```
<label for="gr">Gronin&shy;gen<span class="sr-fout">Deze stad is  
fout.</span><span class="sr-goed">Deze stad is  
goed.</span><span class="fout"  
aria-hidden="true">Fout!</span></label>
```

De <label> die bij Groningen hoort. Het enige dat bij openen van de pagina van dit hele verhaal zichtbaar is, is het woord 'Groningen'.

<label for="gr">: de <label> hoort bij de input met id="gr".

Gronin&shy;gen: in smallere browservensters is er te weinig ruimte om het woord 'Groningen' op één regel neer te zetten. Daarin wordt 'Groningen' afgebroken op de plaats waar &shy; staat, waarbij op de plaats van &shy; een afbreekstreepje verschijnt. In bredere vensters, waarin 'Groningen' op één regel past, zie je hier niets van.

<span class="sr-fout">Deze stad is fout.</span>

<span class="sr-goed">Deze stad is goed.</span>: Deze twee <span>'s zijn bedoeld voor [schermlezers](#). Schermlezers hebben niets aan de groene verkleuring als een hoofdstad goed is ingevuld. De melding 'Fout!' is ook verwarrend (waarover gelijk hieronder meer).

Deze twee <span>'s zijn bij openen van de pagina bij [span](#) met `display: none`; verborgen. Afhankelijk van of een hoofdstad verkeerd of goed is ingevuld, wordt een van deze twee <span>'s zichtbaar gemaakt. Als de pagina voor de tweede keer wordt doorlopen, lezen schermlezers daardoor 'Deze stad is fout.' of 'Deze stad is goed' voor.

<span class="fout" aria-hidden="true">Fout!</span>: als een hoofdstad verkeerd wordt ingevuld, verschijnt op het scherm de melding 'Fout!'. Deze wordt echter niet gelijk voorgelezen door schermlezers, waardoor dit heel verwarrend is. (Wat er precies misgaat staat bij [Bekende problemen \(en oplossingen\)](#) → [Schermlezers](#).)

Daarom wordt deze melding voor schermlezers met behulp van [aria-hidden](#) verborgen.

```
<input id="gr" type="text" required pattern=" *[Gg]roningen *"  
placeholder="">
```

Bij elke provincie hoort een <input type="text">, waarin de bijbehorende provinciehoofdstad ingevuld moet worden.

`required`: het attribuut geeft aan dat iets ingevuld móét worden. Een <input> zal meestal binnen een <form> staan. Als dat zo is, wordt bij het versturen van het formulier gecontroleerd op fouten. In dat geval zou een foutmelding volgen, als niets wordt ingevuld. Je kunt eventueel ook met JavaScript gelijk na het verlaten van de <input> die controle uitvoeren.

Hier gebeurt die controle niet, dus eigenlijk zou dit attribuut niet nodig moeten zijn. Zonder dit attribuut verschijnt echter in Safari, iOS en iPadOS bij openen van de pagina gelijk de melding 'Bravo! Alle twaalf goed!' bovenin het browservenster. Omdat op iOS en iPadOS alle browsers de weergave-machine van Safari gebruiken, is het ook voor alle andere browsers op iOS en iPadOS nodig.

`pattern=" *[Gg]roningen *"`: achter `pattern` staat een zogenaamde 'reguliere expressie' (in het Engels 'regular expression'). Reguliere expressies kunnen ongelooflijk ingewikkeld zijn, er zijn dikke boeken over volgeschreven. Op de pagina met [links](#) is onder het kopje Gereedschap → Reguliere expressies hierover meer te vinden.

Met behulp van een reguliere expressie kan worden gecontroleerd, of de invoer in het tekstveld correct is. Als dat zo is, worden de pseudo-classes `:valid` en `:user-valid` geldig. Met behulp van css kan dan een melding worden gegeven dat de hoofdstad goed is ingevuld. Als de hoofdstad fout is ingevuld, worden de pseudo-classes `:invalid` en `:user-invalid` geldig en kan met behulp van css een foutmelding worden weergegeven.

Deze reguliere expressie controleert of 'Groningen' juist is ingevuld. Voor de meeste andere hoofdsteden is de reguliere expressie hetzelfde, alleen zijn de letters anders. 'Den Bosch' en 'Den Haag' wijken iets af, waarover iets hieronder meer.

Deze reguliere expressie kan worden opgesplitst in een aantal delen:

**\***: om de spatie aan het begin zichtbaar te maken, hebben spatie en asterisk hier even een lichtblauwe achtergrond gekregen. Een spatie wordt hetzelfde behandeld als een gewone letter of cijfer. Het teken (of de groep tekens) voor een '\*' mag wel of niet (meerdere keren) aanwezig zijn. Er mogen dus geen, één of meerdere spaties voorafgaan aan 'Groningen'. 'Groningen', 'Groningen' en 'Groningen' worden alle drie goedgekeurd. Dit is gedaan omdat het, vooral bij een kleinere letter, moeilijk is te zien of er een spatie aanwezig is of niet.

[Gg]: dit zijn twee gewone letters, wat betekent dat je deze letters in móét voeren.

Maar omdat de twee letters tussen twee teksthaken staan, moet je kiezen uit een van de twee letters. Je moet hier dus een 'G' of een 'g' invoeren. Allebei invoeren mag niet, dat levert een foutmelding op.

'groningen' wordt goedgekeurd, maar ook 'Groningen'. 'Ggroningen' daarentegen levert een foutmelding op.

roningen: ook dit zijn gewone letters, wat weer betekent dat ze precies zo moeten worden ingevoerd. Na eventuele spaties aan het begin en de letter 'G' of 'g' moet dus 'roningen' worden ingevoerd. Wat bij elkaar 'Groningen' of 'groningen' oplevert: de hoofdstad van de provincie Groningen. En dus wordt goedgekeurd.

**\***: om de spatie aan het begin zichtbaar te maken, hebben spatie en asterisk hier even een lichtblauwe achtergrond gekregen. Hiervoor geldt precies hetzelfde als bij de iets hierboven beschreven '\*' aan het begin, maar nu voor mogelijke spaties achter de hoofdstad. 'Groningen' wordt goedgekeurd, maar ook 'groningen', 'Groningen' en 'groningen'.

De reguliere expressies voor de steden Den Haag en Den Bosch wijken iets af, omdat deze twee steden op verschillende manieren geschreven kunnen worden. Hieronder wordt 'Den Haag' beschreven, maar voor Den Bosch geldt precies hetzelfde (met natuurlijk andere letters).

De volledige reguliere expressie voor Den Haag is:

```
" * [Dd]en [Hh]aag * | 's-[Gg]ravenhage * "
```

Deze expressie is in twee delen te splitsen, gescheiden door een '|':

```
* [Dd]en [Hh]aag *
```

(Om de spatie aan het begin zichtbaar te maken heeft de expressie hier even een lichtblauwe achtergrond gekregen.)

en

```
* 's-[Gg]ravenhage *
```

(Om de spatie aan het begin zichtbaar te maken heeft de expressie hier even een lichtblauwe achtergrond gekregen.)

Beide delen moet aan precies dezelfde eisen voldoen als hierboven beschreven is voor Groningen. Het eerste deel mag worden voorafgegaan door geen, een of meer spaties, moet beginnen met een 'D' of 'd'. Daarop moet 'en ' volgen. De spatie achter 'en ' geldt hier als een gewoon teken, dus die spatie móét worden ingevoerd. Maar omdat hier geen sterretje staat, mag hier één spatie staan. Op deze spatie moet een 'H' of 'h' volgen, gevolgd door 'aag en eventueel weer een of meer spaties. 'Den haag' wordt dus goedgekeurd, maar ook 'Den Haag' en ' den haag '. Hetzelfde geldt voor het twee deel: 's-gravenhage', maar ook ' 's-Gravenhage' wordt goedgekeurd. 'S-Gravenhage' wordt afgekeurd, want de 's' moet een kleine letter zijn.

Hier staan dus twee mogelijkheden om de hoofdstad van Zuid-Holland te benoemen. Maar het is natuurlijk niet de bedoeling dat je twee hoofdsteden gaat invoeren. Daarom staat tussen de twee mogelijkheden een '|'. Dat wil zeggen dat je moet kiezen tussen één van deze twee mogelijkheden. Allebei mag niet, en je mag ze ook niet mengen. 'Den 's-Gravenhage' wordt dus, hoe deftig dit ook moge klinken, gewoon fout gerekend.

placeholder="": zolang er niets is ingevoerd in het tekstveld, wordt in het tekstveld als hint de tekst tussen de aanhalingstekens getoond. Normaal genomen zal hier iets als 'adres' staan: vul hier je adres in. Hier is die tekst tussen de aanhalingstekens helemaal leeg. Er wordt dus helemaal niets getoond. Dat maakt voor de computer echter niets uit: een hint is een hint, of die nou leeg is of niet. Ook in de echte wereld komt deze eigenaardigheid voor: minister Faber is, ondanks haar overduidelijke leeghoofdigheid, toch minister.

Omdat er, wat de computer betreft, gewoon een hint is, kan hier met de pseudo-class :placeholder-shown op worden getest. Zolang niets is ingevoerd, wordt de hint getoond en is het tekstveld dus leeg. Zodra iets is ingevuld (of dat nou goed of fout is), wordt de hint niet meer getoond en is het tekstveld dus niet meer leeg. Dit kan worden gebruikt om het aantal lege tekstvelden te tellen.

**<span class="afk" aria-hidden="true">Gr:</span>**

In smallere browservensters is er onvoldoende ruimte om de ingevulde provinciehoofdsteden op de kaart van Nederland te laten staan. Daarom worden de goed ingevulde onder de kaart neergezet, voorafgegaan door de afkorting van de provincie. Hier is die afkorting 'Gr' voor Groningen.

[Schermlezers](#) hebben niets aan die afkorting, daarom wordt die voor schermlezers met behulp van [aria-hidden](#) verborgen.

**<div id="uitslag" tabindex="-1">**

Binnen deze <div> staat het aantal goede, fouten en lege tekstvelden (voor zover de browser dit ondersteunt).

Het attribuut `tabindex="-1"` is nodig voor [schermlezers](#) VoiceOver en NVDA. Vanaf de onderkant van de pagina wordt `div#uitslag`:

```
<a id="naar-uitslag" href="#uitslag">Naar de uitslag</a>
```

(Als de browser `:has()` niet ondersteunt, staat er in plaats van deze link een andere link naar `h1#boven`.)

Omdat `div#uitslag` een id heeft, zou een link naar de <div> gewoon horen te werken. Maar in VoiceOver en NVDA blijkt deze link alleen te werken, als de <div> een `tabindex="-1"` heeft. Meer over deze `tabindex` is te vinden bij [tabindex="-1"](#).

Deze <div> wordt absoluut gepositioneerd aan de bovenkant van het browservenster, maar staat in de html vrijwel onderaan. Dat kan niet anders, omdat de tellers anders niet werken.

Onderaan de pagina staat wel een link naar deze <div>, zodat ook in lagere vensters duidelijk is dat er bovenaan de pagina een uitslag staat.

## 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 **vet blauw**. 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 het stijlbestand 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, krijg je gegarandeerd 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 het stijlbestand 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 stijlbestand, maar dat heeft een simpele reden: deze uitleg is in feite één groot commentaar. Op internet zelf is het goed, als het stijlbestand 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 het stijlbestand uploaden. Inspringingen bijvoorbeeld zijn voor mensen handig, een computer heeft ze niet nodig. Je hebt dan eigenlijk twee stijlbestanden. De uitgebreide versie waarin je dingen uitprobeert, verandert, enz., met commentaar, inspringingen, e.d. Dat is de mensvriendelijke versie. Daarnaast is er dan een stijlbestand dat je op de echte site gebruikt: een gecomprimeerde versie.

Dat comprimeren kun je met de hand doen, maar er bestaan ook hulpmiddelen voor. Op de pagina met [links](#) kun je onder het kopje Gereedschap → Snelheid testen en verbeteren, gzip, CLS, comprimeren (inclusief theorie), en dergelijke links naar sites vinden, waar je bestanden kunt comprimeren.

(Stijlbestanden op deze site zijn niet gecomprimeerd. Omdat het vaak juist om de css gaat, kunnen mensen dan zonder al te veel moeite de css bekijken.)

## css voor alle browsers en vensters

```
/* tekst-009-dl.css */
```

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

### body

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

```
background: #ff9;
```

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 stijlbestand, 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;
```

Standaardmarge verwijderen.

### main

Alle <main>'s. Dat is er maar een, waarbinnen de belangrijkste inhoud van de pagina zit. Hier is dat de hele pagina.

```
counter-reset: goed fout leeg 12;
```

Met behulp van counters kunnen allerlei dingen worden geteld. Zo'n counter wordt aangemaakt met counter-reset. Als er meerdere counters zijn, zoals hier het geval is, worden die gewoon allemaal achter elkaar gezet. Standaard krijgt een counter de waarde 0, maar als je 'n ander getal achter de counter zet, wordt dat getal de beginwaarde.

Voor elk gegeven is een eigen counter nodig

goed: houdt het aantal goed ingevulde provinciehoofdsteden bij.

fout: houdt het aantal verkeerd ingevulde provinciehoofdsteden bij.

leeg: houdt het aantal nog niet ingevulde tekstvelden bij.

Achter goed en fout staat geen waarde, dus die tellers beginnen met 0. Logisch, want als de pagina wordt geladen, is er nog niets goed of fout ingevuld.

Achter leeg staat 12, dus die teller begint met 12. Ook logisch, want bij openen van de pagina zijn er 12 lege tekstvelden.

h1

Alle <h1>'s. Dat is er maar een: de belangrijkste titel.

```
background: white;
```

Achtergrondkleur wit.

```
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 stijlbestand, is er nog niets aan de hand, want dan veranderen achtergrond- en voorgrondkleur geen van beide.

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

```
box-sizing: border-box;
```

Hieronder worden een breedte en een maximumbreedte opgegeven. Standaard worden een border en een padding bij deze breedte en maximumbreedte opgeteld.

Het element wordt daardoor dus breder dan de opgegeven breedte.

Hier komt dat slecht uit. Met de hier opgegeven waarde bij `box-sizing` komen border en padding binnen de opgegeven breedte te staan, waardoor het element niet breder wordt dan de opgegeven breedte en maximumbreedte.

```
width: 700px;
```

Breedte.

```
max-width: 94vw;
```

Hier gelijk boven is een breedte van 700 px opgegeven. In browservensters smaller dan 700 px zou hierdoor horizontaal gescrold moeten worden om alles te zien. In sommige browsers op sommige systemen verschijnt daarbij een horizontale scrollbar. Om dat te voorkomen, wordt hier een maximumbreedte opgegeven.

De eenheid `vw` is gebaseerd op de breedte van het venster van de browser. 1 `vw` is 1% van de breedte van het venster, en 94 `vw` is 94% van de breedte. De <h1> wordt hierdoor nooit breder dan 94% van de breedte van het venster, ongeacht de breedte van het venster.

```
font-size: 1.2em;
```

Standaard heeft een <h1> een wel heel erg grote lettergrootte. Hier wordt die iets verkleind.

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.

```
font-weight: normal;
```

Standaard wordt een <h1> vet weergegeven. Hier wordt dat veranderd in normaal.

```
text-align: center;
```

Tekst horizontaal centreren.

```
margin: 10px auto;
```

Omdat voor onder en links geen waarden zijn opgegeven, krijgen die automatisch dezelfde waarde als boven en rechts. Hier staat dus eigenlijk `10px auto 10px auto` in de volgorde boven – rechts – onder – links.

Boven en onder een marge van 10 px. Links en rechts `auto`, wat hier hetzelfde betekent als evenveel. Hierdoor komt de `<h1>` altijd horizontaal gecentreerd binnen ouder `<main>` te staan. `<main>` is een blok-element en wordt daardoor normaal genomen automatisch even breed als de ouder ervan: `<body>`. Ook `<body>` is weer een blok-element dat dus normaal genomen even breed wordt als ouder `<html>`. Omdat `<html>` het buitenste element is, wordt dit normaal genomen even breed als het venster van de browser.

Hierdoor staat de `<h1>` uiteindelijk altijd horizontaal gecentreerd binnen het venster van de browser, ongeacht de breedte van het venster. En daarmee ook de in de `<h1>` zittende tekst.

Deze manier van horizontaal centreren van een blok-element werkt alleen, als het blok-element een breedte heeft, want anders zou het normaal genomen even breed worden als de ouder ervan. Dat is geregeld, want iets hoger heeft de `<h1>` een breedte en een maximumbreedte gekregen.

```
border: black solid 1px;
```

Zwart randje.

```
padding: 5px;
```

Aan alle kanten wat afstand tussen tekst in en buitenkant van de `<h1>`.

## #wrapper

Het element met `id="wrapper"`. De `<div>` waar, behalve de `<h1>` en de `<div>` met de uitslag, alles in zit.

```
width: 700px;
```

Breedte.

```
max-width: 94vw;
```

Hier gelijk boven is een breedte van 700 px opgegeven. In browservensters smaller dan 700 px zou hierdoor horizontaal gescrold moeten worden om alles te zien. In sommige browsers op sommige systemen verschijnt daarbij een horizontale scrollbar. Om dat te voorkomen, wordt hier een maximumbreedte opgegeven. De eenheid `vw` is gebaseerd op de breedte van het venster van de browser. 1 `vw` is 1% van de breedte van het venster, en 94 `vw` is 94% van de breedte. `div#wrapper` wordt hierdoor nooit breder dan 94% van de breedte van het venster, ongeacht de breedte van het venster.

```
margin: 0 auto;
```

Omdat voor onder en links geen waarden zijn opgegeven, krijgen die automatisch dezelfde waarde als boven en rechts. Hier staat dus eigenlijk `0 auto 0 auto` in de volgorde boven – rechts – onder – links.

Boven en onder geen marge. Links en rechts `auto`, wat hier hetzelfde betekent als evenveel. Hierdoor komt `div#wrapper` altijd horizontaal gecentreerd binnen ouder `<main>` te staan. `<main>` is een blok-element en wordt daardoor normaal genomen automatisch even breed als de ouder ervan: `<body>`. Ook `<body>` is weer een blok-element dat dus normaal genomen even breed wordt als ouder `<html>`. Omdat `<html>` het buitenste element is, wordt dit normaal genomen even breed als het venster van de browser.

Hierdoor staat `div#wrapper` uiteindelijk altijd horizontaal gecentreerd binnen het venster van de browser, ongeacht de breedte van het venster. En daarmee ook alles wat in `div#wrapper` zit.

Deze manier van horizontaal centreren van een blok-element werkt alleen, als het blok-element een breedte heeft, want anders zou het normaal genomen even breed worden als de ouder ervan. Dat is geregeld, want iets hierboven heeft `div#wrapper` een breedte en een maximumbreedte gekregen.

#### **position: relative;**

Om nakomelingen van `div#wrapper` te kunnen positioneren ten opzichte van `div#wrapper`, moet `div#wrapper` zelf een zogenaamd 'containing block' voor die nakomelingen vormen. Dit kan door aan `div#wrapper` bepaalde eigenschappen te geven. Een van die eigenschappen is een relatieve positie. Een relatieve positie heeft verder geen invloed op `div#wrapper` zelf, omdat niets voor top e.d. wordt opgegeven.

### **#kaart**

Het element met `id="kaart"`. De `<img>` waar de kaart van Nederland in zit.

`box-sizing: border-box;`

Hier iets onder wordt een maximumbreedte opgegeven. Standaard wordt een border bij deze maximumbreedte opgeteld. Het element kan daardoor dus breder worden dan de maximumbreedte.

Hier komt dat slecht uit. Met de hier opgegeven waarde bij `box-sizing` komt de border binnen de opgegeven maximumbreedte te staan, waardoor het element niet breder wordt dan de opgegeven maximumbreedte.

#### **max-width: 100%;**

De afbeelding met de kaart van Nederland kan breder zijn dan het venster van de browser. Daardoor zou een deel van de afbeelding buiten het venster kunnen komen te staan. Dat wordt voorkomen door een maximumbreedte voor de afbeelding op te geven.

Een breedte in procenten is normaal genomen ten opzichte van de ouder van het element. Die ouder is hier `div#wrapper`. Bij `#wrapper` is geregeld dat `div#wrapper` nooit breder wordt dan het venster van de browser. Omdat de afbeelding nooit breder wordt dan 100% van de breedte van `div#wrapper`, geldt dat nu ook voor de afbeelding.

`border: black solid 1px;`

Zwart randje.

### **label**

Alle `<label>`'s. In elke `<label>` zit een provincienaam, die zichtbaar is. Daarnaast zit in elke `<label>` nog een aantal `<span>`'s met meldingen die alleen zichtbaar worden, als iets goed of fout is ingevuld in het bij de `<label>` horende tekstveld.

#### **font-size: 0.8em;**

Omdat er in kleinere browservensters weinig ruimte is, wordt de lettergrootte iets verkleind.

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.

**text-align: center;**

Tekst horizontaal centreren.

**position: absolute;**

Om de <label> op een bepaalde plaats neer te kunnen zetten.

Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij [#wrapper](#) een relatieve positie heeft gekregen.

Een <label> is van zichzelf een inline-element, waardoor eigenschappen als breedte niet gebruikt kunnen worden. Door de <label> absoluut te positioneren verandert deze in een soort blok-element, waardoor dit soort eigenschappen wel is te gebruiken.

## input

Alle <input>'s. In elk van de twaalf <input>'s moet een provinciehoofdstad worden ingevoerd.

**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 stijlbestand, 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.

**width: 2em;**

Breedte.

Dit is te smal om een hoofdstad in te kunnen voeren, maar meer ruimte is er niet in deze kleinere browservensters. Bij een grotere breedte zouden de tekstvelden en de namen van de provincies over elkaar heen komen te staan. Om toch een hoofdstad in te kunnen vullen krijgt de <input> bij [#wrapper input:focus](#) een breedte van 9em als deze de [focus](#) heeft.

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.

**height: 1.2em;**

Hoogte.

Iets lager dan de hoogte uit zichzelf zou zijn. Dit is ook weer vanwege de weinige ruimte in kleinere browservensters.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de hoogte mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de

hoogte niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de hoogte wordt gebruikt.

**font-size: 0.8em;**

Omdat er in kleinere browservensters weinig ruimte is, wordt de lettergrootte iets verkleind.

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-left: 3.5em;**

Link een marge van 3,5 `em`.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de marge mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de marge niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de marge wordt gebruikt.

Bij de individuele `<input>`'s wordt deze marge soms aangepast, en soms wordt een rechtermarge gebruikt. Hiernaast worden de individuele `<input>`'s vanaf links of rechts gepositioneerd. Daarbij wordt ook nog een combinatie van de eenheden `px`, `em` en `%`, al dan niet met `calc()`, gebruikt. De posities zijn grotendeels gevonden door uitproberen.

Die plaatsing is zo ingewikkeld, omdat verschillende systemen en browsers kleinere en grotere afwijkingen geven bij gebruik van dezelfde eigenschappen en eenheden. Bovendien moeten de letters tot een grootte van 200% vergroot kunnen worden.

Het zou kunnen dat deze marge links eigenlijk niet nodig is, als bij de individuele `<input>`'s aanpassingen aan `margin-left`, `margin-right`, `left` en/of `top` worden gedaan. Maar daarvoor zou nog een fors aantal uren uitgeprobeerd en getest moeten worden, waarbij in het gunstigste geval iets minder `CSS` nodig zou zijn. Op deze manier werkt alles, dus wordt dit lekker zo gelaten.

`border: black solid 1px;`

Zwart randje.

`border-radius: 5px;`

Ronde hoeken.

**position: absolute;**

Om de `<input>` op een bepaalde plaats neer te kunnen zetten.

Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij [#wrapper](#) een relatieve positie heeft gekregen.

Een `<input>` is van zichzelf een inline-element, waardoor eigenschappen als breedte en hoogte niet gebruikt kunnen worden. Door de `<input>` absoluut te positioneren verandert deze in een soort blok-element, waardoor dit soort eigenschappen wel is te gebruiken.

`transition: width 0.5s, margin 0.5s;`

Iets hierboven zijn een breedte van 2 `em` en een linkermarge van 3,5 `em` opgegeven.

Bij [#wrapper input:focus](#) wordt dit veranderd in `width: 9em;` en `margin: 0;;`

de `<input>` wordt breder en de marge links verdwijnt, als de `<input>` de [focus](#) krijgt.

De eigenschap `transition` zorgt ervoor dat die verandering niet in één keer plaatsvindt, maar geleidelijk. Als `width` van 2 `em` wordt veranderd naar 9 `em`, vindt die verandering geleidelijk aan plaats: geleidelijk van 2 `em` naar 9 `em`. Hetzelfde geldt voor de verandering van de marge links van 3,5 `em` naar 0 `em`.

Er worden twee eigenschappen veranderd: `width` en `margin`. De eigenschappen worden gezamenlijk opgegeven achter `transition`, gescheiden door een komma. Hierdoor kan elk van de twee eigenschappen een eigen vertraging aan het begin, een eigen snelheid en een eigen verloop van de verandering krijgen. Dat geeft de mogelijkheid om de verandering van de breedte anders te laten verlopen dan de verandering van de marge. Hier is dat niet het geval, beide eigenschappen hebben dezelfde snelheid: 0,5 seconde.

(Je kunt de te veranderen eigenschappen, vertraging, e.d. ook volledig apart opgeven met behulp van `transition-property`, `transition-delay`, e.d., maar het is vaak duidelijker om de te veranderen eigenschap, vertraging, e.d. bij elkaar te houden.)

De twee eigenschappen `width` en `margin` apart:

`width 0.5s`: als er maar één tijd is opgegeven, zoals hier 0,5 seconde, geeft die de tijdsduur van de verandering aan: 0,5 seconde (met 'n punt voor de decimalen, want `css` gebruikt de Amerikaanse notatie). De verandering van breedte van 2 em naar 9 em duurt 0,5 seconde.

Deze kleine aanpassing zorgt ervoor dat `<input>` wat geleidelijker groter wordt. Zonder deze aanpassing verandert de breedte abrupt in één keer. Hoe kort deze tijd ook is, het effect is duidelijk merkbaar.

`margin 0.5s`: als er maar één tijd staat, zoals hier 0,5 seconde, geeft die de tijdsduur van de verandering aan: 0,5 seconde. De verandering van de marge links van 3,5 em naar 0 em duurt 0,5 seconde.

Eigenlijk gebeurt hier nog meer met de marge. Hierboven is een `margin-left` opgegeven. Daardoor houden `margin-top`, `margin-bottom` en `margin-right` hun standaardwaarde van 0. Ook al zijn deze drie eigenschappen hier niet gebruikt, ze hebben wel degelijk een waarde.

De eigenschap `margin` die hier bij `transition` wordt gebruikt is een zogenaamde 'shorthand' voor alle vier de marges. Bij sommige `<input>`'s wordt later een `margin-right` of een `margin-bottom` opgegeven. De geleidelijke verandering geldt ook voor die `margin-right` en `margin-bottom`. Deze wordt bij die `<input>`'s veranderd van de standaardwaarde 0 naar de nieuwe waarde.

`transition` brengt een geheel eigen risico met zich mee. In de specificatie staat, welke eigenschappen met behulp van `transition` kunnen veranderen. Je kunt eventueel bij `transition` opgeven, voor welke eigenschappen het geldt. Als je bijvoorbeeld `top` en `left` beide wilt veranderen bij `:checked`, kun je bijvoorbeeld aangeven dat de geleidelijke verandering met behulp van `transition` alleen voor `top` geldt. `left` verandert dan gewoon in één keer.

Als je niet opgeeft, voor welke eigenschappen `transition` geldt, geldt het voor alle eigenschappen die bij `:hover`, `:focus`, `:checked`, e.d. worden veranderd.

Als daar 'n eigenschap bij zit die op dit moment niet door `transition` kan worden vertraagd, verandert die gewoon in één keer. Maar als de specificatie of een van de browsermakers de geest krijgt en plotsklaps die eigenschap ook onder `transition` laat vallen, kan dat tot heel onverwachte resultaten leiden.

Stel dat je een bepaalde eigenschap heel traag wilt veranderen bij `:hover` en een andere eigenschap flitsend snel. Mogelijk wordt dat flitsend snel dan opeens ook heel traag, als `transition` in de toekomst die eigenschap ook gaat ondersteunen.

Daarom moet je absoluut zeker weten dat ook in de toekomst geen problemen

kunnen ontstaan. Bij twijfel: geef aan welke eigenschap(pen) `transition` mag aansturen.

In dit voorbeeld speelt dit risico niet, omdat bij `transition` de afzonderlijke eigenschappen zijn opgegeven, waarvoor `transition` geldt. Daardoor blijft `transition`, wat er eventueel ooit nog gaat veranderen, hoe dan ook alleen voor deze twee eigenschappen gelden.

## #groningen label

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

`label` {font-size: 0.8em; text-align: center; position: absolute;}

De `<label>`'s binnen het element met `id="groningen"`. Dat is er maar één: de `<label>` voor provinciehoofdstad 'Groningen' binnen `div#groningen`.

### **width: 3em;**

Breedte.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de breedte mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de breedte niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de breedte wordt gebruikt.

### **bottom: 80%;**

Bij `label` zijn alle `<label>`'s absoluut gepositioneerd. Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij `#wrapper` relatief is gepositioneerd.

De `<label>` komt op 80% vanaf de onderkant van `div#wrapper` te staan. Op deze hoogte overlapt de `<label>` ook in kleine browservensters geen andere elementen, en kan de erin zittende tekst (hier 'Groningen') toch tot 200% worden vergroot.

De hoogte van `div#wrapper` is afhankelijk van de grootte van het browservenster en kan dus veranderen. Omdat de afstand van de `<label>` tot de onderkant van `div#wrapper` in procenten is opgegeven, verandert die afstand mee met de hoogte van `div#wrapper`. Daardoor staat de `<label>` altijd op de juiste plaats ten opzichte van `div#wrapper`, en daarmee ook ten opzichte van de kaart van Nederland en de bij de `<label>` horende `<input>`.

### **left: 74%;**

Hiervoor geldt hetzelfde als gelijk hierboven bij `bottom`, maar nu ten opzichte van de linkerkant van `div#wrapper`. Ook hier verandert de afstand tot de linkerkant van `div#wrapper` mee met de breedte van `div#wrapper`, omdat ook hier de afstand weer in procenten is opgegeven.

## #groningen input

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

```
input {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em; border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}
```

De `<input>`'s binnen het element met `id="groningen"`. Dat is er maar één: de `<input>` waarin 'Groningen' moet worden ingevoerd. (Je kunt ook 'Sinterklaas' invoeren, maar dat wordt niet goedgekeurd.)

**margin-bottom: 10px; margin-left: calc(3.5em + 30px);**

Bij `input` is aan alle `<input>`'s een marge links gegeven. Omdat voor de marge bovenaan, rechts en onder daar niets is opgegeven, krijgen die de standaardwaarde 0. Die marges worden bij sommige `<input>`'s aangepast, zoals hier het geval is. De plaatsing van de `<input>`'s is gruwelijk ingewikkeld, omdat verschillende systemen en browsers kleinere en grotere afwijkingen geven bij gebruik van dezelfde eigenschappen en eenheden. Bovendien moet de tekst tot een grootte van 200% vergroot kunnen worden. Vandaar dat hier verschillende eenheden worden gebruikt. Het zou kunnen dat dit iets eenvoudiger kan, maar daarvoor zou nog een fors aantal uren uitgetest en getest moeten worden. Waarbij in het gunstigste geval iets minder css nodig zou zijn. Op deze manier werkt alles, dus wordt dit lekker zo gelaten.

Aan de onderkant een marge van 10 px. Omdat px een absolute eenheid is, verandert deze niet mee met een andere lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de marge wordt gebruikt.

De marge links is een combinatie van twee waarden:  $3.5em + 30px$ .

Bij `input` iets hierboven is aan alle `<input>`'s een marge links van 3,5 em gegeven. De eenheid em is op de lettergrootte van het bijbehorende element gebaseerd. Hoe groter de letter, hoe groter de eenheid em is. Dus hoe groter de letter, hoe groter ook de marge links van 3,5 em. Als de letters worden vergroot, blijkt dat niet bij alle `<input>`'s goed te werken. Sommige `<input>`'s komen over andere elementen heen te staan. Bij die `<input>`'s is een correctie nodig. Voor deze `<input>` blijkt dat door uitproberen een vermeerdering van de marge links met 30 px te zijn. Anders dan de em verandert een px niet met de lettergrootte: 30 px is altijd 30 px, ongeacht de lettergrootte.

Dit is een van die dingen die te maken hebben met kleinere of grotere verschillen in weergave tussen verschillende browsers en systemen bij dezelfde waarden en eenheden.

De berekening wordt hier gemaakt met twee verschillende eenheden: em en px. Dat kan niet, eerst moeten alle eenheden worden omgerekend naar dezelfde eenheid.

Daarom rekent de browser de eenheid em om naar de eenheid px.

Bij het schrijven van de code kan dat omrekenen niet, omdat je niet weet hoe groot de lettergrootte is. Je kunt wel een lettergrootte opgeven, maar de bezoeker kan dat aanpassen. Maar op het moment van weergave weet de browser de lettergrootte wel. De berekening wordt dan 3,5 keer die lettergrootte in px, plus 30 px.

Bij `input` hebben alle `<input>`'s een lettergrootte van 0,8 em gekregen. Als de gebruiker de lettergrootte niet heeft veranderd, is de standaardlettergrootte 16 px.  $0,8 \times 16 = 12,8$  px: de lettergrootte van de `<input>` is 12,8 px.

De marge links wordt dan  $3,5 \times 12,8 \text{ px} + 30 \text{ px} = 74,8 \text{ px}$ . Dit wordt door de browser afgerond.

Bij een lettergrootte van twee keer de standaardlettergrootte wordt de marge links  $3,5 \times 25,6 \text{ px} + 30 \text{ px} = 119,6 \text{ px}$ , wat ook weer wordt afgerond.

**bottom: calc(80% - 2em);**

Bij [input](#) zijn alle `<input>`'s absoluut gepositioneerd. Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij [#wrapper](#) relatief is gepositioneerd.

De `<input>` komt op 80% vanaf de onderkant van `div#wrapper` te staan. Op die hoogte staat echter ook al de `<label>` die bij Groningen hoort. Om de `<input>` niet gedeeltelijk over die `<label>` te zetten, wordt de afstand tot de onderkant van `div#wrapper` daarom met 2 em verminderd. Op deze hoogte overlapt de `<input>` ook in kleine browservensters geen andere elementen, en kan de erin zittende tekst toch tot 200% worden vergroot.

Hoe de berekening met `calc()` werkt, staat iets hierboven bij [De berekening wordt hier gemaakt ...](#)

De hoogte van `div#wrapper` is afhankelijk van de grootte van het browservenster en kan dus veranderen. Omdat de afstand van de `<input>` tot de onderkant van `div#wrapper` grotendeels in procenten is opgegeven, verandert die afstand mee met de hoogte van `div#wrapper`. Daardoor staat de `<input>` altijd op de juiste plaats ten opzichte van `div#wrapper`, en daarmee ook ten opzichte van de kaart van Nederland en de bij de `<input>` horende `<label>`.

**left: calc(74% - 3.1em);**

Hiervoor geldt hetzelfde als gelijk hierboven bij `bottom`, maar nu ten opzichte van de linkerkant van `div#wrapper`. Ook hier verandert de afstand tot de linkerkant van `div#wrapper` mee met de breedte van `div#wrapper`, omdat ook hier de afstand weer grotendeels in procenten is opgegeven.

```
#friesland label {width: 3em; right: 28%; bottom: 75%;}
#friesland input {margin-right: 3.5em; margin-left: 0; right:
    calc(28% - 3.1em); bottom: calc(75% - 2em);}
#drenthe label {top: 23%; left: 74%;}
#drenthe input {top: calc(23% + 1.5em); left: calc(74% - 3.1em);}
#overijssel label {width: 3em; bottom: 52%; left: 72%;}
#overijssel input {bottom: calc(52% - 2em); left: calc(72% -
    1.4em); }
#flevoland label {width: 3em; bottom: 56%; left: 48%;}
#flevoland input {bottom: calc(56% - 2em); left: calc(48% -
    3.1em);}
#gelderland label {width: 4em; top: 48%; left: 64%;}
#gelderland input {top: calc(47% + 2.8em); left: calc(64% -
    2.6em);}
#utrecht label {top: 50%; left: 40%;}
#utrecht input {top: calc(50% + 1.6em); left: calc(40% - 3.1em);}
#noord-holland label {width: 4em; right: 52%; bottom: 60%;}
#noord-holland input {margin-right: 3.5em; margin-left: 0; right:
    calc(52% - 2.8em); bottom: calc(60% - 2em);}
#zuid-holland label {width: 4em; right: 60%; bottom: 42%;}
#zuid-holland input {margin-right: 3.5em; margin-left: 0; right:
    calc(60% - 2.6em); bottom: calc(42% - 2em);}
#zeeland label {width: 4em; right: 70%; bottom: 30%;}
```

```
#zeeland input {margin-right: 3.5em; margin-left: 0; right:
    calc(70% - 2.8em); bottom: calc(30% - 2em);}
#noord-brabant label {top: 64%; left: 36%;}
#noord-brabant input {top: calc(64% + 1.6em); left: calc(36% -
    1.4em);}
#limburg label {top: 80%; left: 56%;}
#limburg input {top: calc(80% + 1.6em); left: calc(56% - 2.6em);}
```

Voor deze elementen is eerder css opgegeven. Deze wordt hier niet allemaal herhaald, omdat het nogal veel is. Hier wordt alleen de plaats van de <label>'s en <input>'s aangepast. In een enkel geval wordt ook de breedte van een <label> aangepast, omdat in deze iets bredere browservensters iets meer ruimte is.

Dit zijn de <label>'s en <input>'s die bij de provincies en provinciehoofdsteden horen, behalve die van Groningen, want die zijn hierboven al beschreven. De css werkt precies hetzelfde als die bij [# groningen label](#) en [# groningen input](#) hierboven. De beschrijving is daar te vinden. Het enige verschil met Groningen zijn de gebruikte waarden.

## span

Alle <span>'s. In de <span>'s zitten de goed- en foutmeldingen en de afkortingen van de provincienamen.

**display: none;**

Alle <span>'s verbergen. Pas als een melding of afkorting gebruikt moet worden, wordt de desbetreffende <span> zichtbaar gemaakt.

## .sr-fout, .sr-goed

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

Alle elementen met class="sr-fout" en class="sr-goed". In deze <span>'s zit de melding voor [schermlezers](#) of een hoofdstad fout of goed is ingevuld.

**position: absolute;**

Om de <span>'s op een bepaalde plaats neer te kunnen zetten.

Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat zijn hier de <label>'s, waar de <span>'s in zitten, want deze zijn bij [label](#) relatief gepositioneerd.

**top: -1000px; left: -20000px;**

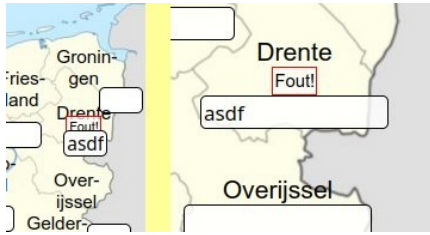
1000 px naar boven en 20.000 px naar links verplaatsen. Hierdoor komen de <span>'s buiten het venster van de browser te staan, waardoor ze niet zichtbaar zijn. Voor een [schermlezer](#) is dat echter geen probleem, deze leest de in de <span>'s zittende tekst gewoon voor.

Er wordt ook naar boven verplaatst, want bij alleen naar links verplaatsen werken in sommige schermlezers links niet altijd goed, als die toevallig op dezelfde hoogte als de <span>'s staan.

## .fout

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

```
span {display: none;}
```



Alle elementen met `class="fout"`. In deze `<span>`'s zit de foutmelding die verschijnt, als een hoofdstad fout wordt ingevuld.

Op de afbeelding is de hoofdstad van Drenthe fout ingevuld. Links de afbeelding in een klein browservenster, rechts in een groter venster.

## `background: rgb(255 255 255 / 0.7);`

Vaak wordt voor 'n kleur de hexadecimale notatie gebruikt, iets als `color:`

`#33ab01;`. Daarbij worden niet alleen cijfers, maar ook letters gebruikt. 0 t/m 9

werken precies hetzelfde als altijd, maar na de 9 komen nog A, B, C, D, E en F.

Als je telt, is 't dus: 0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12, enz. Daarbij is A gelijk aan het tientallige 10, B aan 11, enz. Op deze manier kun je met twee cijfers en/of letters 256 mogelijkheden aangeven: van 00 t/m FF.

De eerste drie getallen bij `rgb()` geven de kleur aan. Deze lopen van 0 t/m 255, dus ook hiermee kun je 256 mogelijkheden aangeven. En omdat er drie getallen zijn levert dat  $256 \times 256 \times 256 = 16.777.216$  mogelijke kleuren op, net iets meer dan het aantal kleurpotloden in de gemiddelde kleurdoos van 'n kind.

De notatie bij `rgb()` geeft dus precies evenveel mogelijkheden als de hexadecimale.

Het eerste getal staat voor rood, het tweede voor groen en het derde voor blauw (feitelijk de Engelse namen, maar de eerste letter is toevallig in het Nederlands hetzelfde). Uit deze drie kleuren zijn alle kleuren op een monitor opgebouwd.

255 wil zeggen volledig aanwezig, 0 wil zeggen helemaal ontbrekend.

`rgb(255 255 255)` levert wit op, `rgb(0 0 0)` zwart.

In plaats van getallen mag je ook percentages gebruiken, bijvoorbeeld: `rgb(10% 20% 100%)`.

Omdat in dit voorbeeld drie keer 255 is opgegeven, levert dit een witte kleur op.

Als de kleur doorsichtig moet zijn, is nog een vierde getal aanwezig. Dit wordt van de eerste drie getallen gescheiden door een schuine streep /. Dit vierde getal staat voor het alfa-kanaal. Hiermee wordt de doorsichtigheid aangegeven. Dit getal loopt van 0 naar 1. Volledig doorsichtig is 0, volledig ondoorsichtig is 1. Het getal voor het alfa-kanaal wordt als decimale breuk aangegeven, dus bijvoorbeeld 0.5 wil zeggen halfdoorsichtig. Let erop dat je 'n punt gebruikt, de Amerikaanse manier om breuken aan te geven. Als je 'n komma gebruikt, denkt de browser dat er twee verschillende getallen staan. Je kunt de doorsichtigheid, net als bij de eerste drie getallen voor de kleuren, ook als een

### Oudere notatie

In eerdere specificaties werden de getallen bij `rgb()` gescheiden door een komma. Wit was dan `rgb(255, 255, 255)` of `rgb(100%, 100%, 100%)`.

Als een kleur doorsichtig moest zijn, werd geen `rgb()` maar `rgba()` gebruikt. Het vierde getal werd dan niet van de drie eerdere gescheiden door een /, maar door een komma: `rgba(255, 255, 255, 0.5)` levert ook hier halfdoorsichtig wit op. En ook hier kon de doorsichtigheid met een percentage worden opgegeven.

Deze oudere notatie werkt nog steeds, en blijft ook werken. De notatie is veranderd om deze in overeenstemming te brengen met andere, nieuwere manieren om kleuren aan te geven.

percentage aangeven: `rgb(100% 100% 100% / 50%)` levert een halfdoorzichtig wit op.

In dit voorbeeld is de achtergrondkleur enigszins doorzichtig: 0.7. Hierdoor zie je grenzen van de provincies nog enigszins door de achtergrond van de foutmelding heen, maar is de foutmelding zelf toch goed zichtbaar.

**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 stijlbestand, is er nog niets aan de hand, want dan veranderen achtergrond- en voorgrondkleur geen van beide.

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

**width: 4ch;**

Breedte.

1 ch is de breedte van het getal '0' van de gebruikte lettersoort. Deze waarde blijkt in alle browsers het best te werken.

**font-size: 0.7em;**

Tamelijk kleine letter.

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. Deze `<span>` zit in een `<label>`, die bij [label](#) ook al een kleinere lettergrootte van 0,8 em heeft gekregen. Omdat de lettergrootte wordt geërfd door de nakomelingen van een element, wordt de uiteindelijke lettergrootte van de foutmelding – als de bezoeker de lettergrootte niet heeft veranderd –  $0,7 \times 0,8 \text{ em} = 0,56 \text{ em}$ . Dat is vrij klein, maar de foutmelding heeft een rode rand en staat ietwat hinderlijk deels over de provincienaam heen. Daardoor valt deze toch nog genoeg op.

**margin: 0 auto;**

Omdat voor onder en links geen waarden zijn opgegeven, krijgen deze automatisch dezelfde waarde als boven en rechts. Hier staat dus eigenlijk `0 auto 0 auto` in de volgorde boven – rechts – onder – links.

Boven en onder geen marge, links en rechts `auto`, wat hier hetzelfde betekent als evenveel. Oftewel: de `<span>` staat altijd gecentreerd ten opzichte van de ouder van de `<span>`. Die ouder is hier de `<label>`, waar de `<span>` in zit. In `<label>` zit de naam van de provincie, die bij [label](#) met `text-align: center;` ook gecentreerd is. De `<span>` met 'Fout!' staat dus altijd netjes in het midden van de naam van de provincie.

Deze manier van horizontaal centreren van een blok-element werkt alleen, als het blok-element een breedte heeft, want anders zou het normaal genomen even breed worden als de ouder ervan. Dat is geregeld, want iets hierboven heeft de `<span>` een breedte van 4 ch gekregen.

Maar hier doet zich een eigenaardig verschijnsel voor. De `<span>` wordt iets hieronder absoluut gepositioneerd, en bij een absoluut gepositioneerd element werkt de truc om met `auto` horizontaal te centreren normaal genomen niet. Hier werkt die echter toch.

Iets hieronder wordt met `right: 0;` en `left: 0;` opgegeven dat de `<span>` van de linker- tot de rechterkant van de `<label>` moet lopen. Iets hierboven is echter een breedte van 4 ch opgegeven. Hier wordt een onmogelijke opdracht gegeven.

Als de `<label>` bijvoorbeeld 3 em breed is en de `<span>` moet van `left: 0;` tot `right: 0;` lopen, én mag vanwege de `width: 4ch;` hierboven niet breder dan 4 ch zijn, dan kan aan één van die drie eigenschappen onmogelijk worden voldaan. De browser raakt hier dermate van in de war, dat `right: 0;` en `left: 0;` gewoon worden genegeerd en `margin: 0 auto;` toch werkt. Oftewel: bij een absolute positie werkt `margin: 0 auto;` niet, tenzij je ook een breedte, `left: 0;` en `right: 0;` opgeeft.

(Deze weinig bekende truc werkt trouwens ook bij `position: fixed;`. Je kunt de horizontale plaatsing nog beïnvloeden door bij `right` of `left` een andere waarde dan 0 in te vullen. En het is ook niet zo dat de browser in de war raakt. Dit staat gewoon in de [specificatie](#). Alleen is het zo weerzinwekkend technisch opgeschreven dat naar verluidt zelfs Einstein het pas na 38 keer lezen begreep. Inmiddels is er een nieuwe ontwerp-specificatie met een andere benadering, maar het resultaat is hetzelfde.)

**`border: red solid 1px;`**

Rood randje.

**`padding: 2px;`**

Kleine afstand tussen 'Fout!' en de rand van de `<span>`.

**`position: absolute;`**

Om de `<span>` op een bepaalde plaats neer te kunnen zetten.

Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier de `<label>`, waar de `<span>` in zit, want deze is bij [label](#) absoluut gepositioneerd.

Een `<span>` is van zichzelf een inline-element, waardoor eigenschappen als breedte niet gebruikt kunnen worden. Door de `<span>` absoluut te positioneren verandert deze in een soort blok-element, waardoor dit soort eigenschappen wel is te gebruiken.

**`top: 1em;`**

1 em onder de bovenkant van de `<label>` neerzetten. Hierdoor komt de foutmelding iets onder de naam van de provincie te staan, maar nog wel gedeeltelijk eroverheen. Dit is niet echt ideaal, maar in kleinere browservensters is er niet echt genoeg ruimte om de foutmelding op de perfecte plaats neer te zetten. De provincienaam blijft trouwens nog wel leesbaar, ook al is deze iets afgedekt door de foutmelding.

**`right: 0; left: 0;`**

Het waarom van deze twee eigenschappen staat iets hierboven bij `margin: 0 auto;` beschreven.

## **.afk**

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

`span {display: none;}`

Alle elementen met `class="afk"`. In deze `<span>`'s zit de afkorting van de provincie die onder de kaart van Nederland komt te staan, als een hoofdstad goed wordt ingevuld.

`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 stijlbestand, is er nog niets aan de hand, want dan veranderen achtergrond- en voorgrondkleur geen van beide.

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

**`box-sizing: border-box;`**

Hier iets onder wordt een hoogte opgegeven. Standaard worden een border en een padding bij deze hoogte opgeteld. Het element wordt daardoor dus hoger dan de opgegeven hoogte.

Hier komt dat slecht uit. Met de hier opgegeven waarde bij `box-sizing` komen border en padding binnen de opgegeven hoogte te staan, waardoor het element niet hoger wordt dan de opgegeven hoogte.

**`height: 1.2rem;`**

Hoogte.

Als eenheid wordt de relatieve eenheid `rem` 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. De minder bekende `rem` is ongeveer hetzelfde als de `em`. Alleen is de lettergrootte bij `rem` gebaseerd op de lettergrootte van het `<html>`-element, waardoor de `rem` overal op de pagina precies even groot is, ook als de bezoeker de lettergrootte heeft veranderd.

Deze regelhoogte is dezelfde als die verderop bij `#wrapper input:user-valid` aan de `<input>` met de provinciehoofdstad wordt gegeven. Hierdoor sluiten de afkorting van de provincie en de naam van de provinciehoofdstad precies op elkaar aan, als deze bij goed invullen naast elkaar onder de kaart van Nederland worden gezet.

**`font-size: 0.8em;`**

Tamelijk kleine letter.

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.

Deze `<span>` zit in een `<label>`, die bij [label](#) ook al een kleinere lettergrootte van 0,8 em heeft gekregen. Omdat de lettergrootte wordt geërfd door de nakomelingen van een element, wordt de uiteindelijke lettergrootte van de afkorting – als de bezoeker de lettergrootte niet heeft veranderd –  $0,8 \times 0,8 \text{ em} = 0,64 \text{ em}$ .

`border: black solid;`

Zwart randje. De breedte van de rand wordt gelijk hieronder opgegeven.

`border-width: 1px 0 1px 1px;`

Aan de boven-, onder- en linkerkant een zwart randje. Kleur en stijl zijn gelijk hierboven al opgegeven.

Tegen deze `<span>` met de afkorting aan komt rechts de provinciehoofdstad te staan. In sommige browsers op sommige systemen wordt een randje rechts bij een bepaalde lettergrootte zichtbaar, waardoor er tussen afkorting en provinciehoofdstad een streepje komt te staan. Door de border rechts weg te halen, wordt dat voorkomen.

`padding: 1px 20px 1px 2px;`

Boven, onder en links kleine afstand tussen afkorting en buitenkant van de `<span>`.

Links een grotere padding. Iets hierboven heeft de `<span>` een witte achtergrondkleur gekregen. Die achtergrondkleur loopt door achter de padding.

Rechts van de `<span>` komt de provinciehoofdstad te staan, ook op een witte achtergrond. Door de padding van de `<span>` rechts breder te maken, loopt de achtergrondkleur een eind onder de naam van de provinciehoofdstad door. Hierdoor kan er nooit een kier zonder witte achtergrond tussen afkorting en hoofdstad ontstaan, ongeacht de lettergrootte.

`position: absolute;`

Om de `<span>` op een bepaalde plaats neer te kunnen zetten.

Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij [#wrapper](#) relatief is gepositioneerd.

Een `<span>` is van zichzelf een inline-element, waardoor eigenschappen als hoogte niet gebruikt kunnen worden. Door de `<span>` absoluut te positioneren verandert deze in een soort blok-element, waardoor dit soort eigenschappen wel is te gebruiken.

`left: 0;`

Helemaal links in `div#wrapper` neerzetten. Omdat ook de kaart van Nederland helemaal links in `div#wrapper` zit, worden de afkortingen netjes uitgelijnd met de kaart.

`div:nth-of-type(even) .afk`

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

`span {display: none;}`

`.afk {background: white; color: black; box-sizing: border-box; height: 1.2rem; font-size: 0.8em; border: black solid; border-width: 1px 0 1px 1px; padding: 1px 20px 1px 2px; position: absolute; left: 0;}`

Kinderen van dezelfde ouder kunnen worden geteld, net zoals dat bij kinderen uit een gezin het geval is: eerste kind, tweede kind, derde kind, enz. Je kunt ook alleen 'n bepaald soort element tellen, net zoals je alleen kinderen van jonger of ouder dan zes jaar kunt tellen. Met de pseudo-class `:nth-of-type()`, onderdeel van deze selector, worden alleen elementen van een bepaald soort geteld.

`div: alle <div>'s.`

`:nth-of-type(even)`: het element met een bepaald volgnummer. Het volgnummer staat tussen de haakjes. In dit geval is het 'volgnummer' geen getal, maar het sleutelwoord `even`: alle even `<div>'s`. Omdat vaak alleen de even (of alleen de

oneven) elementen moeten worden geselecteerd, zijn er sleutelwoorden voor even en oneven.

Omdat voor `:nth-of-type (even)` een `div` staat, worden alleen `<div>`'s geteld. `.afk`: de elementen met `class="afk"` binnen zo'n even `<div>`. Binnen elke `<div>` zit één `<span>` met `class="afk"`, waarbinnen de afkorting van de provincie zit.

De hele selector in gewone taal: de `<span>`'s met de afkorting van de provincie binnen elke even `<div>`. Als een provinciehoofdstad goed is ingevuld, komen afkorting en hoofdstad onder de kaart van Nederland te staan. Ze staan daar in twee kolommen: de afkorting en hoofdstad uit elke even `<div>` staan in de linkerkolom, die uit elke oneven `<div>` in de rechterkolom.

**left: 54%;**

Onder de kaart van Nederland staan twee kolommen met afkortingen en hoofdsteden. Deze `<span>`'s moeten in de rechterkolom komen te staan en worden daarom op 54 % vanaf de linkerkant van `div#wrapper` neergezet. 54% is op iets meer dan de helft, zodat er een kleine afstand tussen linker- en rechterkolom is.

**#wrapper div:has(:focus) label**

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

`label` {font-size: 0.8em; text-align: center; position: absolute;}

Naast bovenstaande css is elke `<label>` met behulp van `top`, `right`, `bottom` en/of `left` op de juiste plaats gepositioneerd. Bij langere provincienamen is met behulp van `width` de breedte van de `<label>` beperkt, zodat de naam op twee regels komt te staan. De beschrijving van deze css is te vinden bij [# groningen label](#) en verder.

Oudere browsers ondersteunen `:has()` niet en negeren daarom deze selector en de in deze regel zittende css. Deze browsers geven daardoor geen tellers voor goed, verkeerd of nog niet ingevulde hoofdsteden weer, en geen goed- en foutmeldingen. Dit staat uitgebreider beschreven bij [Overige problemen](#).

`#wrapper`: het element met `id="wrapper"`. De `<div>` waar behalve de titel en de uitslag alles in zit.

`div`: alle `<div>`'s in `div#wrapper`. De twaalf `<div>`'s met de provincies.

`:has()`: maar alleen als de `<div>`'s voldoen aan de voorwaarde die tussen de haakjes staat.

`:has(:focus)`: dit is de voorwaarde, waaraan de `<div>`'s moeten voldoen. Een element binnen de `<div>` moet de [focus](#) hebben. In dit voorbeeld kan alleen de `<input>` binnen de `<div>` de focus krijgen.

`label`: de `<label>`'s binnen de `<div>`'s. In elke `<label>` zitten meldingen, een provincienaam en een afkorting.

De hele selector in gewone taal: doe iets met de `<label>`'s binnen de `<div>`'s die binnen `div#wrapper` zitten, maar alleen als een element binnen die `<div>` de focus heeft.

**background: rgb(255 255 255 / 0.8);**

Iets doorzichtige witte achtergrond. Een uitgebreide beschrijving van deze eigenschap staat bij [background: rgb\(255 255 255 / 0.7\);](#).

**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 stijlbestand, 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.

**font-weight: bold;**

Vette letter. Hierdoor is duidelijker, welke provincienaam bij de `<input>` met de [focus](#) hoort, en welke hoofdstad je dus in moet voeren.

Als de gebruikte lettersoort een vette variant heeft, wordt die gebruikt. De meeste lettersoorten hebben wel een vette variant, soms zelfs meerdere. Een vette variant wordt speciaal ontworpen, het is iets anders dan 'alle lijntjes wat dikker maken'.

Sommige letters kunnen er zelfs heel anders uitzien.

Als de lettersoort geen vette variant heeft, maakt de browser de letter vet. Dat wil zeggen dat de browser gewoon alle lijntjes wat dikker maakt. Dat kan mooi zijn, maar vaak is het foeilelijk. Het is heel iets anders dan een speciaal ontworpen font. Elke rechtgeaarde typograaf gruwet hiervan.

**z-index: 20;**

Normaal genomen worden elementen in de volgorde van de html op het scherm gezet. Wat later in de html staat, wordt over eerdere elementen gezet. Om te zorgen dat de `<label>`, als de bijbehorende `<input>` de [focus](#) heeft, altijd bovenaan staat, krijgt deze een hogere z-index.

Een z-index werkt alleen in bepaalde omstandigheden. Eén van die omstandigheden is een absolute positie. Die is eerder bij [label](#) aan de `<label>`'s gegeven, dus dat is geregeld.

**#wrapper input:focus**

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

```
input {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em; border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}
```

Naast bovenstaande css is elke `<input>` met behulp van `top`, `right`, `bottom`, `left`, `margin-right` en/of `margin-left` op de juiste plaats neergezet. De beschrijving van deze css is te vinden bij [# groningen input](#) en verder.

Alle `<input>`'s binnen het element met `id="wrapper"`, maar alleen als de `<input>` de [focus](#) heeft.

**width: 9em;**

Breedte.

Omdat er in kleinere browservensters zo weinig ruimte is, hebben de `<input>`'s eerder een breedte van maar 2 em gekregen. Dat is veel te smal om de naam van de provinciehoofdstad in te kunnen voeren. Daarom worden ze, als de `<input>` de focus heeft, breed genoeg gemaakt om dat wel te kunnen.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de breedte mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de

breedte niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de breedte wordt gebruikt.

De `<input>`'s hadden een breedte van 2 em en worden hier 9 em breed. Het verschil is 7 em. Omdat er niets aan de positie van de `<input>`'s wordt veranderd, worden de `<input>`'s vanuit het midden aan de linker- en rechterkant evenveel breder: 3,5 em aan beide kanten. Hier gelijk onder wordt de marge van 3,5 em aan de linkerkant verwijderd, omdat de `<input>`'s anders te veel naar links zouden komen te staan. Eerder is bij [input](#) met `transition: width 0.5s, margin 0.5s;` opgegeven dat de verandering van de breedte en de marge geleidelijk aan moet gebeuren: gedurende een halve seconde. Omdat de marge links even breed is als de breedte van de link toeneemt en de verandering van breedte en marge even snel gaat, groeit de breedte van de `<input>` precies even snel naar links als naar rechts.

**margin: 0;**

De eerder opgegeven marge links van 3,5 em wordt verwijderd. Dit gebeurt geleidelijk aan, zoals gelijk hierboven bij `width: 9em;` wordt beschreven.

**border-radius: 0;**

Eerder opgegeven ronde hoeken verwijderen, omdat anders de border in de hoeken te dicht op de tekst staat.

**z-index: 20;**

Normaal genomen worden elementen in de volgorde van de html op het scherm gezet. Wat later in de html staat, wordt over eerdere elementen gezet. Om te zorgen dat de `<input>`, als deze de [focus](#) heeft, altijd bovenaan staat, krijgt deze een hogere z-index.

Een z-index werkt alleen in bepaalde omstandigheden. Eén van die omstandigheden is een absolute positie. Die is eerder bij [input](#) aan de `<input>`'s gegeven, dus dat is geregeld.

## **#wrapper input:user-valid**

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

```
input {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em;
border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}
#wrapper input:focus {width: 9em; margin: 0; border-radius: 0; z-index: 20;}
```

Naast bovenstaande css is elke `<input>` met behulp van `top`, `right`, `bottom`, `left`, `margin-right` en/of `margin-left` op de juiste plaats neergezet. De beschrijving van deze css is te vinden bij [# groningen input](#) en verder.

Oudere browsers ondersteunen `:user-valid` niet en negeren daarom deze selector en de in deze regel zittende css. Voor deze browsers staat bij [#wrapper input:valid](#) een alternatieve selector.

`#wrapper`: het element met `id="wrapper"`. De `<div>` waar behalve de titel en de uitslag alles in zit.

`input`: alle `<input>`'s in `div#wrapper`.

`:user-valid`: maar alleen als in de `<input>` de bij de provincie horende hoofdstad foutloos is ingevuld. De controle op fouten gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij [<input id="gr"...](#)

De hele selector in gewone taal: doe iets met de `<input>` in `div#wrapper`, maar alleen als de juiste hoofdstad is ingevuld. Anders dan `:valid`, dat al gelijk bij openen van de pagina werkt, werkt de pseudo-class `:user-valid` pas nádat de bezoeker iets heeft ingevoerd in de `<input>`.

```
width: calc(46% - 1.5rem);
```

Breedte.

De 46% van de breedte is normaal genomen ten opzichte van de ouder van het element. Maar omdat de `<input>`'s bij [input](#) absoluut zijn gepositioneerd, is dat hier anders. Bij een absoluut gepositioneerd element is een breedte in procenten ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Hier is dat `div#wrapper`, die bij [#wrapper](#) relatief is gepositioneerd.

Vervolgens wordt met behulp van `calc()` van deze breedte 1,5 rem afgetrokken.

Door de relatieve eenheid `rem` te gebruiken, verandert de breedte enigszins mee met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` zou de breedte niet mee veranderen met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid wordt gebruikt.

De minder bekende `rem` is ongeveer hetzelfde als de `em`. Alleen is de lettergrootte bij `rem` gebaseerd op de lettergrootte van het `<html>`-element, waardoor de `rem` overal op de pagina precies even groot is, ook als de bezoeker de lettergrootte heeft veranderd. Bij de `em` kan de lettergrootte worden beïnvloed door de voorouders van het element, bij de `rem` niet.

Hoe de berekening met `calc()` werkt, staat beschreven bij [De berekening wordt hier gemaakt ...](#)

```
height: 1.2rem;
```

Hoogte.

Als eenheid wordt de relatieve eenheid `rem` gebruikt, omdat bij gebruik van een absolute eenheid zoals `px` de hoogte niet mee verandert met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid wordt gebruikt.

De minder bekende `rem` is ongeveer hetzelfde als de `em`. Alleen is de lettergrootte bij `rem` gebaseerd op de lettergrootte van het `<html>`-element, waardoor de `rem` overal op de pagina precies even groot is, ook als de bezoeker de lettergrootte heeft veranderd. Bij de `em` kan de lettergrootte worden beïnvloed door de voorouders van het element, bij de `rem` niet.

Deze regelhoogte is dezelfde als die eerder bij [.afk](#) aan de `<span>` met de afkorting is gegeven. Als bij goed invullen de afkorting van de provincie en de naam van de provinciehoofdstad naast elkaar onder de kaart van Nederland worden gezet, sluiten deze hierdoor goed op elkaar aan.

```
margin: 0;
```

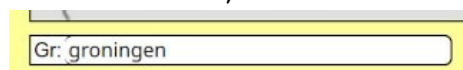
Eerder opgegeven marges verwijderen.

```
border-left-width: 0;
```

Bij [input](#) heeft de `<input>` een zwart randje gekregen.

Tegen deze `<input>` met de provinciehoofdstad aan komt links ervan de afkorting van de provincie te staan. In sommige browsers op sommige systemen wordt een randje links bij een bepaalde lettergrootte zichtbaar, waardoor er tussen afkorting en provinciehoofdstad een streepje komt te staan. Door de border links weg te halen, wordt dat voorkomen.

```
border-radius: 0;
```



Als de ronde hoeken bij de `<input>` niet worden weggehaald, zie je tussen de afkorting en de naam van de hoofdstad kleine ronde streepjes: de helft van de eerder opgegeven ronde hoek (op de afbeelding tussen de dubbele punt en de eerste 'g'). Daarom worden de ronde hoeken weggehaald.

**left: 1.8em;**

Op 1,8 em vanaf de linkerkant van `div#wrapper` neerzetten. Helemaal links staat al de bijbehorende `<span>` met de afkorting van de provincie. Die heeft bij [.afk](#) rechts een brede padding gekregen. Als de `<input>` op deze afstand vanaf links wordt neergezet, staat die daardoor altijd boven de witte achtergrond van de `<span>` met de afkorting.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de afstand mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de afstand niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de afstand wordt gebruikt.

**z-index: 10;**

Normaal genomen worden elementen in de volgorde van de html op het scherm gezet. Wat later in de html staat, wordt over eerdere elementen gezet. De `<span>`'s met de afkorting staan in de html na de `<input>`'s, waardoor deze boven de `<input>`'s met de provincienaam zouden komen te staan. Bij een standaardlettergrootte is dat niet zichtbaar, maar wel als de lettergrootte wordt verhoogd: dan kan het begin van de naam van de hoofdstad verdwijnen achter de witte achtergrond van de afkorting. Om te zorgen dat de `<input>`'s met de naam van de hoofdstad altijd bovenaan staan, krijgen deze een hogere `z-index`.

Een `z-index` werkt alleen in bepaalde omstandigheden. Eén van die omstandigheden is een absolute positie. Die hebben de `<input>`'s eerder bij [input](#) gekregen, dus dat is geregeld.

```
transition-property: width, border-left, top, right, bottom, left;
```

```
transition-duration: 2s, 0s, 2s, 2s, 2s, 2s;
```

```
transition-timing-function: ease-in;
```

```
transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;
```

Als de hoofdstad goed is ingevuld, wordt de `<input>` onder de kaart van Nederland gezet. Daarbij veranderen de breedte, het randje links en de positie: de eigenschappen `width`, `border-left`, `top`, `right`, `bottom` en `left` veranderen.

Als je bij `:hover`, `:focus`, `:checked`, `:user-valid` e.d. een andere waarde bij een eigenschap opgeeft, kun je er met `transition` bij veel eigenschappen voor zorgen dat de oude waarde niet in één keer in de nieuwe waarde wordt veranderd, maar geleidelijk, gedurende een op te geven tijdsduur.

Een paar van deze vijf eigenschappen zijn hierboven al aangepast. Bij sommige `<input>`'s worden hieronder nog een of meer eigenschappen aangepast. Maar niet bij elke `<input>` worden alle vijf de eigenschappen aangepast. Dat maakt niet uit: als een eigenschap niet wordt aangepast, wordt die gewoon genegeerd.

De vier eigenschappen hierboven die beginnen met `transition` horen bij elkaar.

Je kunt ze ook combineren in een zogenaamde 'shorthand': `transition`.

Daarachter komen dan in een bepaalde volgorde alle waarden die hierboven achter de vier aparte eigenschappen staan. Bij zo'n lange serie veranderingen zijn de aparte eigenschappen vaak duidelijker, maar dat is vooral een kwestie van voorkeur.

```
transition-property: width, border-left, top, right, bottom, left;
```

Dit zijn de vijf eigenschappen waarvoor `transition` geldt, gescheiden door een komma. De andere drie `transition`-eigenschappen hebben alleen invloed op deze vijf eigenschappen.

`transition: duration: 2s, 0s, 2s, 2s, 2s, 2s;`

Hierachter staat de tijdsduur van de verandering. De gebruikte eenheid is 's': seconde. Bij `transition-property` zijn vijf eigenschappen opgegeven, hier worden vijf tijdsduren opgegeven. De volgorde van de opgegeven tijdsduren correspondeert met de volgorde van de opgegeven eigenschappen.

Alle tijdsduren zijn twee seconden, behalve de tweede. De tweede eigenschap achter `transition-property` is `border-left`, dus de tweede tijdsduur hoort bij `border-left`. De rand links verandert in één keer, zonder geleidelijke overgang. De andere vier eigenschappen die met breedte en plaatsing te maken hebben, veranderen gedurende een tijdsduur van twee seconden.

`transition-timing-function: ease-in;`

Hiermee wordt het verloop van de verandering aangegeven: snel aan het begin, snel aan het eind, en dergelijke. `ease-in` laat de verandering enigszins langzaam beginnen en versnelt dan wat. Omdat hier maar één waarde is opgegeven, geldt deze voor alle vijf bij `transition-property` opgegeven eigenschappen.

`transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;`

Hier wordt een eventuele vertraging aan het begin aangegeven, ook weer in seconden. Alle veranderingen beginnen na nul seconden, dus direct. Alleen de vertraging bij de tweede eigenschap is anders: twee seconden. De tweede eigenschap achter `transition-property` is `border-left`, dus de tweede vertraging hoort bij `border-left`. Het randje links verandert pas na twee seconden. Omdat bij `transition-duration` voor de tijdsduur van de verandering van het randje nul seconden is opgegeven, betekent dit dat het randje na twee seconden in één keer verdwijnt. Gelijk met het einde van de veranderingen bij de andere vijf eigenschappen.

De `<input>` houdt dus tijdens de beweging naar beneden aan alle vier de kanten de eerder opgeven rand. Pas als de `<input>` met de naam van de provinciehoofdstad is aangekomen naast de bijbehorende afkorting van de provincie, verdwijnt het randje links. Dat is mooier dan dat het randje eerder al (geleidelijk aan) verdwijnt.

`transition` brengt een geheel eigen risico met zich mee. In de specificatie staat, welke eigenschappen met behulp van `transition` kunnen veranderen. Als je niet opgeeft, voor welke eigenschappen `transition` geldt, geldt het voor alle eigenschappen die bij `:hover`, `:focus`, `:checked`, `:user-valid`, e.d. worden veranderd. Als daar 'n eigenschap bij zit die op dit moment niet door `transition` kan worden vertraagd, verandert die gewoon in één keer. Maar als de specificatie of een van de browsermakers de geest krijgt en plotsklaps die eigenschap ook onder `transition` laat vallen, kan dat tot heel onverwachte resultaten leiden.

Stel dat je een bepaalde eigenschap heel traag wilt veranderen bij `:hover` en een andere eigenschap flitsend snel. Mogelijk wordt dat flitsend snel dan opeens ook heel traag, als `transition` in de toekomst die eigenschap ook gaat ondersteunen.

Daarom moet je absoluut zeker weten dat ook in de toekomst geen problemen kunnen ontstaan. Bij twijfel: geef aan welke eigenschap(en) `transition` mag aansturen.

Hier speelt dat risico niet, omdat bij `transition-property` alle eigenschappen die geleidelijk mogen veranderen zijn opgegeven. Op andere eigenschappen hebben de `transition`-eigenschappen geen enkel effect, of ze nu veranderen of niet.

## #wrapper div:nth-of-type(even) input:user-valid

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

[input](#) {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em; border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}

[#wrapper input:focus](#) {width: 9em; margin: 0; border-radius: 0; z-index: 20;}

[#wrapper input:user-valid](#) {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0; border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width, border-left, top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s; transition-timing-function: ease-in; transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;}

Naast bovenstaande css is elke <input> met behulp van top, right, bottom, left, margin-right en/of margin-left op de juiste plaats neergezet. De beschrijving van deze css is te vinden bij [# groningen input](#) en verder.

Oudere browsers ondersteunen :user-valid niet en negeren daarom deze selector en de in deze regel zittende css. Voor deze browsers staat bij [#wrapper div:nth-of-type\(even\) input:valid](#) een alternatieve selector.

Kinderen van dezelfde ouder kunnen worden geteld, net zoals dat bij kinderen uit een gezin het geval is: eerste kind, tweede kind, derde kind, enz. Je kunt ook alleen 'n bepaald soort element tellen, net zoals je alleen kinderen van jonger of ouder dan zes jaar kunt tellen. Met de pseudo-class :nth-of-type(), onderdeel van deze selector, worden alleen elementen van een bepaald soort geteld.

#wrapper: het element met id="wrapper". De <div> waar behalve de titel en de uitslag alles in zit.

div: alle <div>'s in div#wrapper. De twaalf <div>'s met de provincies.

:nth-of-type(even): het element met een bepaald volgnummer. Het volgnummer staat tussen de haakjes. In dit geval is het 'volgnummer' geen getal, maar een sleutelwoord: even: alle even <div>'s. Omdat vaak alleen de even (of alleen de oneven) elementen moeten worden geselecteerd, zijn er sleutelwoorden voor even en oneven.

Omdat voor :nth-of-type(even) een div staat, worden alleen <div>'s geteld.

Als binnen #wrapper 327 <p>'s zitten, tellen die niet mee. Hadden ze maar 'n <div> moeten zijn.

input: alle <input>'s in zo'n even <div>.

:user-valid: maar alleen als in de <input> de bij de provincie horende hoofdstad foutloos is ingevuld. De controle op fouten gebeurt met behulp van het pattern-attribuut en staat beschreven bij [<input id="gr">...](#)

De hele selector in gewone taal: doe iets met de <input> in elke tweede div binnen div#wrapper, maar alleen als de juiste hoofdstad is ingevuld. Anders dan :valid, dat al gelijk bij openen van de pagina werkt, werkt de pseudo-class :user-valid pas nádat de bezoeker iets heeft ingevoerd in de <input>.

De <input>'s van de goed ingevulde provinciehoofdsteden komen in twee kolommen onder de kaart van Nederland te staan. Deze <input>'s worden hieronder in de rechterkolom geplaatst.

De pseudo-class :nth-of-type() (of de bijzondere vormen :first-of-type en :last-of-type voor het eerste en laatste element uit een serie) kan onverwachte bijwerkingen hebben. In dit geval is er maar één serie <span>'s in het voorbeeld. Maar als in div#wrapper nog andere series <div>'s zouden zitten, zou deze selector ook voor de

even `<div>`'s binnen die series gelden. (Dit zou je in dit geval kunnen oplossen door het toevoegen van `>: #wrapper > div:nth-of-type(even)`. Dan geldt de selector alleen voor directe kinderen van `div#wrapper`.)

**right: 0;**

Helemaal rechts in `div#wrapper` zetten.

**left: auto;**

Bij `#wrapper input:user-valid` is `left: 1.8em;` opgegeven. Bovendien is daar een breedte opgegeven. Hier gelijk boven is `right: 0;` opgegeven. Deze `<input>`'s moeten dus van bijna links in `div#wrapper` naar helemaal rechts lopen, maar ze mogen maar een bepaalde breedte hebben. Bij zo'n onmogelijke opdracht wordt `right` genegeerd. Daarom wordt bij `left` hier de standaardwaarde `auto` opgegeven. Nu werkt `right` wel.

### **#wrapper div:has(:user-valid) label**

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

`label` {font-size: 0.8em; text-align: center; position: absolute;}

`#wrapper div:has(:focus) label` {background: rgb(255 255 255 / 0.8); color: black; font-weight: bold; z-index: 20;}

Naast bovenstaande css is elke `<label>` met behulp van `top`, `right`, `bottom` en/of `left` op de juiste plaats gepositioneerd. Bij langere provincienamen is met behulp van `width` de breedte van de `<label>` beperkt, zodat de naam op twee regels komt te staan. De beschrijving van deze css is te vinden bij [# groningen label](#) en verder.

Oudere browsers ondersteunen `:user-valid` niet en negeren daarom deze selector en de in deze regel zittende css. Voor deze browsers staat bij `#wrapper div:has(:valid) label` een alternatieve selector.

`#wrapper`: het element met `id="wrapper"`. De `<div>` waar behalve de titel en de uitslag alles in zit.

`div`: alle `<div>`'s in `div#wrapper`. De twaalf `<div>`'s met de provincies.

`:has()`: maar alleen als de `<div>`'s voldoen aan de voorwaarde die tussen de haakjes staat.

`:has(:user-valid)`: dit is de voorwaarde, waaraan de `<div>`'s moeten voldoen. Een element binnen de `<div>` moet een juiste invoer hebben. Hier kan alleen de `<input>` binnen de `<div>` die juiste invoer hebben: de bij de provincie horende provinciehoofdstad. De controle op fouten gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij [<input id="gr">...](#)

`label`: de `<label>`'s binnen de `<div>`'s. In elke `<label>` zitten meldingen, een provincienaam en een afkorting.

De hele selector in gewone taal: doe iets met de `<label>`'s binnen de `<div>`'s die binnen `div#wrapper` zitten, maar alleen als een element binnen die `<div>` (de `<input>`) een juiste waarde heeft (de juiste provinciehoofdstad). Anders dan `:valid`, dat al gelijk bij openen van de pagina werkt, werkt de pseudo-class `:user-valid` pas nádat de bezoeker iets heeft ingevoerd in de `<input>`.



Hier worden, als een hoofdstad goed is ingevuld, achtergrondkleur en rand van de provincienaam veranderd. In een kleiner browservenster ziet dat eruit zoals op de afbeelding links. Daarin wordt, bij gebrek aan ruimte, de naam van de hoofdstad bij goed invullen onder de kaart van Nederland gezet. In een groter

venster is er voldoende ruimte om de naam van de hoofdstad op dezelfde plaats te laten staan, zoals op de rechterafbeelding is te zien.

**background: lightgreen;**

Lichtgroene 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 stijlbestand, is er nog niets aan de hand, want dan veranderen achtergrond- en voorgrondkleur geen van beide.

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

**outline: darkgreen solid 2px;**

Donkergroene rand.

**padding: 1px;**

Kleine afstand tussen de naam van de provincie en de buitenkant van de `<label>`.

Eerder was dit niet nodig, omdat de `<label>`'s met de provincienamen geen outline en achtergrondkleur hadden.

**input:user-valid ~ .afk**

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

`span {display: none;}`

`.afk {background: white; color: black; box-sizing: border-box; height: 1.2rem; font-size: 0.8em; border: black solid; border-width: 1px 0 1px 1px; padding: 1px 20px 1px 2px; position: absolute; left: 0;}`

`div:nth-of-type(even) .afk {left: 54%;}`

Oudere browsers ondersteunen `:user-valid` niet en negeren daarom deze selector en de in deze regel zittende css. Voor deze browsers staat bij `input:valid ~ .afk` een alternatieve selector.

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

`:user-valid`: maar alleen als in de `<input>` de bij de provincie horende hoofdstad foutloos is ingevuld. De controle op fouten gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij `<input id="gr"...`

`~`: het hierachter staande element moet ergens in de html staan, na het hiervoor 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: `input` voor de `~` en `span.afk` na de `~` hebben beide als ouder dezelfde `<div>`.

.afk: Alle elementen met class="afk". In deze <span>'s zit de afkorting van de provincie die onder de kaart van Nederland komt te staan, als een hoofdstad goed wordt ingevuld.

De hele selector in gewone taal: doe iets met `span.afk` die volgt op een <input>, maar alleen als in die <input> de juiste hoofdstad is ingevuld. Anders dan `:valid`, dat al gelijk bij openen van de pagina werkt, werkt de pseudo-class `:user-valid` pas nádat de bezoeker iets heeft ingevoerd in de <input>.

**display: block;**

Bij [span](#) zijn de <span>'s met de afkorting van de provincie met `display: none`; verborgen. Hier worden ze zichtbaar gemaakt.

**# groningen input:user-valid, # groningen input:user-valid ~ .afk,  
# friesland input:user-valid, # friesland input:user-valid ~ .afk**

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

```
input {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em;
border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}
# groningen input {margin-bottom: 10px; margin-left: calc(3.5em + 30px); bottom: calc(80% - 2em); left:
calc(74% - 3.1em);}
# friesland input {margin-right: 3.5em; margin-left: 0; right: calc(28% - 3.1em); bottom: calc(75% - 2em);}
.afk {background: white; color: black; box-sizing: border-box; height: 1.2rem; font-size: 0.8em; border: black
solid; border-width: 1px 0 1px 1px; padding: 1px 20px 1px 2px; position: absolute; left: 0;}
div:nth-of-type(even) .afk {left: 54%;}
#wrapper input:focus {width: 9em; margin: 0; border-radius: 0; z-index: 20;}
#wrapper input:user-valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0;
border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width, border-left,
top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s; transition-timing-function: ease-in;
transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;}
#wrapper div:nth-of-type(even) input:user-valid {right: 0; left: auto;}
input:user-valid ~ .afk {display: block;}
```

Oudere browsers ondersteunen `:user-valid` niet en negeren daarom deze selector en de in deze regel zittende css. Voor deze browsers staan bij [# groningen input:valid](#), [# groningen ...](#) alternatieve selectors.

Vier selectors, gescheiden door een komma:

```
# groningen input:user-valid
# groningen input:user-valid ~ .afk
# friesland input:user-valid
# friesland input:user-valid ~ .afk
```

De twee selectors die eindigen op `:user-valid` zijn vrijwel hetzelfde als die bij [# wrapper input:user-valid](#), de beschrijving staat daar. Alleen is `# wrapper` hier veranderd in `# groningen` en `# friesland`. Hierdoor gelden deze selectors niet voor de hele `div#wrapper`, maar alleen voor `div# groningen` en `div# friesland`.

De twee selectors die eindigen op `.afk` zijn vrijwel hetzelfde als die bij [input:user-valid ~ .afk](#), de beschrijving staat daar. Alleen worden de selectors hier voorafgegaan door `# groningen` en `# friesland`, waardoor ze niet voor alle <span>'s met class="afk" gelden, maar alleen voor die in `div# groningen` en `div# friesland`.

Als een hoofdstad goed is ingevuld, worden de bij Groningen en Friesland horende afkorting van de provincie en de provinciehoofdstad onder de kaart van Nederland gezet.

`bottom: 0;`

Bij `input` en `.afk` zijn de `<input>`'s en `<span>`'s met `class="afk"` absoluut gepositioneerd. Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij `#wrapper` relatief is gepositioneerd.

Zoals beschreven bij `#wrapper input:user-valid` veranderen de `<input>`'s met behulp van `transition` geleidelijk van positie. Omdat bij `# groningen input` en `#friesland input` de positie in de hoogte met `bottom` is opgegeven, moet hier ook `bottom` worden gebruikt. Als je hier `top` zou gebruiken, zou de verandering niet geleidelijk verlopen.



In `div#wrapper` zit de kaart van Nederland. De `<input>`'s en `<span>`'s worden tegen de onderkant van `div#wrapper` neergezet. Je zou dus kunnen denken dat `<input>`'s en `<span>`'s over de kaart van Nederland komen te staan. Dat is echter niet zo.

Op de afbeelding heeft `div#wrapper` een rode outline gekregen, zodat de grootte ervan zichtbaar is.

Onder de kaart staat de link 'Naar de uitslag' of (afhankelijk van de ondersteuning door de browser) 'Terug naar boven'. Die link wordt met behulp van `position: relative;` een eind naar beneden verplaatst. Op de afbeelding is die verplaatsing weggehaald en staat de link waar deze zonder relatieve

verplaatsing zou staan.

Als een element met behulp van `position: relative;` wordt verplaatst, blijft dat element de originele ruimte in beslag nemen. Op het scherm staat het element op een andere plaats, maar de originele plaats blijft ook in gebruik.

`div#wrapper` wordt even hoog als de inhoud ervan. Dat is de kaart van Nederland plus de ruimte die de link onder de kaart inneemt, ook al is die link verplaatst en zie je die daar niet meer. Door de `<input>`'s en de `<span>`'s met `class="afk"` met `bottom: 0;` onderaan `div#wrapper` neer te zetten, staan deze op de plek waar op de afbeelding de link staat en staan ze precies onder de kaart van Nederland.

## #drenthe input:user-valid, #drenthe input:user-valid ~ .afk

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

```
input {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em;
border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}
#drenthe input {top: calc(23% + 1.5em); left: calc(74% - 3.1em);}
.afk {background: white; color: black; box-sizing: border-box; height: 1.2rem; font-size: 0.8em; border: black
solid; border-width: 1px 0 1px 1px; padding: 1px 20px 1px 2px; position: absolute; left: 0;}
div:nth-of-type(even) .afk {left: 54%;}
#wrapper input:focus {width: 9em; margin: 0; border-radius: 0; z-index: 20;}
#wrapper input:user-valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0;
border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width, border-left,
top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s; transition-timing-function: ease-in;
transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;}
input:user-valid ~ .afk {display: block;}
```

Oudere browsers ondersteunen :user-valid niet en negeren daarom deze selectors en de in deze regel zittende css. Voor deze browsers staan bij [#drenthe input:valid](#), [#drenthe input:valid ~ .afk](#) alternatieve selectors.

Twee selectors, gescheiden door een komma:

```
#drenthe input:user-valid
#drenthe input:user-valid ~ .afk
```

De selector die eindigt op :user-valid is vrijwel hetzelfde als die bij [#wrapper input:user-valid](#), de beschrijving staat daar. Alleen is #wrapper hier veranderd in #drenthe. Hierdoor geldt deze selector niet voor de hele div#wrapper, maar alleen voor div#drenthe.

De selector die eindigt op .afk is vrijwel hetzelfde als die bij [input:user-valid ~ .afk](#), de beschrijving staat daar. Alleen wordt de selector hier voorafgegaan door #drenthe, waardoor deze niet voor alle <span>'s met class="afk" geldt, maar alleen voor die in div#drenthe.

Als een hoofdstad goed is ingevuld, worden de bij Drenthe horende afkorting van de provincie en de provinciehoofdstad onder de kaart van Nederland gezet.

### top: calc(100% + 0.4em);

Bij [input](#) en [.afk](#) zijn de <input> en de <span> met class="afk" absoluut gepositioneerd. Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier div#wrapper, die bij [#wrapper](#) relatief is gepositioneerd.

Zoals beschreven bij [#wrapper input:user-valid](#) verandert de <input> met behulp van transition geleidelijk van positie. Omdat bij [#drenthe input](#) de positie in de hoogte met top is opgegeven, moet hier ook top worden gebruikt. Als je hier bottom zou gebruiken, zou de verandering niet geleidelijk verlopen.

100% vanaf de bovenkant van div#wrapper neerzetten. Daarmee zouden de <input> en de <span> gelijk tegen de erboven staande <input> en <span> die bij Groningen horen aan komen te staan, zonder enige ruimte ertussen. Daarom wordt bij die 100% nog 0,4 em opgeteld, zodat er wat afstand tussen de <input>'s en <span>'s komt.

Omdat de afstand tot de bovenkant in de relatieve eenheden % en em is opgegeven, verandert de afstand mee met de lettergrootte. Bij een absolute eenheid zoals px zou dat niet het geval zijn.

Hoe de berekening met `calc()` werkt, staat beschreven bij [De berekening wordt hier gemaakt ...](#)

## #overijssel input:user-valid

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

```
input {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em;
border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}
#overijssel input {bottom: calc(52% - 2em); left: calc(72% - 1.4em);}
#wrapper input:focus {width: 9em; margin: 0; border-radius: 0; z-index: 20;}
#wrapper input:user-valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0;
border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width, border-left,
top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s; transition-timing-function: ease-in;
transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;}
#wrapper div:nth-of-type(even) input:user-valid {right: 0; left: auto;}
```

Oudere browsers ondersteunen `:user-valid` niet en negeren daarom deze selector en de in deze regel zittende css. Voor deze browsers staat bij [#overijssel input:valid](#) een alternatieve selector.

Deze selector is vrijwel hetzelfde als die bij [#wrapper input:user-valid](#), de beschrijving staat daar. Alleen is `#wrapper` hier veranderd in `#overijssel`. Hierdoor geldt deze selector niet voor de hele `div#wrapper`, maar alleen voor `div#overijssel`.

Als een hoofdstad goed is ingevuld, worden de bij Overijssel horende afkorting van de provincie en de provinciehoofdstad onder de kaart van Nederland gezet.

### bottom: -2em;

Bij [input](#) is de `<input>` absoluut gepositioneerd. Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij [#wrapper](#) relatief is gepositioneerd.

Zoals beschreven bij [#wrapper input:user-valid](#) verandert de `<input>` met behulp van `transition` geleidelijk van positie. Omdat bij [#overijssel input](#) de positie in de hoogte met `bottom` is opgegeven, moet hier ook `bottom` worden gebruikt. Als je hier `top` zou gebruiken, zou de verandering niet geleidelijk verlopen.

Op een afstand van 2 em onder de onderkant van `div#wrapper` komt de `<input>` op een klein afstand onder de erboven staande `<input>`'s te staan. Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de afstand mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de afstand niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de hoogte wordt gebruikt.

### left: calc(54% + 1.9em);

Bij [input](#) en [afk](#) zijn de `<input>` en de `<span>` met `class="afk"` absoluut gepositioneerd. Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij [#wrapper](#) relatief is gepositioneerd.

Zoals beschreven bij [#wrapper input:user-valid](#) verandert de `<input>` met behulp van `transition` geleidelijk van positie. Omdat bij [#overijssel input](#) de positie in de breedte met `left` is opgegeven, moet hier ook `left` worden gebruikt. Als je hier `right` zou gebruiken, zou de verandering niet geleidelijk verlopen.

54% vanaf de linkerkant van `div#wrapper` neerzetten. Daarmee zou de `<input>` met de provinciehoofdstad op dezelfde plaats als de `<span>` met de afkorting van de

provincie komen te staan. Daarom wordt bij die 100% nog 1,9 em opgeteld, zodat de provinciehoofdstad links naast de afkorting van de provincie komt te staan. Omdat de afstand tot links in de relatieve eenheden % en em is opgegeven, verandert de afstand mee met de lettergrootte. Bij een absolute eenheid zoals px zou dat niet het geval zijn.

Hoe de berekening met `calc()` werkt, staat beschreven bij [De berekening wordt hier gemaakt ...](#)

```
#overijssel input:user-valid ~ .afk {bottom: -2em;}
#flevoland input:user-valid, #flevoland input:user-valid ~ .afk {
    bottom: -3.8em;}
#gelderland input:user-valid {top: calc(100% + 2.4em); left:
    calc(54% + 1.9em);}
#gelderland input:user-valid ~ .afk {top: calc(100% + 2.4em);}
#utrecht input:user-valid, #utrecht input:user-valid ~ .afk {top:
    calc(100% + 4.2em);}
#noord-holland input:user-valid, #noord-holland input:user-valid ~
    .afk {bottom: -5.8em;}
#zuid-holland input:user-valid {right: 54%; bottom: -7.8em; left:
    auto;}
#zuid-holland input:user-valid ~ .afk {bottom: -7.8em;}
#zeeland input:user-valid, #zeeland input:user-valid ~ .afk {
    bottom: -7.8em;}
#noord-brabant input:user-valid {top: calc(100% + 8.4em); left:
    1.9em;}
#noord-brabant input:user-valid ~ .afk {top: calc(100% + 8.4em);}
#limburg input:user-valid {top: calc(100% + 8.4em); left: calc(54%
    + 1.9em);}
#limburg input:user-valid ~ .afk {bottom: -9.9em;}
```

Voor deze elementen is eerder css opgegeven. Deze wordt hier niet allemaal herhaald, omdat het nogal veel is. Hier wordt alleen de plaats van de `span.afk`'s en `<input>`'s aangepast.

Oudere browsers ondersteunen `:user-valid` niet en negeren daarom deze selectors en de in deze regel zittende css. Voor deze browsers staan bij [#flevoland input:valid](#), [#flevoland input:valid ~ .afk](#) en daaronder alternatieve selectors.

In elke `<span class="afk">` zit een afkorting van een provincie. In elke `<input>` zit een bijbehorende provinciehoofdstad. De css werkt precies hetzelfde als die bij [# groningen input:user-valid](#), [# groningen input:user-valid ~ .afk](#), [# friesland input:user-valid](#), [# friesland input:user-valid ~ .afk](#), [#drenthe input:user-valid](#), [#drenthe input:user-valid ~ .afk](#) en [#overijssel input:user-valid](#) hierboven. De beschrijving is daar te vinden. Hier wordt alleen met behulp van `top`, `right`, `bottom` en `left` de positie van de `<span>`'s met `class="afk"` en van de `<input>`'s aangepast.

**#wrapper div:has(:user-invalid):not(:has(:focus)) .fout**

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

span {display: none;}

.fout {background: rgb(255 255 255 / 0.7); color: black; width: 4ch; font-size: 0.7em; margin: 0 auto; border: red solid 1px; padding: 2px; position: absolute; top: 1em; right: 0; left: 0;}

Oudere browsers ondersteunen `:user-invalid` niet en negeren daarom deze selector en de in deze regel zittende css. Daardoor geven deze niet alle tellers en meldingen weer. Meer hierover staat bij [Overige problemen](#).

Een tamelijk lange selector, maar in stukjes gehakt, wordt het overzichtelijker.

`#wrapper`: het element met `id="wrapper"`. De `<div>` waar behalve de titel en de uitslag alles in zit.

`div`: alle `<div>`'s binnen `div#wrapper`. In elk van die `<div>`'s zit een provincie.

`:has()`: maar alleen als de `<div>`'s voldoen aan de voorwaarde die tussen de haakjes staat.

`:has(:user-invalid)`: dit is de voorwaarde, waaraan de `<div>`'s moeten voldoen. Een element binnen de `<div>` moet een onjuiste invoer hebben. Hier kan alleen de `<input>` binnen de `<div>` die onjuiste invoer hebben: een verkeerd ingevoerde provinciehoofdstad. De controle op fouten gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij [<input id="gr"...](#)

`:not()`: maar de `<div>`'s mogen juist níét voldoen aan de voorwaarde die tussen de haakjes achter `:not` staat.

`:not(:has())`: tussen de haakjes achter `:has` staat de voorwaarde, waaraan de `<div>`'s juist níét mogen voldoen.

`:not(:has(:focus))`: dit is de voorwaarde, waaraan de `<div>`'s juist níét mogen voldoen. Geen enkel element binnen de `<div>` mag de [focus](#) hebben. Hier kan alleen de `<input>` binnen de `<div>` de focus hebben, namelijk als er iets kan worden ingevoerd.

`.fout`: de elementen met `class="fout"`. De `<span>` met 'Fout!'.

De selector in gewone taal: doe iets met de `<span>` met 'Fout!' die in elke `<div>` binnen `div#wrapper` zit, maar alleen als er iets verkeerd is ingevoerd binnen de `<div>` en geen enkel element binnen de `<div>` de focus heeft.

**display: block;**

Bij [span](#) zijn de `<span>`'s met 'Fout!' verborgen. Hier worden ze zichtbaar gemaakt. De foutmelding wordt alleen zichtbaar gemaakt, als er iets verkeerd is ingevoerd. Iets invoeren kan alleen maar in de `<input>` die binnen de `<div>` zit. Als die `<input>` de [focus](#) heeft, kun je iets invoeren en wordt de foutmelding verborgen. De reden daarvan: het is wat frustrerend als je een fout probeert te verbeteren, en ondertussen wappert er 'n spandoek met 'Fout!' voor je ogen. Pas als de `<input>` weer is verlaten, niet meer de focus heeft, wordt eventueel de foutmelding weer getoond.

## #wrapper div:has(:user-invalid) .sr-fout

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

[span](#) {display: none;}

[.sr-fout, .sr-goed](#) {position: absolute; top: -1000px; left: -20000px;}

Oudere browsers ondersteunen `:user-invalid` niet en negeren daarom deze selector en de in deze regel zittende css. Daardoor geven deze niet alle tellers en meldingen weer. Meer hierover staat bij [Overige problemen](#).

`#wrapper`: het element met `id="wrapper"`. De `<div>` waar behalve de titel en de uitslag alles in zit.

`div`: alle `<div>`'s binnen `div#wrapper`. In elk van die `<div>`'s zit een provincie.

`:has()`: maar alleen als de `<div>`'s voldoen aan de voorwaarde die tussen de haakjes staat.

`:has(:user-invalid)`: dit is de voorwaarde, waaraan de `<div>`'s moeten voldoen. Een element binnen de `<div>` moet een onjuiste invoer hebben. Hier kan alleen de `<input>` binnen de `<div>` die onjuiste invoer hebben: een verkeerd ingevoerde provinciehoofdstad. De controle op fouten gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij [<input id="gr">...](#)

`.sr-fout`: de elementen met `class="sr-fout"`. De `<span>` met 'Deze stad is fout.', een melding die bij een onjuiste invoer wordt voorgelezen door [schermlezers](#).

De selector in gewone taal: doe iets met de `<span>` met 'Deze stad is fout.' die in elke `<div>` binnen `div#wrapper` zit, maar alleen als er iets verkeerd is ingevoerd binnen de `<div>`.

Een korte beschrijving van `.sr-fout` (en van `.sr-goed`) staat bij [<label](#)

[for="gr">Gronin&shy;gen<span class="sr-fout">Deze...](#)

### **display: block;**

Bij [span](#) zijn de `<span>`'s met 'Deze stad is fout.' verborgen. Hier worden ze zichtbaar gemaakt, als er iets verkeerd is ingevoerd. Of eigenlijk: ze worden niet zichtbaar, want ze zijn bij [.sr-fout, .sr-goed](#) buiten het scherm geparkeerd. Maar schermlezers lezen dat gewoon voor.

## #wrapper div:has(:user-valid) .sr-goed {display: block;}

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

[span](#) {display: none;}

[.sr-fout, .sr-goed](#) {position: absolute; top: -1000px; left: -20000px;}

Oudere browsers ondersteunen `:user-valid` niet en negeren daarom deze selector en de in deze regel zittende css. Daardoor geven deze niet alle tellers en meldingen weer. Meer hierover staat bij [Overige problemen](#).

Dit is de tegenhanger van [#wrapper div:has\(:user-invalid\) .sr-fout](#) gelijk hierboven. De beschrijving staat daar. Het enige verschil: in deze `<span>`'s staat 'Deze stad is goed.' en de melding hoort bij een juist ingevoerd provinciestad.

## #naar-uitslag, #naar-boven

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

`span {display: block;}`

De elementen met `id="naar-uitslag"` en `id="naar-boven"`. De links onderaan de pagina, waarmee je weer naar boven gaat. Welke van deze twee links zichtbaar is, is afhankelijk van de ondersteuning door de browser van `:has()`.

`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 stijlbestand, 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: none;`

Verbergen. Afhankelijk van de ondersteuning van `:has()` door de browser wordt een van de twee later zichtbaar gemaakt.

`box-sizing: border-box;`

Hier iets onder worden een breedte en een maximumbreedte opgegeven. Standaard worden een border en een padding bij deze breedte opgeteld. Het element wordt daardoor dus breder dan de opgegeven breedte en maximumbreedte.

Hier komt dat slecht uit. Met de hier opgegeven waarde bij `box-sizing` komen border en padding binnen de opgegeven breedte en maximumbreedte te staan, waardoor het element niet breder wordt dan de opgegeven breedte en maximumbreedte.

`width: 700px;`

Breedte.

`max-width: 94vw;`

Hier gelijk boven is een breedte van 700 px opgegeven. In browservensters smaller dan 700 px zou hierdoor horizontaal gescrold moeten worden om alles te zien. In sommige browsers op sommige systemen verschijnt daarbij een horizontale scrollbar. Om dat te voorkomen, wordt hier een maximumbreedte opgegeven.

De eenheid `vw` is gebaseerd op de breedte van het venster van de browser. 1 `vw` is 1% van de breedte van het venster, en 94 `vw` is 94% van de breedte. De `<a>`'s worden hierdoor nooit breder dan 94% van de breedte van het venster, ongeacht de breedte van het venster.

`font-size: 0.8em;`

Iets kleinere letter.

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.

```
text-align: center;
```

Tekst horizontaal centreren.

```
margin-bottom: 20em;
```

Marge aan de onderkant. De links zijn het onderste element op de pagina. Door wat ruimte aan de onderkant is duidelijker, dat dit de onderkant van de pagina is.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de marge mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de marge niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de marge wordt gebruikt.

Deze marge is heel erg groot. Iets hieronder wordt de links relatief gepositioneerd en 12,4 `em` naar beneden verplaatst. De marge verplaatst aan de onderkant echter niet mee, die wordt nog steeds gemeten vanaf de oorspronkelijke plaats van de links. In werkelijkheid is deze marge dus slechts 20 `em` – 12,4 `em` = 7,6 `em`.

```
border: black solid 1px;
```

Zwart randje.

```
padding: 3px;
```

Kleine afstand tussen tekst in en buitenkant van de links.

```
position: relative;
```

Door de links een relatieve positie te geven, kunnen deze worden verplaatst. Die verplaatsing is ten opzichte van de link zelf en wordt gelijk hieronder bij `top` opgegeven.

```
top: 12.4em;
```

De links 12,4 `em` naar beneden verplaatsen ten opzichte van zichzelf.

Anders dan bij een absolute positie blijven de links de oorspronkelijke ruimte in beslag nemen, alsof deze niet zijn verplaatst.

De links worden dus een heel eind naar beneden verplaatst. Hierdoor ontstaat er tussen de kaart van Nederland en de links ruimte om daar de afkortingen voor de provincies met de bijbehorende hoofdsteden neer te zetten.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de verplaatsing mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de verplaatsing niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de hoogte wordt gebruikt.

## #naar-boven

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

```
#naar-uitslag, #naar-boven {background: white; color: black; display: none; box-sizing: border-box; width: 700px; max-width: 94vw; font-size: 0.8em; text-align: center; margin-bottom 20em; border: black solid 1px; padding: 3px; position: relative; top: 12.4em;}
```

Het element met `id="naar-boven"`. De link met 'Terug naar boven' die onderaan wordt getoond, als de browser `:has()` niet ondersteunt.

```
display: block;
```

De links met 'Terug naar boven' en 'Naar de uitslag' zijn gelijk hierboven bij [#naar-uitslag](#), [#naar-boven](#) met `display: none;` verborgen. Hier wordt de link met 'Terug naar boven' weer zichtbaar gemaakt.

Als de browser `:has()` wel ondersteunt, wordt deze link later weer verborgen en vervangen door de link met 'Naar de uitslag' getoond.

## #uitslag

Het element met id="uitslag". In deze <div> staan het aantal goed, verkeerd en nog niet ingevulde hoofdsteden, en een melding als alles goed is ingevuld.

## display: none;

Verbergen. De uitslag met de tellers en de melding dat alles goed is ingevuld, worden later in browsers die :has() ondersteunen weer zichtbaar gemaakt.

## css alleen voor browsers die :has() ondersteunen

### @supports selector(:has(\*))

De css die binnen deze 'feature query' staat, geldt alleen voor browsers die :has() ondersteunen. (Een 'feature query' is een vraag, of een bepaalde eigenschap en waarde, of een bepaalde selector, worden ondersteund: iets als 'eigenschap vraag'.)

@supports: hierachter komt tussen haakjes de te onderzoeken selector (of de te onderzoeken eigenschap) met eventueel een bepaalde waarde te staan. selector(): als je, zoals hier gebeurt, niet op een eigenschap met een waarde, maar op een selector wilt testen, gebruik je het sleutelwoord selector(). Tussen de haakjes komt dan de selector te staan, waarvan je de ondersteuning wilt testen. (:has(\*)): de selector, waarvan je de ondersteuning wilt testen. Omdat :has altijd met iets tussen de haakjes wordt gebruikt, wordt tussen de haakjes een sterretje gezet. (Het sterretje staat voor 'alle elementen').

Gelijk na deze regel komt een { te staan, en aan het einde van de css die binnen deze query valt een bijbehorende afsluitende }. Die zijn in de regel hierboven weggefallen, maar het geheel ziet er zo uit:

```
@supports selector(:has(*)) {  
    body {color: silver;}  
    (...) rest van de css voor deze feature query (...)  
    footer {color: gold;}  
}
```

Voor de eerste css binnen deze feature query staat dus een extra {, en aan het eind staat een extra }.

Je kunt ook als voorwaarde opgeven dat meerdere eigenschappen en/of waarden en/of selectors moeten worden ondersteund, of dat een eigenschap of selector juist niet mag worden ondersteund, maar dat gebeurt hier allemaal niet. Hier is de enige voorwaarde dat :has() wordt ondersteund.

Alleen browsers die :has() ondersteunen, voeren de css binnen deze feature query uit. Oudere browsers kennen :has() niet en voeren de css binnen deze feature query daarom niet uit. Als een browser zo oud is dat het hele @supports niet wordt ondersteund, negeert ook die browser alle css binnen de feature query.

Het testen op eigenschappen wordt al langer ondersteund, maar het op selectors testen met behulp van selector() is relatief nieuw. Niet alle browsers die @supports ondersteunen, ondersteunen ook al selector(). Als selector() niet wordt ondersteund, werkt het weer hetzelfde: de css binnen de feature query wordt genegeerd. (Standaard negeren browsers css die ze niet kennen.)

Omdat selector() relatief nieuw is, valideert dit nog niet in de [css-validator](#) van w3c.

Maar omdat nieuwere browsers het allemaal al ondersteunen, is dat geen enkel probleem en kan de foutmelding van de validator gewoon worden genegeerd.

Browsers die `:has()` en/of `@supports selector()` niet ondersteunen, geven geen tellers, geen goed- en geen foutmeldingen weer. Meer hierover bij [Overige problemen](#).

## h1

Deze selector werkt alleen in browsers die `:has()` ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

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

```
h1 {background: white; color: black; box-sizing: border-box; width: 700px; max-width: 94vw; font-size: 1.2em; font-weight: normal; text-align: center; margin: 10px auto; border: black solid 1px; padding: 5px;}
```

Alle `<h1>`'s. Dat is er maar een: de belangrijkste titel.

### **padding-bottom: 2.2em;**

In browsers die `:has()` ondersteunen, komt onder de `<h1>` een regel met tellers te staan. Die regel met tellers wordt absoluut gepositioneerd, gelijk onder de `<h1>`. De simpelste manier om alles naadloos op elkaar aan te laten sluiten is het verhogen van de padding aan de onderkant van de `<h1>`. Omdat de eerder bij de `<h1>` opgeven zwarte rand ook de witte achtergrondkleur en de padding omsluit, kan de regel met de tellers gewoon op de door de padding gecreëerde lege ruimte worden neergezet, boven de witte achtergrond en binnen het zwarte randje van de `<h1>`.

## #uitslag

Deze selector werkt alleen in browsers die `:has()` ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

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

```
#uitslag {display: none;}
```

Het element met `id="uitslag"`. In deze `<div>` staan het aantal goed, verkeerd en nog niet ingevulde hoofdsteden, en een melding als alle hoofdsteden goed zijn ingevuld.

### **display: block;**

Deze css is voor browsers die `:has()` ondersteunen, dus de `<div>` met de tellers wordt zichtbaar gemaakt.

```
width: 700px;
```

Breedte.

```
max-width: 94vw;
```

Hier gelijk boven is een breedte van 700 px opgegeven. In browservensters smaller dan 700 px zou hierdoor horizontaal gescrold moeten worden om alles te zien. In sommige browsers op sommige systemen verschijnt daarbij een horizontale scrollbar. Om dat te voorkomen, wordt hier een maximumbreedte opgegeven. De eenheid `vw` is gebaseerd op de breedte van het venster van de browser. 1 `vw` is 1% van de breedte van het venster, en 94 `vw` is 94% van de breedte. De `div#uitslag` wordt hierdoor nooit breder dan 94% van de breedte van het venster, ongeacht de breedte van het venster.

```
font-size: 0.9em;
```

Iets kleinere letter.

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.

```
text-align: center;
```

Tekst horizontaal centreren.

```
margin: 0 auto;
```

Omdat voor onder en links geen waarden zijn opgegeven, krijgen deze automatisch dezelfde waarde als boven en rechts. Hier staat dus eigenlijk `0 auto 0 auto` in de volgorde boven – rechts – onder – links.

Boven en onder geen marge, links en rechts `auto`, wat hier hetzelfde betekent als evenveel. Oftewel: `div#uitslag` staat altijd gecentreerd ten opzichte van de ouder van de `<div>`. Die ouder is het blok-element `<main>`. Een blok-element wordt normaal genomen automatisch even breed als diens ouder. De ouder van `<main>` is `<body>`, ook weer een blok-element, wat dus normaal genomen ook weer even breed als diens ouder wordt. De ouder van `<body>` is `<html>`. Omdat `<html>` het buitenste element is, wordt dit normaal genomen even breed als het venster van de browser. Hierdoor staat `div#uitslag` automatisch horizontaal gecentreerd in het venster, ongeacht de breedte van het venster.

Deze manier van horizontaal centreren van een blok-element werkt alleen, als het blok-element een breedte heeft, want anders zou het normaal genomen even breed worden als de ouder ervan. Dat is geregeld, want iets hierboven heeft `div#uitslag` een breedte van 700 px en een maximumbreedte van 94 vw gekregen.

Maar hier doet zich een eigenaardig verschijnsel voor. `div#uitslag` wordt iets hieronder absoluut gepositioneerd, en bij een absoluut gepositioneerd element werkt de truc om met `auto` horizontaal te centreren normaal genomen niet. Hier werkt die echter toch.

Iets hieronder wordt met `right: 0;` en `left: 0;` opgegeven dat `div#uitslag` van de linker- tot de rechterkant van het browservenster moet lopen. Iets hierboven zijn echter een breedte van 700px en een maximumbreedte van 94 vw opgegeven. Hier wordt een onmogelijke opdracht gegeven.

Als het browservenster bijvoorbeeld 1000 px breed is en `div#uitslag` moet van `left: 0;` tot `right: 0;` lopen, én mag vanwege de `max-width: 94vw;` niet breder dan 940 px zijn, dan kan aan één van die drie eigenschappen onmogelijk worden voldaan.

De browser raakt hier dermate van in de war, dat `right: 0;` en `left: 0;` gewoon worden genegeerd en `margin: 0 auto;` toch werkt. Oftewel: bij een absolute positie werkt `margin: 0 auto;` niet, tenzij je ook een breedte, `left: 0;` en `right: 0;` opgeeft.

(Deze weinig bekende truc werkt trouwens ook bij `position: fixed;`. Je kunt de horizontale plaatsing nog beïnvloeden door bij `right` of `left` een andere waarde dan 0 in te vullen. En het is ook niet zo dat de browser in de war raakt. Dit staat gewoon in de [specificatie](#). Alleen is het zo weerzinwekkend technisch opgeschreven dat naar verluidt zelfs Einstein het pas na 38 keer lezen begreep. Inmiddels is er een nieuwe ontwerpspecificatie met een andere benadering, maar het resultaat is hetzelfde.)

**`scroll-margin-top: 3.5em;`**



Op de afbeelding heeft `div#uitslag` een blauwe achtergrond en een outline gekregen, zodat de grootte van de `<div>` zichtbaar wordt. Vanuit link `a#naar-uitslag` onderaan de pagina wordt naar deze `<div>` gelinkt. Zonder aanpassing zou `div#uitslag` helemaal bovenaan het browservenster komen te staan, als de link

ernaartoe wordt gevolgd, waardoor de pagina iets naar beneden zou scrollen. Daardoor zou de <h1> met 'Vul de provinciehoofdsteden in:' boven het venster komen te staan en niet meer zichtbaar zijn.

Deze eigenschap zorgt ervoor dat `div#uitslag`, als de link ernaartoe wordt gevolgd, aan de bovenkant een marge van 3,5 em krijgt. Hierdoor komt de <div> 3,5 em lager te staan dan normaal. Nu scrolt de pagina niet omlaag en is de <h1> zichtbaar.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de marge mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de marge niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de hoogte wordt gebruikt.

**position: absolute;**

Om de <div> op een bepaalde plaats neer te kunnen zetten.

Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Als zo'n voorouder er niet is, zoals hier het geval is, wordt gepositioneerd ten opzichte van het venster van de browser.

**top: 3.8em;**

Op 3,8 em vanaf de bovenkant van het browservenster neerzetten.

In de html staat `div#uitslag` helemaal onderaan. Dat kan niet anders, omdat in deze <div> de tellers zitten. En die tellers moeten onder de tekstvelden staan, waar ze bij horen.

Op deze afstand van de bovenkant van het browservenster komt de <div> onder de <h1> met 'Vul de provinciehoofdsteden in:' te staan.

`right: 0; left: 0;`

`div#uitslag` van helemaal links naar helemaal rechts laten lopen. Waarom dit toch niet gebeurt, staat iets hoger bij `margin: 0 auto;`.

## #alles-goed

|                                                                                                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------------|
| Deze selector werkt alleen in browsers die <code>:has()</code> ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang. |
|-------------------------------------------------------------------------------------------------------------------------------------------|

Het element met `id="alles-goed"`. De <p> met de melding die bovenaan verschijnt als alle twaalf hoofdsteden goed zijn ingevuld.

**background: lightgreen;**

Lichtgroene 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 stijlbestand, 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: none;**

Verbergen.

font-size: 1.2em;

Iets grotere letter.

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;**

Standaard heeft een <p> een marge aan boven- en onderkant. Die wordt hier weggehaald, omdat p#uitslag anders iets te laag komt te staan.

border: black solid 1px;

Zwart randje.

padding: 3px;

Kleine afstand tussen tekst in en rand om de <p>.

#met-fout

Deze selector werkt alleen in browsers die :has() ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Het element met id="met-fout". De <p> met de tellers.

margin: 0;

Standaard heeft een <p> een marge aan boven- en onderkant. Die wordt hier weggehaald.

**#naar-uitslag**

Deze selector werkt alleen in browsers die :has() ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

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

[#naar-uitslag](#), [#naar-boven](#) {background: white; color: black; display: none; box-sizing: border-box; width: 700px; max-width: 94vw; font-size: 0.8em; text-align: center; margin-bottom: 20em; border: black solid 1px; padding: 3px; position: relative; top: 12.4em;}

Het element met id="naar-uitslag". De link naar div#uitslag met de tellers bovenaan de pagina.

**display: block;**

De tellers werken alleen in browsers die :has() ondersteunen. Omdat deze css alleen voor die browsers is, wordt de link met 'Naar de uitslag' naar div#uitslag met de tellers getoond.

## #naar-boven

Deze selector werkt alleen in browsers die `:has()` ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

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

```
#naar-uitslag, #naar-boven {background: white; color: black; display: none; box-sizing: border-box; width: 700px; max-width: 94vw; font-size: 0.8em; text-align: center; margin-bottom: 20em; border: black solid 1px; padding: 3px; position: relative; top: 12.4em;}
```

```
naar-boven {display: block;}
```

Het element met `id="naar-boven"`. De link met 'Terug naar boven' die onderaan wordt getoond, als de browser `:has()` niet ondersteunt.

### display: none;

Deze css is voor browsers die `:has()` ondersteunen, dus de link met 'Terug naar boven' wordt verborgen.

## css voor alle browsers en vensters

### #goed, #fout, #leeg

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

```
span {display: none;}
```

De elementen met `id="goed"`, `id="fout"` en `id="leeg"`. De drie `<span>`'s waarachter de tellers komen te staan.

### display: inline;

De `<span>`'s zichtbaar maken. Omdat deze drie `<span>`'s op één regel moeten staan, worden ze als inline-elementen getoond.

## input:user-invalid

Voor deze elementen is eerder css opgegeven. Deze wordt hier niet allemaal herhaald, omdat hier alleen de waarde van de tellers `fout` en `leeg` wordt aangepast.

Oudere browsers ondersteunen `:user-invalid` niet en negeren daarom deze selector en de in deze regel zittende css. Daardoor geven deze niet alle tellers en meldingen weer. Meer hierover staat bij [Overige problemen](#).

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

`:user-invalid`: maar alleen als de ingevoerde tekst niet aan een bepaalde voorwaarde voldoet.

De selector in gewone taal: als iets is ingevoerd dat niet aan een bepaalde voorwaarde voldoet. Hier is die voorwaarde dat de bij de provincie horende provinciehoofdstad foutloos moet zijn ingevoerd. De controle hierop gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij [<input id="gr">...](#)

### counter-increment: fout leeg -1;

Verhoog de bij [main](#) aangemaakte teller 'fout'. Omdat geen waarde wordt opgegeven, wordt met 1 verhoogd. De teller is bij `<main>` geïnitieerd op 0 en staat dus na de eerste verkeerde invoer op 1, na de tweede verkeerde invoer op 2, enz.

Er staat nog een tweede teller: 'leeg', de teller voor de velden waarin nog niets is ingevoerd. Achter deze teller staat wel een waarde: '-1'. Daarom wordt deze teller

met 1 verlaagd. Deze teller is bij `<main>` geïnitieerd op 12, dus na de eerste verkeerde invoer staat deze op 11, na de tweede op 10, enz.

Als een veld weer wordt geleegd, wordt deze teller niet aangepast. Zodra er eenmaal iets is ingevoerd in een veld, blijft dit veld gelden als gewijzigd, ook als dit veld later weer helemaal leeg wordt gemaakt.

### `input:user-valid`

Voor deze elementen is eerder css opgegeven. Deze wordt hier niet allemaal herhaald, omdat hier alleen de waarde van de tellers `goed` en `leeg` wordt aangepast.

Oudere browsers ondersteunen `:user-valid` niet en negeren daarom deze selector en de in deze regel zittende css. Daardoor geven deze niet alle tellers en meldingen weer. Meer hierover staat bij [Overige problemen](#).

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

`:user-valid`: maar alleen als de ingevoerde tekst aan een bepaalde voorwaarde voldoet.

De selector in gewone taal: als iets is ingevoerd dat aan een bepaalde voorwaarde voldoet. Hier is die voorwaarde dat de bij de provincie horende provinciehoofdstad foutloos moet zijn ingevoerd. De controle hierop gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij [<input id="gr">...](#)

### `counter-increment: goed leeg -1;`

Verhoog de bij [main](#) aangemaakte teller 'goed' voor de velden met een juiste invoer. Omdat geen waarde wordt opgegeven, wordt met 1 verhoogd. De teller is geïnitieerd op 0 en staat dus na de eerste juiste invoer op 1, na de tweede juiste invoer op 2, enz.

Er staat nog een tweede teller: 'leeg', de teller voor de velden waarin nog niets is ingevoerd. Achter deze teller staat wel een waarde: '-1'. Daarom wordt deze teller met 1 verlaagd. Deze teller is bij `<main>` geïnitieerd op 12, dus na de eerste juiste invoer staat deze op 11, na de tweede op 10, enz.

Als een veld weer wordt geleegd, wordt deze teller niet aangepast. Zodra er eenmaal iets is ingevoerd in een veld, blijft dit veld gelden als gewijzigd, ook als dit later weer helemaal leeg wordt gemaakt.

### `#goed::after`

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

`span` {display: none;}

`#goed, #fout, #leeg` {display: inline;}

Met behulp van `::after` wordt bij het element met `id="goed"`, de `<span>` met 'Goed: ', een pseudo-element gemaakt. Dit pseudo-element wordt gebruikt om de teller met het aantal goed ingevoerde provinciehoofdsteden weer te geven.

### `content: counter(good);`

De inhoud van `content` komt in het met behulp van `::after` gemaakte pseudo-element te staan, achter de `<span>` met 'Goed: '. Hier is die inhoud de teller met het aantal goed ingevoerde velden.

## css alleen voor browsers die :user-valid níét ondersteunen

### @supports not selector(:user-valid)

De css die binnen deze 'feature query' staat, geldt alleen voor browsers die :user-valid niet ondersteunen. Als een browser :user-valid niet ondersteunt, ondersteunt deze ook de tegenhanger :user-invalid niet. (Een 'feature query' is een vraag of een bepaalde eigenschap en waarde, of een bepaalde selector, worden ondersteund: iets als 'eigenschap vraag'.)

@supports: hierachter komen tussen haakjes de te onderzoeken selector(s) en/of de te onderzoeken eigenschap(en) met eventueel een bepaalde waarde te staan.

not selector(:user-valid):

not: hier wordt gekeken, of iets juist níét wordt ondersteund. Het gebruik van het sleutelwoord not keert het erachter staande selector(:user-valid) om.

selector(): als je, zoals hier gebeurt, niet op een eigenschap met waarde, maar op een selector wilt testen, gebruik je het sleutelwoord selector().

Tussen de haakjes komt dan de selector te staan, waarvan je de ondersteuning wilt testen.

:user-valid: dit is de selector, waarop wordt getest: :user-valid.

not selector(:user-valid) bij elkaar: :user-valid mag niet worden ondersteund.

Gelijk na deze regel komt een { te staan, en aan het einde van de css die binnen deze query valt een bijbehorende afsluitende }. Die zijn in de regel hierboven weggefallen, maar het geheel ziet er zo uit:

```
@supports not selector:user-valid {  
    body {color: silver;}  
    (...) rest van de css voor deze feature query (...)  
    footer {color: gold;}  
}
```

Voor de eerste css binnen deze feature query staat dus een extra {, en aan het eind staat een extra }.

Alleen browsers die :user-valid níét ondersteunen, voeren de css binnen deze feature query uit. Oudere browsers kennen :user-valid niet en voeren de css binnen deze feature query daarom uit. Als een browser zo oud is dat het hele @supports niet wordt ondersteund, negeert ook die browser alle css binnen de feature query.

Het testen op eigenschappen wordt al langer ondersteund, maar het op selectors testen met behulp van selector() is relatief nieuw. Niet alle browsers die @supports ondersteunen, ondersteunen ook al selector(). Ook als dat het geval is, werkt het weer hetzelfde: de css binnen de feature query wordt genegeerd. (Standaard negeren browsers css die ze niet kennen.)

Omdat selector() relatief nieuw is, valideert dit nog niet in de [css-validator](#) van w3c.

Maar omdat nieuwere browsers het allemaal al ondersteunen, is dat geen enkel probleem en kan de foutmelding van de validator gewoon worden genegeerd.

Browsers die :user-valid niet ondersteunen, geven de meldingen 'Fout' en 'Bravo!

Alle twaalf goed!' niet weer. Nog niet ingevulde en verkeerd ingevulde velden worden in een gecombineerde teller weergegeven. Meer hierover bij [Overige problemen](#).

```
#wrapper input:valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0; border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width, border-left, top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s; transition-timing-function: ease-in; transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;}
```

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selector en css zijn vrijwel hetzelfde als die bij [#wrapper input:user-valid](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid en :user-invalid](#).

```
#wrapper div:nth-of-type(even) input:valid {right: 0; left: auto;}
```

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selector en css zijn vrijwel hetzelfde als die bij [#wrapper div:nth-of-type\(even\) input:user-valid](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid en :user-invalid](#).

```
#wrapper div:has(:valid) label {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}
```

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selector en css zijn vrijwel hetzelfde als die bij [#wrapper div:has\(:user-valid\) label](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid en :user-invalid](#).

```
input:valid ~ .afk {display: block;}
```

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selector en css zijn vrijwel hetzelfde als die bij [input:user-valid ~ .afk](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid en :user-invalid](#).

```
# groningen input:valid, # groningen input:valid ~ .afk, # friesland input:valid, # friesland input:valid ~ .afk {bottom: 0;}
```

Deze selectors werken alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selectors en css zijn vrijwel hetzelfde als die bij [# groningen input:user-valid](#), [# groningen input:user-valid ~ .afk](#), [# friesland input:user-valid](#), [# friesland input:user-valid ~ .afk](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid en :user-invalid](#).

```
#drenthe input:valid, #drenthe input:valid ~ .afk {top: calc(100% + 0.4em);}
```

Deze selectors werken alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selectors en css zijn vrijwel hetzelfde als die bij [#drenthe input:user-valid, #drenthe input:user-valid ~ .afk](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid](#) en [:user-invalid](#).

```
#overijssel input:valid {bottom: -2em; left: calc(54% + 1.9em);}
```

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selector en css zijn vrijwel hetzelfde als die bij [#overijssel input:user-valid](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid](#) en [:user-invalid](#).

```
#overijssel input:valid ~ .afk {bottom: -2em;}
```

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selector en css zijn vrijwel hetzelfde als die bij [#overijssel input:user-valid ~ .afk](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid](#) en [:user-invalid](#).

```
#flevoland input:valid, #flevoland input:valid ~ .afk {bottom: -3.8em;}
```

```
# gelderland input:valid {top: calc(100% + 2.4em); left: calc(54% + 1.9em);}
```

```
# gelderland input:valid ~ .afk {top: calc(100% + 2.4em);}
```

```
# utrecht input:valid, # utrecht input:valid ~ .afk {top: calc(100% + 4.2em);}
```

```
# noord-holland input:valid, # noord-holland input:valid ~ .afk {bottom: -5.8em;}
```

```
# zuid-holland input:valid {right: 54%; bottom: -7.8em; left: auto;}
```

```
# zuid-holland input:valid ~ .afk {bottom: -7.8em;}
```

```
# zeeland input:valid, # zeeland input:valid ~ .afk {bottom: -7.8em;}
```

```
# noord-brabant input:valid {top: calc(100% + 8.4em); left: 1.9em;}
```

```
# noord-brabant input:valid ~ .afk {top: calc(100% + 8.4em);}
```

```
# limburg input:valid {top: calc(100% + 8.4em); left: calc(54% + 1.9em);}
```

```
# limburg input:valid ~ .afk {bottom: -9.9em;}
```

Deze selectors werken alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selectors en css zijn vrijwel hetzelfde als die bij [#overijssel input:user-valid ~ .afk](#) en daaronder. De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid](#) en [:user-invalid](#).

**input:valid {counter-increment: goed leeg -1;}**

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

Deze selector en css zijn vrijwel hetzelfde als die bij [input:user-valid](#). De beschrijving staat daar. Alleen wordt hier geen `:user-valid`, maar `:valid` gebruikt. Het verschil tussen deze twee staat beschreven bij [:user-valid en :user-invalid](#).

**#fout, #leeg**

Deze selectors werken alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

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

[span](#) {display: none;}

[#goed, #fout, #leeg](#) {display: inline;}

De elementen met `id="leeg"` en `id="fout"`. De `<span>`'s met 'Fout: ' en 'Nog niet ingevuld: '.

**display: none;**

In browsers die `:user-valid` en `:user-invalid` niet ondersteunen worden de tellers voor verkeerd ingevulde en nog niet ingevulde velden gecombineerd weergegeven. Daarom worden hier de afzonderlijke `<span>`'s voor de verkeerd ingevulde en nog niet ingevulde velden (met de daarachter staande tellers) verborgen.

**#fout-of-leeg**

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

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

[span](#) {display: none;}

Het element met `id = "fout-of-leeg"`. De `<span>` met 'Fout of nog niet ingevuld: '.

**display: inline;**

De `<span>` zichtbaar maken. Omdat deze `<span>` samen met `span#goed` (de `<span>` met 'Goed: ') op één regel moet staan, wordt deze als inline-element getoond.

**#fout-of-leeg::after**

Deze selector werkt alleen in browsers die `:user-valid` níét ondersteunen. Voor andere browsers is de uitleg hieronder niet van belang.

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

[span](#) {display: none;}

[#fout-of-leeg](#) {display: inline;}

Met behulp van `::after` wordt bij het element met `id="goed"`, de `<span>` met 'Goed: ', een pseudo-element gemaakt. Dit pseudo-element wordt gebruikt om de teller met het aantal goed ingevoerde provinciehoofdsteden weer te geven.

**content: counter(leeg);**

De inhoud van `content` komt in het met behulp van `::after` gemaakte pseudo-element te staan, achter de `<span>` met 'Fout of nog niet ingevuld: '. Hier is die inhoud de teller met het aantal lege of verkeerd ingevoerde velden.

Deze teller is bij [main](#) geïnitieerd met de waarde 12. Bij deze browsers die `:user-valid` en `:user-invalid` niet ondersteunen wordt de teller voor nog niet ingevulde velden alleen verlaagd, als een veld goed is ingevuld. Als het veld verkeerd wordt ingevuld, wordt de teller niet verlaagd. Daardoor geeft in deze browsers teller 'leeg' het gecombineerde aantal van nog niet ingevulde en verkeerd ingevulde velden weer.

## css voor alle browsers en vensters

### #fout::after

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

```
span {display: none;}  
#goed, #fout, #leeg {display: inline;}
```

Als `:user-valid` niet wordt ondersteund:

```
#fout, #leeg {display: none;}
```

Met behulp van `::after` wordt bij het element met `id="fout"`, de `<span>` met 'Fout: ', een pseudo-element gemaakt. Dit pseudo-element wordt gebruikt om de teller met het aantal fout ingevoerde provinciehoofdsteden weer te geven.

### content: counter(fout);

De inhoud van `content` komt in het met behulp van `::after` gemaakte pseudo-element te staan, achter de `<span>` met 'Fout: '. Hier is die inhoud de teller met het aantal fout ingevoerde velden.

### #leeg::after

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

```
span {display: none;}  
#goed, #fout, #leeg {display: inline;}
```

Als `:user-valid` niet wordt ondersteund:

```
#fout, #leeg {display: none;}
```

Met behulp van `::after` wordt bij het element met `id="leeg"`, de `<span>` met 'Leeg: ', een pseudo-element gemaakt. Dit pseudo-element wordt gebruikt om de teller met het aantal nog niet ingevulde velden weer te geven.

### content: counter(leeg);

De inhoud van `content` komt in het met behulp van `::after` gemaakte pseudo-element te staan, achter de `<span>` met 'Leeg: '. Hier is die inhoud de teller met het aantal goed nog niet ingevulde velden.

## css voor vensters minimaal 500 px breed

### @media screen and (min-width: 500px)

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 minimaal 500 px breed zijn. In deze iets bredere vensters is er iets meer ruimte, waardoor de namen van de provincies en de `<input>`'s voor de provinciehoofdsteden iets beter verdeeld kunnen worden.

`@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 veel meer 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.

`(min-width: 500px):` het browservenster moet minimaal 500 px breed zijn. Is het venster smaller, dan wordt de css die binnen deze media query staat genegeerd. Gelijk na deze regel komt een `{` te staan, en aan het einde van de css die binnen deze query valt een bijbehorende afsluitende `}`. Die zijn in de regel hierboven weggevallen, maar het geheel ziet er zo uit:

```
@media screen and (min-width: 500px) {  
    body {color: silver;}  
    (...) rest van de css voor deze media query (...)  
    footer {color: gold;}  
}
```

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

Als je nou 'n mobieltje hebt met een resolutie van – om maar wat te noemen – 900 x 768 px, dan geldt deze media query soms toch niet voor dat mobieltje. Terwijl dat toch echt meer dan 500 px breed is. Een vuig complot van gewetenloze multinationals? Voordat je je gaat beklagen bij Radar, zou ik eerst even verder lezen.

Mobiele apparaten, maar ook steeds meer gewone beeldschermen, hebben een hogere resolutiedichtheid. Dat wil zeggen dat ze kleinere pixels hebben, die dichter bij elkaar staan. Daardoor zijn foto's, tekst, e.d. veel scherper weer te geven. Hoe kleiner de puntjes (de pixels) zijn, waaruit een afbeelding is opgebouwd, hoe duidelijker het wordt.

Er ontstaat alleen één probleem: als je de pixels twee keer zo klein maakt, wordt ook wat je ziet twee keer zo klein. En inmiddels zijn er al apparaten met pixels die meer dan vier keer zo klein zijn. Een lijntje van 1 px breed zou op die apparaten minder dan 'n kwart van de oorspronkelijke breedte krijgen en vrijwel onzichtbaar zijn. Een normale foto zou in een thumbnail veranderen. Kolommen zouden heel smal worden. Tekst zou onleesbaar klein worden. Allemaal fantastisch scherp, maar je hebt 'n vergrootglas nodig om 't te kunnen zien.

Om dit te voorkomen wordt een verschil gemaakt tussen css-pixels en schermpixels (in het Engels 'device-pixels'). De css-pixels zijn gebaseerd op de – tot voor kort – normale beeldschermen van de desktop. 1 css-pixel is op zo'n beeldscherm 1 pixel. Het aantal schermpixels is het werkelijk op het apparaat aanwezige aantal pixels (dat is het aantal pixels, waarvoor je hebt betaald).

Dat eerder genoemde mobieltje van 900 x 768 px heeft wel degelijk het aantal pixels, waarvoor je hebt betaald. Maar die zitten dichter bij elkaar. Op een gewoon beeldscherm zitten 96 pixels per inch, wat wordt uitgedrukt met de eenheid ppi ('pixels per inch'). (Vaak wordt foutief de eenheid dpi ('dots per inch') gebruikt. Die eenheid is voor printers.) Als dat mobieltje een resolutie van 192 ppi heeft, 192 pixels per inch, zijn de pixels ervan twee keer zo klein als op een origineel beeldscherm. Er zijn per inch twee keer zoveel schermpixels aanwezig.

Om nu te voorkomen dat alles op dat mobieltje twee keer zo klein wordt, geeft het mobieltje niet het echte aantal schermpixels (900 x 768), maar een lager aantal css-pixels door bij een media query. De 192 ppi van het mobieltje is twee keer zo veel als de 96 ppi van een normaal beeldscherm. Het aantal css-pixels is dan het aantal schermpixels gedeeld door 2. 1024 x 768 gedeeld door 2 is 512 x 384 px. Het aantal css-pixels is 512 x 384 px en zit daarmee dus ruim onder de grens van deze media query.

Je bent dus niet opgelicht, of in ieder geval niet wat betreft het aantal pixel.

Door deze truc is een lijn van 1 px breed op een normaal beeldscherm ook op het mobieltje nog steeds 1 px breed, alleen wordt die ene (css-)pixel opgebouwd uit twee schermpixels

(feitelijk vier, want het verhaal geldt voor breedte én hoogte). De dikte van het lijntje is hetzelfde, maar het is veel fijner opgebouwd. Bij lijntjes is dat verschil bijvoorbeeld in bochten goed te zien.

Hetzelfde verhaal geldt voor hogere resoluties. Een tablet met een breedte van 1800 scherpixels en een ppi van 384 (vier keer de originele dichtheid) geeft 1800 gedeeld door 4 = 450 css-pixel door. Het lijntje van 1 px breedte op de originele monitor is nog steeds 1 css pixel breed op de tablet, maar die ene css-pixel is nu opgebouwd uit zestien scherpixels. Kort samengevat: omdat niet het aantal scherpixels (waarvoor je hebt betaald), maar het aantal css-pixels (de door de ontwerper bedoelde afmeting) wordt doorgegeven, wordt voorkomen dat een hogeresolutiescherm onleesbaar klein wordt.

## # groningen label

Deze selector werkt alleen in browservensters minimaal 500 px breed. Voor andere vensters is de uitleg hieronder niet van belang.

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.)

```
label {font-size: 0.8em; text-align: center; position: absolute;}
```

```
#groningen label {width: 3em; bottom: 80%; left: 74%;}
```

```
#wrapper div:has\(:focus\) label {background: rgb(255 255 255 / 0.8); color: black; font-weight: bold; z-index: 20;}
```

```
#wrapper div:has\(:user-valid\) label {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}
```

Als `:user-valid` niet wordt ondersteund:

```
#wrapper div:has\(:valid\) label {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}
```

De `<label>`'s binnen het element met `id="groningen"`. Dat is er maar één: de `<label>` voor provinciehoofdstad 'Groningen' binnen `div#groningen`.

**bottom: 84%; left: 76%;**

De eerder bij [#groningen label](#) opgegeven waarden worden hier iets aangepast, omdat in deze iets bredere browservensters de `<label>` met de provincienaam iets beter kan worden gepositioneerd.

## # groningen input

Deze selector werkt alleen in browservensters minimaal 500 px breed. Voor andere vensters is de uitleg hieronder niet van belang.

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.)

```
input {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em;
border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}
# groningen input {margin-bottom: 10px; margin-left: calc(3.5em + 30px); bottom: calc(80% - 2em); left:
calc(74% - 3.1em);}
#wrapper input:focus {width: 9em; margin: 0; border-radius: 0; z-index: 20;}
#wrapper input:user-valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0;
border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width, border-left,
top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s; transition-timing-function:
ease-in; transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;}
#wrapper div:nth-of-type(even) input:user-valid {right: 0; left: auto;}
# groningen input:user-valid, # groningen input:user-valid ~ .afk, # friesland input:user-valid, # friesland
input:user-valid ~ .afk {bottom: 0;}
```

Als :user-valid niet wordt ondersteund:

```
#wrapper input:valid {display: block; box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem;
margin: 0; border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width,
border-left, top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s;
transition-timing-function: ease-in; transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;}
#wrapper div:nth-of-type(even) input:valid {right: 0; left: auto;}
```

De <input>'s binnen het element met id="groningen". Dat is er maar één: de <input> waarin 'Groningen' moet worden ingevoerd.

**bottom: calc(84% - 2em); left: calc(76% - 3.1em);**

De eerder bij # groningen input opgegeven waarden worden hier iets aangepast, omdat in deze iets bredere browservensters de <input> iets beter kan worden gepositioneerd.

```
# friesland label {right: 32%; bottom: 79%;}
# friesland input {right: calc(32% - 3.1em); bottom: calc(79% -
2em);}
# overijssel label {width: auto; bottom: 58%;}
# overijssel input {bottom: calc(58% - 2em); left: calc(74% -
3.1em);}
# flevoland label {left: 52%;}
# flevoland input {bottom: calc(56% - 2em); left: calc(52% -
3.1em);}
# gelderland label {width: auto; top: 49%; left: 62%;}
# gelderland input {top: calc(49% + 1.7em); left: calc(62% -
2.2em);}
# utrecht label {left: 44%;}
# utrecht input {left: calc(44% - 3.1em);}
# noord-holland label {right: 56%;}
# noord-holland input {right: calc(56% - 2.7em);}
# zuid-holland label {right: 62%;}
# zuid-holland input {right: calc(62% - 2.7em);}
# zeeland label {width: 4em; right: 74%; bottom: 30%;}
# zeeland input {right: calc(74% - 2.8em); bottom: calc(30% -
2em);}
# noord-brabant label {left: 40%;}
```

```
#noord-brabant input {left: calc(40% - 1.4em);}
#limburg label {top: 83%; left: 60%;}
#limburg input {top: calc(83% + 1.6em); left: calc(60% - 2.8em);}
```

Deze selectors werken alleen in browservensters minimaal 500 px breed. Voor andere vensters is de uitleg hieronder niet van belang.  
Voor deze elementen is eerder css opgegeven. Deze wordt hier niet allemaal herhaald, omdat het nogal veel is. Hier wordt alleen de plaats van de <label>'s en <input>'s aangepast.

Dit zijn de <label>'s en <input>'s die bij de provincies en provinciehoofdsteden horen. De css werkt precies hetzelfde als die bij [# groningen label](#) en [# groningen input](#) hierboven. De beschrijving is daar te vinden.

## #wrapper div:has(:focus) label

Deze selector werkt alleen in browservensters minimaal 500 px breed. Voor andere vensters is de uitleg hieronder niet van belang.  
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](#) {font-size: 0.8em; text-align: center; position: absolute;}

[#wrapper div:has\(:focus\) label](#) {background: rgb(255 255 255 / 0.8); color: black; font-weight: bold; z-index: 20;}

[#wrapper div:has\(:user-valid\) label](#) {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}

Als :user-valid niet wordt ondersteund:

[#wrapper div:has\(:valid\) label](#) {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}

Naast bovenstaande css is elke <label> met behulp van top, right, bottom en/of left op de juiste plaats gepositioneerd. Bij langere provincienamen is met behulp van width de breedte van de <label> beperkt, zodat de naam op twee regels komt te staan. De beschrijving van deze css is te vinden bij [# groningen label](#) en verder.

Oudere browsers ondersteunen :has() niet en negeren daarom deze selector en de in deze regel zittende css. Deze browsers geven daardoor geen tellers voor goed, verkeerd of nog niet ingevulde hoofdsteden weer, en geen goed- en foutmeldingen. Dit staat uitgebreider beschreven bij [Overige problemen](#).

Doe iets met de <label>'s binnen de <div>'s die binnen div#wrapper zitten, maar alleen als een element binnen die <div> de [focus](#) heeft. Deze selector werkt precies hetzelfde als die bij [#wrapper div:has\(:focus\) label](#). De beschrijving staat daar.

## font-size: 1em; font-weight: normal;

Omdat in deze iets bredere browservensters iets meer ruimte is, kan bij [focus](#) de lettergrootte van de <label>'s iets worden verhoogd. Voor smallere vensters is eerder opgegeven dat de tekst bij focus vet moet worden. Dat is, in combinatie met de iets grotere letter hier, iets te veel van het goede, daarom wordt in deze iets bredere vensters de tekst weer normaal weergegeven.

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.

## css voor vensters minimaal 760 px breed

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

De opbouw van deze regel staat beschreven bij [@media screen and \(min-width: 500px\)](#). Er is één verschil: de breedte van het browservenster moet minimaal 760 px zijn.

body

Deze selector werkt alleen browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.

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;}`

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 in deze grotere browservensters. 't Zal de leeftijd zijn, maar de standaardgrootte is wat 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.

## #wrapper div

Deze selector werkt alleen browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.

`#wrapper`: het element met `id="wrapper"`. De `<div>` waar behalve de titel en de uitslag alles in zit.

`div`: elke provincie staat met alle toebehoren in een eigen `<div>`.

## `display: flex;`

De waarde `flex` is onderdeel van een hele reeks eigenschappen en waarden, die bij elkaar 'flexbox' worden genoemd. Met `display: flex;` wordt elke `<div>` in een zogenaamde 'flex container' veranderd. Dit maakt het veel makkelijker om de directe kinderen van dit element, de 'flex items', op een bepaalde plaats neer te zetten. Elke `<div>` heeft hier als direct kind een `<label>`, een `<input>` en een `<span>`. In dit geval komen deze `<label>`, `<input>` en `<span>` naast elkaar te staan, omdat dat de standaardinstelling van `flex-direction` is. Met `flex-direction` kan de richting van de 'hoofdas' (in het Engels 'main axis') worden opgegeven: regel of kolom. De andere richting heet 'dwarsas' (in het Engels 'cross axis'). De standaardinstelling is `flex-direction: row;`, en die wordt hier niet aangepast. Zou je hier `flex-direction: column;` gebruiken, dan zouden de flex items onder elkaar komen te staan.

De waarde van `flex-direction` is ook van belang voor een aantal andere eigenschappen van flexbox, zoals het iets hieronder gebruikte `align-content` en `justify-content`.

## `flex-wrap: wrap;`

De standaardwaarde bij `flex-wrap` is `no-wrap`: zet alle flex items, hier de `<label>`, de `<input>` en de `<span>` in elke `<div>`, naast elkaar op één regel. Als het niet meer op één regel past, maak ze dan smaller.

Bij `#wrapper input` iets hieronder worden de `<label>`'s 9 em breed gemaakt. Bij `#wrapper label` verderop worden de `<label>`'s met `width: auto;` zo breed gemaakt als nodig is voor de erin zittende provincienaam. Elke `<div>` krijgt hieronder een eigen breedte en hoogte. Als dat niet wordt aangepast, ziet dat eruit als op de afbeelding:

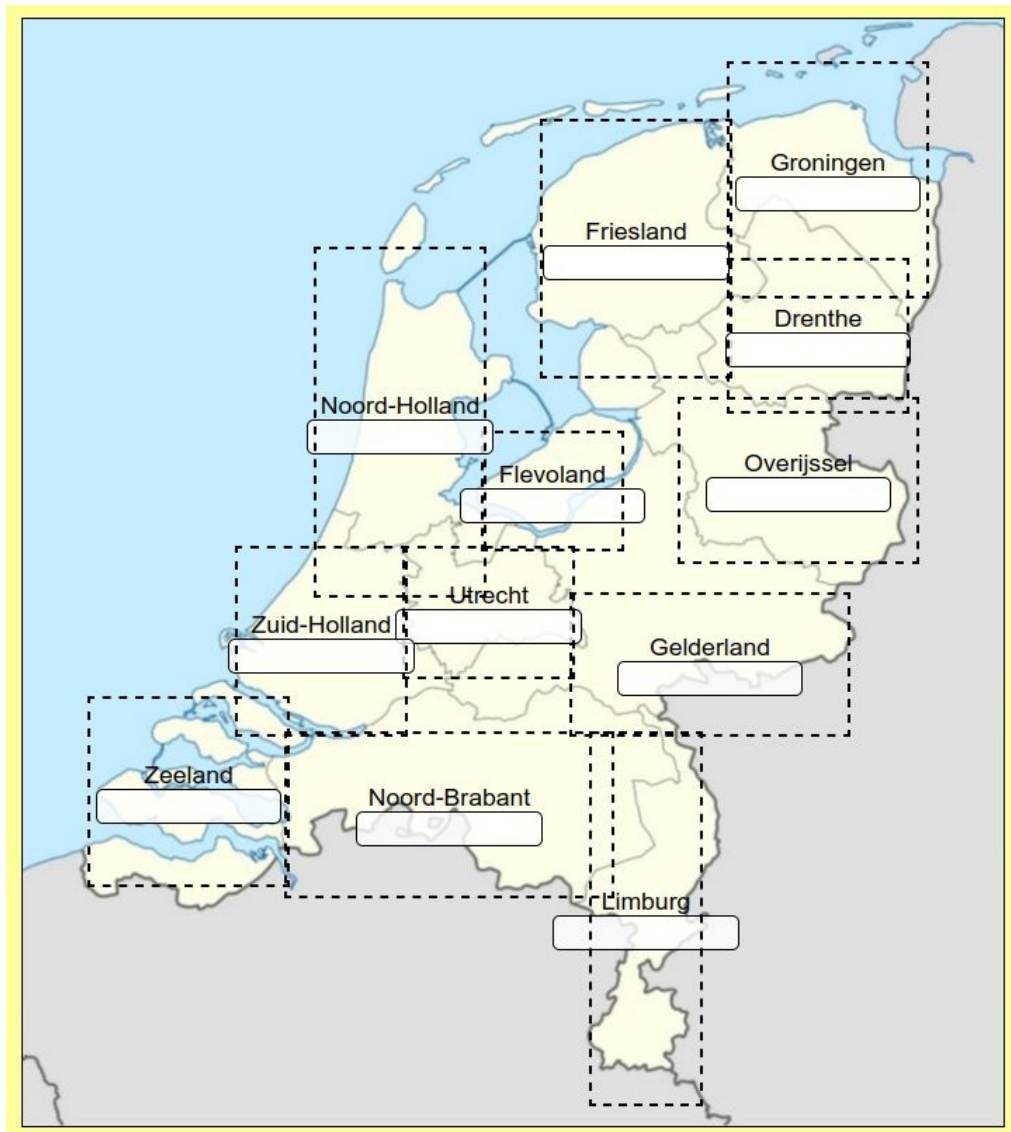


Elke `<div>` krijgt later een hoogte, die is hier voor de duidelijkheid even weggehaald. De zwarte streepjeslijn geeft de omtrek van elke `<div>` aan. Daarbinnen staat links de `<label>` met de provincienaam, met daarnaast de 9 em brede `<input>`. Omdat `<label>` en `<input>` te breed zijn voor de `<div>`, staan de `<input>`'s voor een (groot) deel rechts buiten de `<div>`. En als sommige provincienamen geen `&shy;` in de naam zouden hebben, waardoor ze kunnen afbreken en op twee regels komen te staan, zou het er nog eh, artistieker uitzien.

De waarde `wrap` voorkomt deze puinhoop. Als `<label>` en `<input>` te breed worden om naast elkaar te staan (en dat is hier altijd het geval), worden ze op meerdere regels gezet.

De `<span>`, het derde directe kind, komt in dit verhaal helemaal niet meer voor. In deze `<span>` zit de afkorting voor de provincie, die in smallere browservensters onder de kaart van Nederland wordt gezet. Hier is die niet nodig en bovendien zijn twee directe kinderen wel genoeg met al die overbevolking, dus die `<span>` wordt later met `display: none;` verborgen.

`align-content: center; justify-content: center;`



Elke `<div>` krijgt hieronder een eigen grootte en positie, maar normaal genomen zie je daar niets van. Op de afbeelding zijn grootte en positie van elke `<div>` met een zwarte streepjeslijn zichtbaar gemaakt.

`align-content` en `justify-content` zijn twee van de eigenschappen die bij flexbox horen.

Met `align-content: center;` worden de flex items (hier `<label>` en `<input>`) gecentreerd in de richting van de dwarsas. Hier is die richting verticaal: centreer de flex items verticaal binnen de `<div>`'s, ongeacht de hoogte van de `<div>`.

Met `justify-content: center;` gebeurt hetzelfde, maar nu in de richting van de hoofdas, die hier horizontaal is: centreer de flex items horizontaal binnen de `<div>`.

Nu staan de `<label>`'s en `<input>`'s altijd horizontaal en verticaal gecentreerd binnen de `<div>`, ongeacht breedte en hoogte van de `<div>`.

**`position: absolute;`**

Om de `<div>`'s op een bepaalde plaats neer te kunnen zetten.

Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, die bij [#wrapper](#) een relatieve positie heeft gekregen.

## # groningen

Deze selector werkt alleen in browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.

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.)

`#wrapper div {display: flex; flex-wrap: wrap; align-content: center; justify-content: center; position: absolute;}`

Het element met id="groningen". De <div> met de provincie Groningen.

**width: 20%; height: 20%;**

Bij een absoluut gepositioneerd element, zoals deze <div>, wordt een breedte in procenten genomen ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, waarin ook de afbeelding met de kaart van Nederland zit.

Bij `#wrapper` en `#kaart` is geregeld dat de grootte van beide wordt aangepast aan de grootte van het browservenster: hoe smaller het venster, hoe smaller `div#wrapper` en de afbeelding met de kaart van Nederland worden.

Omdat de breedte van `div#groningen` mee verandert met de breedte van `div#wrapper` (en dus ook met de daarin zittende afbeelding van de kaart van Nederland), blijft de <div> altijd ongeveer even breed als de provincie Groningen op de kaart.

Hetzelfde geldt voor de hoogte van twintig procent: omdat ook de hoogte in procenten is opgegeven, verandert ook deze mee met de grootte van `div#wrapper` en de kaart van Nederland. Hierdoor blijft `div#groningen` ook in de hoogte altijd ongeveer boven de provincie Groningen staan.

Dit werkt voor alle <div>'s met een provincie op dezelfde manier. De positie is niet voor alle provincies helemaal optimaal, want de lettergrootte moet tot 200% vergroot kunnen worden, dus er moest een evenwicht gevonden worden tussen grootte en de mogelijkheid de letters te vergroten.

Omdat bij `#wrapper div` hierboven de inhoud van `div#groningen` met `align-content: center;` en `justify-content: center;` horizontaal en verticaal is gecentreerd binnen `div#groningen`, staat ook die inhoud (provincienaam, tekstveld, e.d.) altijd boven de provincie Groningen.

**top: 4%; left: 72%;**

Bij `#wrapper div` zijn de <div>'s met de provincies absoluut gepositioneerd. Er wordt gepositioneerd ten opzichte van het 'containing block'. Dat is de eerste voorouder die zelf een bepaalde eigenschap heeft, zoals een andere positie dan statisch. Dat is hier `div#wrapper`, de <div> waar ook de kaart van Nederland in zit.

Voor de positie geldt hetzelfde als hier gelijk boven voor de breedte en hoogte: deze verandert mee met de grootte van `div#wrapper` en de kaart van Nederland, omdat de positie in procenten is opgegeven.

```
#friesland {width: 19%; height: 22%; top: 9%; left: 53%;}
#drenthe {width: 18%; height: 13%; top: 21%; left: 72%;}
#overijssel {width: 24%; height: 14%; top: 33%; left: 67%;}
#flevoland {width: 14%; height: 10%; top: 36%; left: 47%;}
#gelderland {width: 28%; height: 12%; top: 50%; left: 56%;}
#utrecht {width: 17%; height: 11%; top: 46%; left: 39%;}
```

```
#noord-holland {width: 17%; height: 30%; top: 20%; left: 30%;}
#zuid-holland {width: 17%; height: 16%; top: 46%; left: 22%;}
#zeeland {width: 20%; height: 16%; top: 59%; left: 7%;}
#noord-brabant {width: 33%; height: 14%; top: 62%; left: 27%;}
#limburg {width: 11%; height: 32%; top: 62%; left: 58%;}
```

Deze selectors werken alleen in browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.  
 Voor deze elementen is eerder css opgegeven. Deze wordt hier niet allemaal herhaald, omdat het nogal veel is. Hier wordt alleen de plaats van de <div>'s met de provincies aangepast.

Dit zijn de <div>'s, waarin de <input>'s en tekstvelden met bijbehorende meldingen en dergelijke zitten. De css werkt precies hetzelfde als die bij [# groningen](#). De beschrijving is daar te vinden. Alleen zijn de waarden voor elke <div> anders, omdat elke <div> boven een andere provincie staat.

## #wrapper label

Deze selector werkt alleen in browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.  
 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](#) {font-size: 0.8em; text-align: center; position: absolute;}

[#wrapper div:has\(:focus\) label](#) {background: rgb(255 255 255 0.8); color: black; font-weight: bold; z-index: 20;}

[#wrapper div:has\(:user-valid\) label](#) {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}

Als :user-valid niet wordt ondersteund:

[#wrapper div:has\(:valid\) label](#) {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}

Naast bovenstaande css is elke <label> met behulp van top, right, bottom en/of left op de juiste plaats gepositioneerd. Bij langere provincienamen is met behulp van width de breedte van de <label> beperkt, zodat de naam op twee regels komt te staan. De beschrijving van deze css is te vinden bij [# groningen label](#) en verder en voor browservensters minimaal 500 px breed bij [# groningen label](#) en verder.

Alle <label>'s binnen het element met id="wrapper". In elke <label> zit een provincienaam, die zichtbaar is. Daarnaast zit in elke <label> nog een aantal <span>'s met meldingen die alleen zichtbaar worden als iets goed of fout is ingevuld in het bij de <label> horende tekstveld.

## width: auto;

In kleinere browservensters is de breedte van de <label>'s bij gebrek aan ruimte beperkt. In deze grotere vensters is dat niet meer nodig.

## font-size: 1em;

Voor kleinere browservensters is de lettergrootte eerder wat verkleind. In deze grotere vensters is dat niet meer nodig.

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.

## position: static;

Eerder werden de <label>'s met behulp van position: absolute; gepositioneerd. In deze bredere browservensters wordt dat bij [#wrapper div](#) met behulp van flexbox gedaan. Om flexbox goed te laten werken, moet position terug worden veranderd naar de standaardwaarde static.

## #wrapper div:has(:focus) label

Deze selector werkt alleen in browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.

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](#) {font-size: 0.8em; text-align: center; position: absolute;}

[#wrapper div:has\(:focus\) label](#) {background: rgb(255 255 255 / 0.8); color: black; font-weight: bold; z-index: 20;}

[#wrapper div:has\(:user-valid\) label](#) {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}

Als :user-valid niet wordt ondersteund:

[#wrapper div:has\(:valid\) label](#) {background: lightgreen; color: black; outline: darkgreen solid 2px; padding: 1px;}

Alleen in browservensters minimaal 760 px breed:

[#wrapper label](#) {width: auto; font-size: 1em; position: static;}

Naast bovenstaande css is elke <label> met behulp van top, right, bottom en/of left op de juiste plaats gepositioneerd. Bij langere provincienamen is met behulp van width de breedte van de <label> beperkt, zodat de naam op twee regels komt te staan. De beschrijving van deze css is te vinden bij [# groningen label](#) en verder en voor browservensters minimaal 500 px breed bij [# groningen label](#) en verder.

Oudere browsers ondersteunen :has() niet en negeren daarom deze selector en de in deze regel zittende css. Deze browsers geven daardoor geen tellers voor goed, verkeerd of nog niet ingevulde hoofdsteden weer, en geen goed- en foutmeldingen. Dit staat uitgebreider beschreven bij [Overige problemen](#).

#wrapper: het element met id="wrapper". De <div> waar behalve de titel en de uitslag alles in zit.

div: alle <div>'s in div#wrapper. De twaalf <div>'s met de provincies.

:has(): maar alleen als de <div>'s voldoen aan de voorwaarde die tussen de haakjes staat.

:has(:focus): dit is de voorwaarde, waaraan de <div>'s moeten voldoen. Een element binnen de <div> moet de [focus](#) hebben. Hier kan alleen de <input> binnen de <div> de focus krijgen.

label: de <label>'s binnen de <div>'s. In elke <label> zitten meldingen, een provincienaam en een afkorting.

De selector in gewone taal: doe iets met de <label> in de <div> binnen div#wrapper, maar alleen als binnen die <div> een element de focus heeft.

### background: transparent;

Achtergrond doorzichtig maken. Het contrast is hier, in combinatie met de lettergrootte, ook zonder gekleurde achtergrond voldoende.

### font-size: 1.5em;

Letter vergroten, zodat goed duidelijk is bij welke provincie het tekstveld hoort dat de [focus](#) heeft.

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.

## .fout

Deze selector werkt alleen in browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.

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.)

`span {display: none;}`

`.fout {background: rgb(255 255 255 / 0.7); color: black; width: 4ch; font-size: 0.7em; margin: 0 auto; border: red solid 1px; padding: 2px; position: absolute; top: 1em; right: 0; left: 0;}`

`#wrapper div:has\(:user-invalid\):not\(:has\(:focus\)\) .fout {display: block;}`

Alle elementen met class="fout". In deze `<span>`'s zit de foutmelding die verschijnt, als een hoofdstad fout wordt ingevuld.

## position: static;

Eerder werden de `<span>`'s met class="fout" met behulp van `position:`

`absolute`; gepositioneerd. Hier worden ze gewoon met het ook eerder opgegeven `margin: 0 auto`; horizontaal gecentreerd binnen de `<label>`, waar ze in zitten.

Om dat goed te laten werken, moet de absolute positie worden veranderd naar de standaardwaarde statisch.

## #wrapper input, #wrapper input:valid

Deze selector werkt alleen in browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.

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

`input {background: white; color: black; width: 2em; height: 1.2em; font-size: 0.8em; margin-left: 3.5em; border: black solid 1px; border-radius: 5px; position: absolute; transition: width 0.5s, margin 0.5s;}`

`#wrapper input:focus {width: 9em; margin: 0; border-radius: 0; z-index: 20;}`

`#wrapper input:user-valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0; border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width, border-left, top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s; transition-timing-function: ease-in; transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;}`

`#wrapper div:nth-of-type\(even\) input:user-valid {right: 0; left: auto;}`

Als `:user-valid` niet wordt ondersteund:

`#wrapper input:valid {box-sizing: border-box; width: calc(46% - 1.5rem); height: 1.2rem; margin: 0; border-left-width: 0; border-radius: 0; left: 1.8em; z-index: 10; transition-property: width, border-left, top, right, bottom, left; transition-duration: 2s, 0s, 2s, 2s, 2s, 2s; transition-timing-function: ease-in; transition-delay: 0s, 2s, 0s, 0s, 0s, 0s; padding: 1px;}`

`#wrapper div:nth-of-type\(even\) input:valid {right: 0; left: auto;}`

Naast bovenstaande css is elke `<input>` met behulp van `top`, `right`, `bottom`, `left`, `margin-right`

en/of `margin-left` op de juiste plaats neergezet. De beschrijving van deze css is te vinden bij [# groningen input](#) en verder en voor browservensters minimaal 500 px breed bij [# groningen input](#) en verder.

`#wrapper`: het element met id="wrapper". De `<div>` waar behalve de titel en de uitslag alles in zit.

`input`: alle `<input>`'s in `div#wrapper`.

Daarna volgt nog een tweede selector `#wrapper input:valid`: als de `<input>` binnen `div#wrapper` een geldige invoer heeft. (De controle op fouten gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij [<input id="gr"...>](#))

Deze tweede selector is alleen nodig, omdat eerder selectors als `#wrapper`

`input:focus` en `#wrapper input:user-invalid` zijn gebruikt. css wordt in

principe uitgevoerd in de volgorde, waarin deze in het stijlbestand staat: css bij latere

selectors 'wint' van css bij eerdere selectors. Maar alleen als die selectors evenveel gewicht hebben, evenveel 'specificiteit'.

De eerste selector `#wrapper input` hier heeft 1 id en 1 element.

Een eerder gebruikte selector als `#wrapper input:focus` heeft dat ook, maar die heeft ook nog de pseudo-class `:focus`. Voor de specificiteit telt een pseudo-class als een gewone class, dus `#wrapper input:focus` heeft 1 id, 1 element en 1 class. En 'wint' daarmee van de selector `#wrapper input`, die maar 1 id en 1 element heeft, ook al staat deze selector alter in de css.

Daarom wordt hier ook de tweede selector `#wrapper input:valid` gebruikt: deze heeft ook 1 id, 1 element en 1 pseudo-class. Daarmee heeft deze selector evenveel specificiteit als de eerdere selectors met `:focus` en dergelijke, en omdat deze selector later in de css staat, 'wint' deze nu van de eerdere selectors. Waardoor de bij deze selector staande css wordt toegepast.

**background: `rgb(255 255 255 / 0.8);`**

Iets doorzichtige witte achtergrond. Een uitgebreide beschrijving van deze eigenschap staat bij [background: `rgb\(255 255 255 / 0.7\);`](#).

**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 stijlbestand, 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.

**width: `9em;`**

Breedte.

Als eenheid wordt de relatieve eenheid `em` gebruikt, zodat de breedte mee verandert met de lettergrootte. Bij gebruik van een absolute eenheid zoals `px` verandert de breedte niet mee met de lettergrootte. Zoomen kan wel altijd, ongeacht welke eenheid voor de breedte wordt gebruikt.

**height: `auto;`**

Eerder is bij [input](#) de hoogte van de `<input>`'s iets verkleind. In deze grotere browservensters is dit niet meer nodig en mogen de `<input>`'s hun standaardhoogte krijgen.

**margin: `0;`**

Alle eerder opgegeven marges verwijderen.

De `<input>`'s zijn hier iets boven bij [#wrapper div](#) met behulp van `align-content: center;` en `justify-content: center;` horizontaal en verticaal gecentreerd binnen hun `<div>`, dus marges zijn hier niet meer nodig.

`border-left: black solid 1px; border-radius: 5px;`

In smallere browservensters wordt de `<input>` onder de kaart van Nederland gezet, als de juiste hoofdstad is ingevoerd. Daar moeten de border links en de ronde hoeken

worden weggehaald. In deze grotere vensters blijft de `<input>` gewoon op dezelfde plaats staan en mogen de border links en de ronde hoeken blijven staan.

**position: static;**

Eerder werden de `<input>`'s gepositioneerd met behulp van een absolute positie. Hier zijn de `<input>`'s iets hierboven bij [#wrapper div](#) met behulp van `align-content: center;` en `justify-content: center;` horizontaal en verticaal gecentreerd binnen hun `<div>`. Om dat goed te laten werken, moet de absolute positie terug worden naar de standaardwaarde statisch.

## **#wrapper .afk**

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Deze selector werkt alleen in browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.<br>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.)<br><a href="#">.afk</a> {background: white; color: black; box-sizing: border-box; height: 1.2rem; font-size: 0.8em; border: black solid 1px; border-width: 1px 0 1px 1px; padding: 1px 20px 1px 2px; position: absolute; left: 0;}<br><a href="#">div:nth-of-type(even) .afk</a> {left: 54%;}<br><a href="#">input:user-valid ~ .afk</a> {display: block;} |
| Als <code>:user-valid</code> niet wordt ondersteund:<br><a href="#">input:valid ~ .afk</a> {display: block;}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Naast bovenstaande css is elke <code>.afk</code> met behulp van <code>top</code> , <code>bottom</code> en/of <code>left</code> op de juiste plaats gepositioneerd. De beschrijving van deze css is te vinden bij <a href="#"># groningen input:user-valid</a> , <a href="#"># groningen input:user-valid ~ .afk</a> , <a href="#"># friesland input:user-valid</a> , <a href="#"># friesland input:user-valid ~ .afk</a> en verder.<br>(De css als <code>:user-valid</code> niet wordt ondersteund is hierbij hetzelfde als wanneer <code>:user-valid</code> wel wordt ondersteund, alleen zit in de selector <code>:valid</code> in plaats van <code>:user-valid</code> .)                                                                                                |

Alle elementen met `class="afk"` in het element met `div="wrapper"`. In deze `<span>`'s zit de afkorting van de provincie die, in smallere browservensters, onder de kaart van Nederland komt te staan, als een hoofdstad goed wordt ingevuld.

**display: none;**

In deze bredere browservensters wordt de afkorting voor de provincie helemaal niet gebruikt, dus wordt deze verborgen.

## **#naar-uitslag, #naar-boven**

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Deze selector werkt alleen in browservensters minimaal 760 px breed. Voor andere vensters is de uitleg hieronder niet van belang.<br>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.)<br><a href="#">#naar-uitslag, #naar-boven</a> {background: white; color: black; display: none; box-sizing: border-box; width: 700px; max-width: 94vw; font-size: 0.8em; text-align: center; margin-bottom 20em; border: black solid 1px; padding: 3px; position: relative; top: 12.4em;}<br><a href="#">#naar-boven</a> {display: block;} |
| Alleen als <code>:has()</code> wordt ondersteund:<br><a href="#">#naar-uitslag</a> {display: block;}<br><a href="#">#naar-boven</a> {display: none;}                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

De elementen met `id="naar-uitslag"` en `id="naar-boven"`. De links onderaan de pagina, waarmee je weer naar boven gaat. Welke van deze twee links zichtbaar is, is afhankelijk van de ondersteuning door de browser van `:has()`.

`font-size: 1em;`

In deze grotere browservensters is genoeg ruimte voor een iets grotere letter.

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: 30px;
```

Marge aan de onderkant, zodat duidelijk is dat de onderkant van de pagina is bereikt.

**top: 6px;**

Bij [#naar-uitslag](#), [#naar-boven](#) zijn deze links met een relatieve positie een heel eind naar beneden verplaatst. Hierdoor was er genoeg ruimte om de afkortingen van de provincies en de provinciehoofdsteden tussen de links en de kaart van Nederland te zetten. Hier is die ruimte niet nodig en is een kleine afstand tussen links en kaart voldoende.

### css voor alle browsers en vensters

```
#wrapper:not(:has(:placeholder-shown)):not(:has(:invalid)) +  
#uitslag #met-fout
```

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

Alleen in browsers die :has() ondersteunen:

```
#met-fout {margin: 0;}
```

Ziet er erger uit dan het is. Eerdere selectors heb je ook overleefd, dus...

#wrapper: het element met id="wrapper". De <div> waar behalve de titel en de uitslag alles in zit.

:not(): maar div#wrapper mag juist níét voldoen aan de voorwaarde die tussen de haakjes achter :not staat.

:not(:has()): tussen de haakjes achter :has staat de voorwaarde, waaraan div#wrapper juist níét mag voldoen.

:not(:has(:placeholder-shown)): dit is de voorwaarde, waaraan div#wrapper juist níét mag voldoen. Geen enkel element binnen div#wrapper mag de pseudo-class :placeholder-shown hebben.

In de html is bij elke <input> het attribuut placeholder-shown="" opgegeven.

Normaal genomen staat tussen de aanhalingstekens een hint als "adres": voer hier je adres in. Zodra er iets wordt ingevoerd, verdwijnt de hint.

Alleen een veld, waarin nog helemaal niets is ingevoerd, kan dus deze pseudo-class hebben. Met andere woorden: als geen enkele <input> binnen div#wrapper nog deze pseudo-class heeft, is in elke <input> iets ingevoerd. Of dat goed of fout is of later eventueel weer wordt verwijderd, maakt niets uit. Hoewel :placeholder-shown hier niet voor is bedoeld, kun je met deze truc prima vaststellen, of er nog lege velden zijn.

Een iets langer verhaal over :placeholder-shown staat bij [placeholder=""](#).

:not(): tussen de haakjes staat nog een voorwaarde, waaraan div#wrapper níét mag voldoen.

:not(:has()): tussen de haakjes achter :has staat weer de voorwaarde, waaraan div#wrapper juist níét mag voldoen.

:not(:has(:invalid)): dit is de tweede voorwaarde, waaraan div#wrapper juist níét mag voldoen. Geen enkel element binnen div#wrapper mag een verkeerde waarde hebben. Alleen de <input>'s binnen div#wrapper kunnen een ongeldige waarde hebben. Met andere woorden: geen van de twaalf <input>'s binnen div#wrapper mag een foutieve provincienaam hebben.

De controle op fouten gebeurt met behulp van het `pattern`-attribuut en staat beschreven bij [<input id="gr">...](#)

`+`: het element achter de `+` moet in de html direct volgen op het element voor de `+`. In dit geval gaat het om `div#uitslag` die gelijk op `div#wrapper` volgt. Beide elementen moeten ook nog dezelfde ouder hebben. Dat is hier het geval, ze hebben beide als ouder `<main>`.

`#uitslag`: het element met `id="uitslag"`. In deze `<div>` staan het aantal goed, verkeerd en nog niet ingevulde hoofdsteden, en een melding als alles goed is ingevuld.

`#met-fout`: het element met `id="met-fout"`. De `<p>` met de tellers met het aantal goede, verkeerde en nog niet ingevulde provinciehoofdsteden.

De hele selector in gewoon Nederlands: doe iets met het element, waarin de tellers staan, maar alleen als in alle `<input>`'s binnen `div#wrapper` iets is ingevoerd én als geen van deze `<input>`'s een fout bevat.

Als in elke `<input>` iets is ingevoerd, ook al is dat eventueel later weer verwijderd, én geen enkele `<input>` bevat een verkeerde hoofdstad, dan kan het niet anders dan dat alle hoofdsteden goed zijn ingevuld.

**`display: none;`**

Verberg de `<p>` met de tellers. Alle twaalf provinciehoofdsteden zijn goed ingevuld, de drie tellers met goed, fout en leeg worden verborgen.

**`#wrapper:not(:has(:placeholder-shown)):not(:has(:invalid)) + #uitslag #alles-goed`**

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

Alleen in browsers die `:has()` ondersteunen:

`#alles-goed` {background: lightgreen; color: black; display: none; font-size: 1.2em; margin: 0; border: black solid 1px; padding: 3px;}

Deze selector is vrijwel hetzelfde als die gelijk hierboven bij

`#wrapper:not(:has(:placeholder-shown)):not(:has(:invalid)) + #uitslag #met-fout`. Het enige verschil: hier is `p#met-fout` aan het eind van de selector vervangen door `p#alles-goed`: de `<p>` met de melding dat alles goed is ingevuld.

**`display: block;`**

Toon de `<p>` met de melding 'Bravo! Alle twaalf goed!'

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

```
<!doctype html>
<html lang="nl">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
    initial-scale=1">
```

```

<meta name="description" content="Invullen van
    provinciehoofdsteden met markering en aantal goede en
    foute - voorbeeld. Gebruikt alleen html en css">
<title>Invullen van provinciehoofdsteden met markering en
    aantal goede en foute - voorbeeld</title>
<link rel="stylesheet" href="009-css-dl/tekst-009-dl.css">
</head>
<body>
    <main>
        <h1 id="boven" tabindex="-1">Vul de
            provincie&shy;hoofdsteden in:</h1>
        <div id="wrapper">
            
            <div id=" groningen">
                <label for="gr">Gronin&shy;gen<span
                    class="sr-fout">Deze stad is
                    fout.</span><span class="sr-goed">Deze
                    stad is goed.</span><span class="fout"
                    aria-hidden="true">Fout!</span></label>
                <input id="gr" type="text" required
                    pattern=" *[Gg]roningen *" placeholder="">
                <span class="afk" aria-hidden="true">Gr:</span>
            </div>
            <div id="friesland">
                <label for="fr">Fries&shy;land<span
                    class="sr-fout">Deze stad is
                    fout.</span><span class="sr-goed">Deze
                    stad is goed.</span><span class="fout"
                    aria-hidden="true">Fout!</span></label>
                <input id="fr" type="text" required
                    pattern=" *[Ll]eeuwarden " placeholder="">
                <span class="afk" aria-hidden="true">Fr:</span>
            </div>

```

```

<div id="drenthe">
  <label for="dr">Drenthe<span
    class="sr-fout">Deze stad is
    fout.</span><span class="sr-goed">Deze
    stad is goed.</span><span class="fout"
    aria-hidden="true">Fout!</span></label>
  <input id="dr" type="text" required
    pattern=" *[Aa]ssen *" placeholder="">
  <span class="afk" aria-hidden="true">Dr:</span>
</div>
<div id="overijssel">
  <label for="ov">Over&shy;ijssel<span
    class="sr-fout">Deze stad is
    fout.</span><span class="sr-goed">Deze
    stad is goed.</span><span class="fout"
    aria-hidden="true">Fout!</span></label>
  <input id="ov" type="text" required
    pattern=" *[Zz]wolle *" placeholder="">
  <span class="afk" aria-hidden="true">Ov:</span>
</div>
<div id="flevoland">
  <label for="fl">Flevo&shy;land<span
    class="sr-fout">Deze stad is
    fout.</span><span class="sr-goed">Deze
    stad is goed.</span><span class="fout"
    aria-hidden="true">Fout!</span></label>
  <input id="fl" type="text" required
    pattern=" *[Ll]elystad *" placeholder="">
  <span class="afk" aria-hidden="true">Fl:</span>
</div>
<div id=" gelderland">
  <label for="ge">Gelder&shy;land<span
    class="sr-fout">Deze stad is
    fout.</span><span class="sr-goed">Deze
    stad is goed.</span><span class="fout"
    aria-hidden="true">Fout!</span></label>
  <input id="ge" type="text" required
    pattern=" *[Aa]rnhem *" placeholder="">
  <span class="afk" aria-hidden="true">Ge:</span>
</div>
<div id="utrecht">
  <label for="ut">Utrecht<span
    class="sr-fout">Deze stad is
    fout.</span><span class="sr-goed">Deze
    stad is goed.</span><span class="fout"
    aria-hidden="true">Fout!</span></label>
  <input id="ut" type="text" required
    pattern=" *[Uu]trecht *" placeholder="">

```

```

        <span class="afk" aria-hidden="true">Ut:</span>
    </div>
    <div id="noord-holland">
        <label for="nh">Noord-Holland<span
            class="sr-fout">Deze stad is
            fout.</span><span class="sr-goed">Deze
            stad is goed.</span><span class="fout"
            aria-hidden="true">Fout!</span></label>
        <input id="nh" type="text" required
            pattern=" *[Hh]aarlem *" placeholder="">
        <span class="afk" aria-hidden="true">NH:</span>
    </div>
    <div id="zuid-holland">
        <label for="zh">Zuid-Holland<span
            class="sr-fout">Deze stad is
            fout.</span><span class="sr-goed">Deze
            stad is goed.</span><span class="fout"
            aria-hidden="true">Fout!</span></label>
        <input id="zh" type="text" required
            pattern=" *[Dd]en [Hh]aag *|
            *'s-[Gg]ravenhage *" placeholder="">
        <span class="afk" aria-hidden="true">ZH:</span>
    </div>
    <div id="zeeland">
        <label for="ze">Zeeland<span
            class="sr-fout">Deze stad is
            fout.</span><span class="sr-goed">Deze
            stad is goed.</span><span class="fout"
            aria-hidden="true">Fout!</span></label>
        <input id="ze" type="text" required
            pattern=" *[Mm]iddelburg *"
            placeholder="">
        <span class="afk" aria-hidden="true">Ze:</span>
    </div>
    <div id="noord-brabant">
        <label for="nb">Noord-Brabant<span
            class="sr-fout">Deze stad is
            fout.</span><span class="sr-goed">Deze
            stad is goed.</span><span class="fout"
            aria-hidden="true">Fout!</span></label>
        <input id="nb" type="text" required
            pattern=" *[Dd]en [Bb]osch *|
            *'s-[Hh]ertogenbosch *" placeholder="">
        <span class="afk" aria-hidden="true">NB:</span>
    </div>

```

```

        <div id="limburg">
            <label for="li">Limburg<span
                class="sr-fout">Deze stad is
                fout.</span><span class="sr-goed">Deze
                stad is goed.</span><span class="fout"
                aria-hidden="true">Fout!</span></label>
            <input id="li" type="text" required
                pattern=" *[Mm]aastricht *"
                placeholder="">
            <span class="afk" aria-hidden="true">Li:</span>
        </div>
        <a id="naar-uitslag" href="#uitslag">Naar de
            uitslag</a>
        <a id="naar-boven" href="#boven">Terug naar
            boven</a>
    </div>
    <div id="uitslag" tabindex="-1">
        <p id="met-fout">Goed: <span id="goed"></span>.
        <span
            id="fout-of-leeg">Fout of nog niet ingevuld:
            </span><span id="fout">Fout: </span><span
            id="leeg">. Nog niet ingevuld: </span>.</p>
        <p id="alles-goed">Bravo! Alle twaalf goed!</p>
    </div>
</main>
</body>
</html>

```

## CSS

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

```

/* tekst-009-dl.css */

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

main {counter-reset: goed fout leeg 12;}

h1 {
    background: white;
    color: black;
    box-sizing: border-box;

```

```
width: 700px;
max-width: 94vw;
font-size: 1.2em;
font-weight: normal;
text-align: center;
margin: 10px auto;
border: black solid 1px;
padding: 5px;
}
```

```
#wrapper {
  width: 700px;
  max-width: 94vw;
  margin: 0 auto;
  position: relative;
}
```

```
#kaart {
  box-sizing: border-box;
  max-width: 100%;
  border: black solid 1px;
}
```

```
label {
  font-size: 0.8em;
  text-align: center;
  position: absolute;
}
```

```
input {
  background: white;
  color: black;
  width: 2em;
  height: 1.2em;
  font-size: 0.8em;
  margin-left: 3.5em;
  border: black solid 1px;
  border-radius: 5px;
  position: absolute;
  transition: width 0.5s, margin 0.5s;
}
```

```
# groningen label {
  width: 3em;
  bottom: 80%;
  left: 74%;
}
```

```
# groningen input {
    margin-bottom: 10px;
    margin-left: calc(3.5em + 30px);
    bottom: calc(80% - 2em);
    left: calc(74% - 3.1em);
}

# friesland label {
    width: 3em;
    right: 28%;
    bottom: 75%;
}

# friesland input {
    margin-right: 3.5em;
    margin-left: 0;
    right: calc(28% - 3.1em);
    bottom: calc(75% - 2em);
}

# drenthe label {
    top: 23%;
    left: 74%;
}

# drenthe input {
    top: calc(23% + 1.5em);
    left: calc(74% - 3.1em);
}

# overijssel label {
    width: 3em;
    bottom: 52%;
    left: 72%;
}

# overijssel input {
    bottom: calc(52% - 2em);
    left: calc(72% - 1.4em);
}

# flevoland label {
    width: 3em;
    bottom: 56%;
    left: 48%;
}
```

```
#flevoland input {
    bottom: calc(56% - 2em);
    left: calc(48% - 3.1em);
}

#gelderland label {
    width: 4em;
    top: 48%;
    left: 64%;
}

#gelderland input {
    top: calc(47% + 2.8em);
    left: calc(64% - 2.6em);
}

#utrecht label {
    top: 50%;
    left: 40%;
}

#utrecht input {
    top: calc(50% + 1.6em);
    left: calc(40% - 3.1em);
}

#noord-holland label {
    width: 4em;
    right: 52%;
    bottom: 60%;
}

#noord-holland input {
    margin-right: 3.5em;
    margin-left: 0;
    right: calc(52% - 2.8em);
    bottom: calc(60% - 2em);
}

#zuid-holland label {
    width: 4em;
    right: 60%;
    bottom: 42%;
}

#zuid-holland input {
    margin-right: 3.5em;
    margin-left: 0;
```

```

        right: calc(60% - 2.6em);
        bottom: calc(42% - 2em);
    }

#zeeland label {
    width: 4em;
    right: 70%;
    bottom: 30%;
}

#zeeland input {
    margin-right: 3.5em;
    margin-left: 0;
    right: calc(70% - 2.8em);
    bottom: calc(30% - 2em);
}

#noord-brabant label {
    top: 64%;
    left: 36%;
}

#noord-brabant input {
    top: calc(64% + 1.6em);
    left: calc(36% - 1.4em);
}

#limburg label {
    top: 80%;
    left: 56%;
}

#limburg input {
    top: calc(80% + 1.6em);
    left: calc(56% - 2.6em);
}

span {display: none;}

.sr-fout, .sr-goed {
    position: absolute;
    top: -1000px;
    left: -20000px;
}

.fout {
    background: rgb(255 255 255 / 0.7);
    color: black;
}

```

```

    width: 4ch;
    font-size: 0.7em;
    margin: 0 auto;
    border: red solid 1px;
    padding: 2px;
    position: absolute;
    top: 1em;
    right: 0;
    left: 0;
}

.afk {
    background: white;
    color: black;
    box-sizing: border-box;
    height: 1.2rem;
    font-size: 0.8em;
    border: black solid;
    border-width: 1px 0 1px 1px;
    padding: 1px 20px 1px 2px;
    position: absolute;
    left: 0;
}

div:nth-of-type(even) .afk {left: 54%;}

#wrapper div:has(:focus) label {
    background: rgb(255 255 255 / 0.8);
    color: black;
    font-weight: bold;
    z-index: 20;
}

#wrapper input:focus {
    width: 9em;
    margin: 0;
    border-radius: 0;
    z-index: 20;
}

#wrapper input:user-valid {
    box-sizing: border-box;
    width: calc(46% - 1.5rem);
    height: 1.2rem;
    margin: 0;
    border-left-width: 0;
    border-radius: 0;
    left: 1.8em;
}

```

```

    z-index: 10;
    transition-property: width, border-left, top, right, bottom,
        left;
    transition-duration: 2s, 0s, 2s, 2s, 2s, 2s;
    transition-timing-function: ease-in;
    transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;
}

#wrapper div:nth-of-type(even) input:user-valid {
    right: 0;
    left: auto;
}

#wrapper div:has(:user-valid) label {
    background: lightgreen;
    color: black;
    outline: darkgreen solid 2px;
    padding: 1px;
}

input:user-valid ~ .afk {display: block;}

# groningen input:user-valid, # groningen input:user-valid ~ .afk,
    # friesland input:user-valid, # friesland input:user-valid
    ~ .afk {bottom: 0;}

#drenthe input:user-valid, #drenthe input:user-valid ~ .afk {top:
    calc(100% + 0.4em);}

#overijssel input:user-valid {bottom: -2em; left: calc(54% +
    1.9em);}

#overijssel input:user-valid ~ .afk {bottom: -2em;}

#flevoland input:user-valid, #flevoland input:user-valid ~ .afk
    {bottom: -3.8em;}

#gelderland input:user-valid {
    top: calc(100% + 2.4em);
    left: calc(54% + 1.9em);
}

#gelderland input:user-valid ~ .afk {top: calc(100% + 2.4em);}

#utrecht input:user-valid, #utrecht input:user-valid ~ .afk {top:
    calc(100% + 4.2em);}

```

```
#noord-holland input:user-valid, #noord-holland input:user-valid ~
    .afk {bottom: -5.8em;}

#zuid-holland input:user-valid {
    right: 54%;
    bottom: -7.8em;
    left: auto;
}

#zuid-holland input:user-valid ~ .afk {bottom: -7.8em;}

#zeeland input:user-valid, #zeeland input:user-valid ~ .afk
    {bottom: -7.8em;}

#noord-brabant input:user-valid {
    top: calc(100% + 8.4em);
    left: 1.9em;
}

#noord-brabant input:user-valid ~ .afk {top: calc(100% + 8.4em);}

#limburg input:user-valid {
    top: calc(100% + 8.4em);
    left: calc(54% + 1.9em);
}

#limburg input:user-valid ~ .afk {bottom: -9.9em;}

#wrapper div:has(:user-invalid):not(:has(:focus)) .fout {display:
    block;}

#wrapper div:has(:user-invalid) .sr-fout {display: block;}

#wrapper div:has(:user-valid) .sr-goed {display: block;}

#naar-uitslag, #naar-boven {
    background: white;
    color: black;
    display: none;
    box-sizing: border-box;
    width: 700px;
    max-width: 94vw;
    font-size: 0.8em;
    text-align: center;
    margin-bottom: 20em;
    border: black solid 1px;
    padding: 3px;
    position: relative;
```

```

        top: 12.4em;
    }

#naar-boven {display: block;}

#uitslag {display: none;}

@supports selector(:has(*)) {
    h1 {padding-bottom: 2.2em;}
    #uitslag {
        display: block;
        width: 700px;
        max-width: 94vw;
        font-size: 0.9em;
        text-align: center;
        margin: 0 auto;
        scroll-margin-top: 3.5em;
        position: absolute;
        top: 3.8em;
        right: 0;
        left: 0;
    }
    #alles-goed {
        background: lightgreen;
        color: black;
        display: none;
        font-size: 1.2em;
        margin: 0;
        border: black solid 1px;
        padding: 3px;
    }
    #met-fout {margin: 0;}
    #naar-uitslag {display: block;}
    #naar-boven {display: none;}
}

#goed, #fout, #leeg {display: inline;}

input:user-invalid {counter-increment: fout leeg -1;}

input:user-valid {counter-increment: goed leeg -1;}

#goed::after {content: counter(goed);}

```

```

@supports not selector(:user-valid) {
  #wrapper input:valid {
    box-sizing: border-box;
    width: calc(46% - 1.5rem);
    height: 1.2rem;
    margin: 0;
    border-left-width: 0;
    border-radius: 0;
    left: 1.8em;
    z-index: 10;
    transition-property: width, border-left, top, right,
      bottom, left;
    transition-duration: 2s, 0s, 2s, 2s, 2s, 2s;
    transition-timing-function: ease-in;
    transition-delay: 0s, 2s, 0s, 0s, 0s, 0s;
  }
  #wrapper div:nth-of-type(even) input:valid {
    right: 0;
    left: auto;
  }
  #wrapper div:has(:valid) label {
    background: lightgreen;
    color: black;
    outline: darkgreen solid 2px;
    padding: 1px;
  }
  input:valid ~ .afk {display: block;}
  # groningen input:valid, # groningen input:valid ~ .afk,
    # friesland input:valid, # friesland input:valid ~ .afk
    {bottom: 0;}
  #drenthe input:valid, #drenthe input:valid ~ .afk {top:
    calc(100% + 0.4em);}
  #overijssel input:valid {
    bottom: -2em;
    left: calc(54% + 1.9em);
  }
  #overijssel input:valid ~ .afk {bottom: -2em;}
  #flevoland input:valid, #flevoland input:valid ~ .afk {bottom:
    -3.8em;}
  # gelderland input:valid {
    top: calc(100% + 2.4em);
    left: calc(54% + 1.9em);
  }
  # gelderland input:valid ~ .afk {top: calc(100% + 2.4em);}
  # utrecht input:valid, # utrecht input:valid ~ .afk {top:
    calc(100% + 4.2em);}
  # noord-holland input:valid, # noord-holland input:valid ~ .afk
    {bottom: -5.8em;}

```

```

#zuid-holland input:valid {
    right: 54%;
    bottom: -7.8em;
    left: auto;
}
#zuid-holland input:valid ~ .afk {bottom: -7.8em;}
#zeeland input:valid, #zeeland input:valid ~ .afk {bottom:
    -7.8em;}
#noord-brabant input:valid {
    top: calc(100% + 8.4em);
    left: 1.9em;
}
#noord-brabant input:valid ~ .afk {top: calc(100% + 8.4em);}
#limburg input:valid {
    top: calc(100% + 8.4em);
    left: calc(54% + 1.9em);
}
#limburg input:valid ~ .afk {bottom: -9.9em;}
input:valid {counter-increment: goed leeg -1;}
#fout, #leeg {display: none;}
#fout-of-leeg {display: inline;}
#fout-of-leeg::after {content: counter(leeg);}
}

#fout::after {content: counter(fout);}

#leeg::after {content: counter(leeg);}

@media screen and (min-width: 500px) {
    # groningen label {
        bottom: 84%;
        left: 76%;
    }
    # groningen input {
        bottom: calc(84% - 2em);
        left: calc(76% - 3.1em);
    }
    # friesland label {
        right: 32%;
        bottom: 79%;
    }
    # friesland input {
        right: calc(32% - 3.1em);
        bottom: calc(79% - 2em);
    }
}

```

```

#overijssel label {
    width: auto;
    bottom: 58%;
}
#overijssel input {
    bottom: calc(58% - 2em);
    left: calc(74% - 3.1em);
}
#flevoland label {left: 52%;}
#flevoland input {
    bottom: calc(56% - 2em);
    left: calc(52% - 3.1em);
}
#gelderland label {
    width: auto;
    top: 49%;
    left: 62%;
}
#gelderland input {
    top: calc(49% + 1.7em);
    left: calc(62% - 2.2em);
}
#utrecht label {left: 44%;}
#utrecht input {left: calc(44% - 3.1em);}
#noord-holland label {right: 56%;}
#noord-holland input {right: calc(56% - 2.7em);}
#zuid-holland label {right: 62%;}
#zuid-holland input {right: calc(62% - 2.7em);}
#zeeland label {
    width: 4em;
    right: 74%;
    bottom: 30%;
}
#zeeland input {
    right: calc(74% - 2.8em);
    bottom: calc(30% - 2em);
}
#noord-brabant label {left: 40%;}
#noord-brabant input {left: calc(40% - 1.4em);}
#limburg label {top: 83%; left: 60%;}
#limburg input {
    top: calc(83% + 1.6em);
    left: calc(60% - 2.8em);
}
#wrapper div:has(:focus) label {
    font-size: 1em;
    font-weight: normal;
}

```

```
}

@media screen and (min-width: 760px) {
  body {font-size: 110%;}
  #wrapper div {
    display: flex;
    flex-wrap: wrap;
    align-content: center;
    justify-content: center;
    position: absolute;
  }
  # groningen {
    width: 20%;
    height: 20%;
    top: 4%;
    left: 72%;
  }
  #friesland {
    width: 19%;
    height: 22%;
    top: 9%;
    left: 53%;
  }
  #drenthe {
    width: 18%;
    height: 13%;
    top: 21%;
    left: 72%;
  }
  #overijssel {
    width: 24%;
    height: 14%;
    top: 33%;
    left: 67%;
  }
  #flevoland {
    width: 14%;
    height: 10%;
    top: 36%;
    left: 47%;
  }
  # gelderland {
    width: 28%;
    height: 12%;
    top: 50%;
    left: 56%;
  }
}
```

```
#utrecht {
    width: 17%;
    height: 11%;
    top: 46%;
    left: 39%;
}
#noord-holland {
    width: 17%;
    height: 30%;
    top: 20%;
    left: 30%;
}
#zuid-holland {
    width: 17%;
    height: 16%;
    top: 46%;
    left: 22%;
}
#zeeland {
    width: 20%;
    height: 16%;
    top: 59%;
    left: 7%;
}
#noord-brabant {
    width: 33%;
    height: 14%;
    top: 62%;
    left: 27%;
}
#limburg {
    width: 11%;
    height: 32%;
    top: 62%;
    left: 58%;
}
#wrapper label {
    width: auto;
    font-size: 1em;
    position: static;
}
#wrapper div:has(:focus) label {
    background: transparent;
    font-size: 1.5em;
}
.fout {position: static;}
#wrapper input, #wrapper input:valid {
    background: rgb(255 255 255 / 0.8);
```

```
        color: black;
        width: 9em;
        height: auto;
        margin: 0;
        border-left: black solid 1px;
        border-radius: 5px;
        position: static;
    }
    #wrapper .afk {display: none;}
    #naar-uitslag, #naar-boven {
        font-size: 1em;
        margin-bottom: 30px;
        top: 6px;
    }
}

#wrapper:not(:has(:placeholder-shown)):not(:has(:invalid)) +
    #uitslag #met-fout {display: none;}

#wrapper:not(:has(:placeholder-shown)):not(:has(:invalid)) +
    #uitslag #alles-goed {display: block;}
```