BASIS CSS

Inhoudsopgave

Een introductie in het vormgeven van je eigen website

CSS - de visuele opmaak van HTML
CSS staat voor Cascading Style Sheets. Het wordt gebruikt om HTML visueel vorm te geven. De opmaak staat dan in een los bestand (bijvoorbeeld opmaak.css), gescheiden van de content. Een van de voordelen hiervan is dat de vormgeving van de gehele site centraal aanpasbaar is. Een ander voordeel is dat je vormgeving media-afhankelijk kan zijn: dezelfde site kan er op mobiel, tablet, desktop of geprint telkens anders uitzien.

Over CSS

CSS beschrijft de opmaak van de HTML code. Door middel van de CSS komt de website er pas echt mooi uit te zien.

CSS gebruikt vele eigenschappen, die je allemaal waardes kan geven. Dit kan gebeuren in het HTML bestand (*inline,* al eerder besproken bij het onderdeel HTML) maar wordt meestal gedaan in een apart CSS bestand. De syntax daarvan ziet er zo uit:

```
element {
     eigenschap: waarde;
}
```

Eerst het *element* onderdeel. Dit laat het element (onderdeel) zien dat je wilt stijlen. Dit kan gaan over alle elementen van een bepaalde categorie, bijvoorbeeld alle *divs.* Dat zou er dus zo uitzien:

```
div {
     eigenschap: waarde;
}
```

Id's en classes

Nou is dat niet altijd even handig, soms wil je een specifiek element stijlen, of een groep met zelf bepaalde elementen. Dat kan, door middel van:

- *ids* (1 element die een bepaald id kan hebben) en
- *classes* (groep elementen die een bepaalde class kunnen hebben).

Een element kan meerdere *classes* hebben en maar 1 *id.* In de HTML geef je *ids* en *classes* als volgt aan een element*:*

```
<div id="naam" class="naam"></div>
```

Je kunt een *id* en een *class* tegelijk hebben op een element, maar dat hoeft niet. Je kan het je als volgt inbeelden: soms wil je dat een element dezelfde opmaak heeft als een groep andere elementen, maar afwijkt op 1 bepaalde eigenschap. Dan gebruik je dus een *class* om het dezelfde opmaak te geven als de andere elementen en een *id* om een stukje specifieke opmaak te geven.

Wanneer je een *class* of *id* hebt gegeven aan een element, kan je dat als volgt opvragen in CSS:

```
#naam {
     eigenschap: waarde;
}
```

bij een id

```
.naam {
     eigenschap: waarde;
}
```

bij een class

Je kan ook een element selecteren die in een bepaald element zit. Wanneer je bijvoorbeeld elke *h1* wilt selecteren die zich bevindt in een *div,* doe je dat als volgt:

```
div h1 {
     eigenschap: waarde;
}
```

Eigenschappen

Nu je weet hoe je elementen kan selecteren, is het behoorlijk belangrijk om ook te weten welke eigenschappen er zijn. Er is een hele waslijst aan eigenschappen, hieronder staan de meest gebruikte. Je kan per selector meerdere eigenschappen aanpassen natuurlijk.

Zorg er gewoon voor dat je de ; gebruikt.

Voor deze lijst is het belangrijk om te beseffen dat je vele eigenschappen verschillende type waardes kan geven;

- in pixels of
- procenten.

In het geval dat je bij bijvoorbeeld een breedte gebruik maakt van pixels (zal je later zien bij die waardes), betekent het dus dat het op elk apparaat dat aantal pixels breed is, hoeveel pixels dat apparaat ook bevat.

Bij procenten gaat het uit van hoeveel ruimte het bovenliggende element heeft. Stel je gebruikt een *div* waar een plaatje in staat die een width heeft van 100% (hele scherm) en de breedte van het plaatje is 50%, is de breedte van het plaatje dus de helft van het scherm. Dit zorgt er dus voor dat dit beter zou werken op een telefoon, omdat je dan relatief werkt en niet absoluut. Als je dit begrijpt; hier het lijstje met allemaal veelgebruikte eigenschappen:

Color:

verandert kleur van tekst (kan ook op *div*, geldt dan voor alle tekst in die *div),* je kan gebruik maken van hexadecimale waardes, rgb of de naam van de kleur.

Bijvoorbeeld:

```
.testclass {
	color: #fffff;
}
```

Background-color:

de achtergrondkleur van dat stukje, je kan gebruik maken van hexadecimale waardes, rgb of de naam van de kleur

Bijvoorbeeld:

```
.testclass {
	background-color: rgb(210, 104, 82);
}
```

Height:

de hoogte van een object, je kan gebruik maken van % of px

Bijvoorbeeld:

```
#testid {
      height: 40%;
}
```

Width:

de breedte van object, je kan gebruik maken van % of px

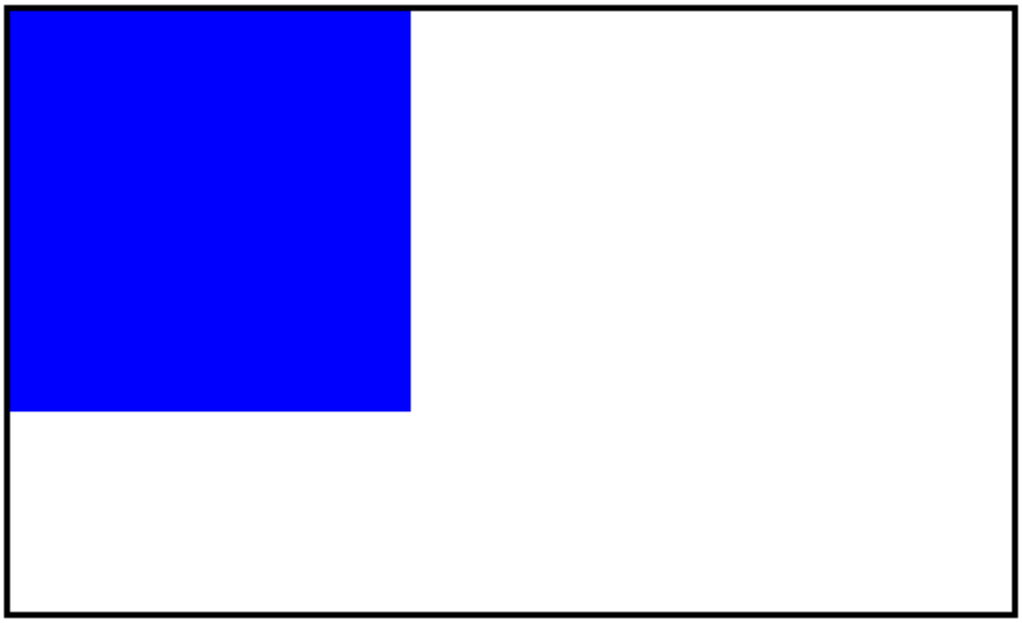
Bijvoorbeeld:

```
#testid {
      width: 60px;
}
```

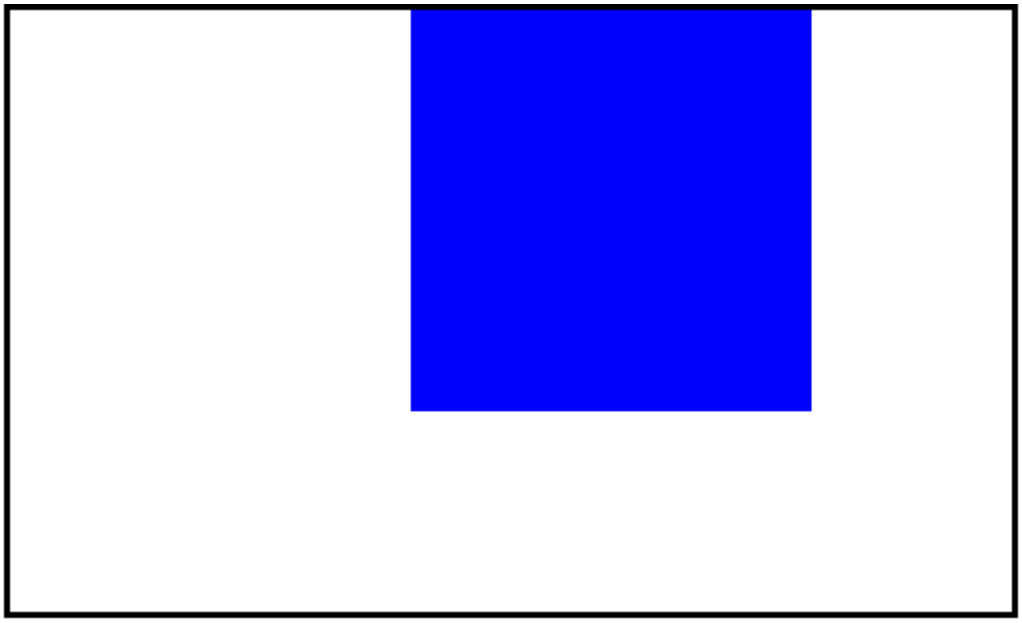
Position:

Bepaalt de plaatsing, de positie van een object. Je kan daar gebruik maken van deze waardes:

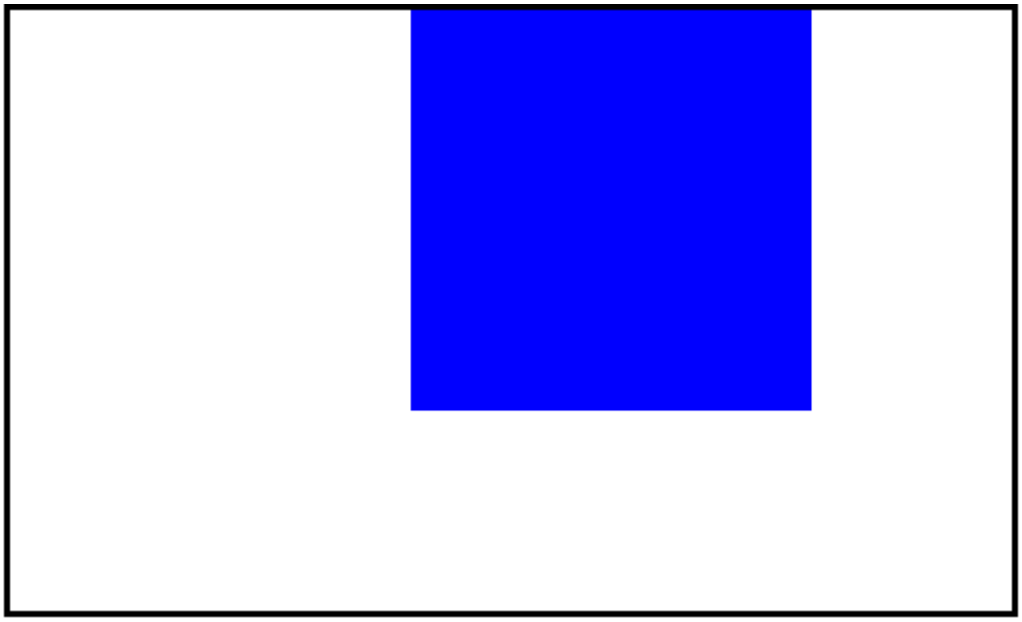
- Static = standaard, niet speciaal gepositioneerd:



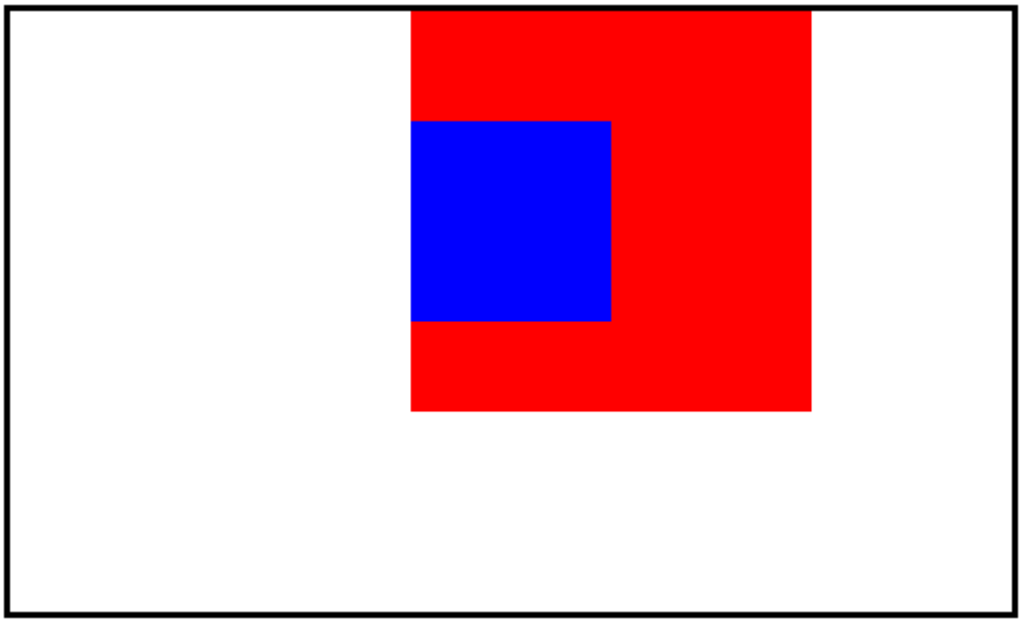
- Relative = relatief ten opzichte van normale positie (in het voorbeeld met een waarde *left: 200px,* leer je verderop meer over)



- Fixed = relatief tot het beeld waarop je het ziet (viewport), dus staat stil, ook met scrollen (in het voorbeeld met een waarde *left: 200px,* leer je verderop meer over, maar is exact hetzelfde als vorige voorbeeld, het verschil is dat het bij het scrollen nu stil blijft staan)



- Absolute = relatief tot een in de hiërarchie bovenliggend gepositioneerd object (alles behalve static). In het voorbeeld is het blauwe de *div* met de *absolute* waarde, met een *top: 50px*:



Bijvoorbeeld:

```
#testid {
     position: relative;
}
```

Right, left, top, bottom:

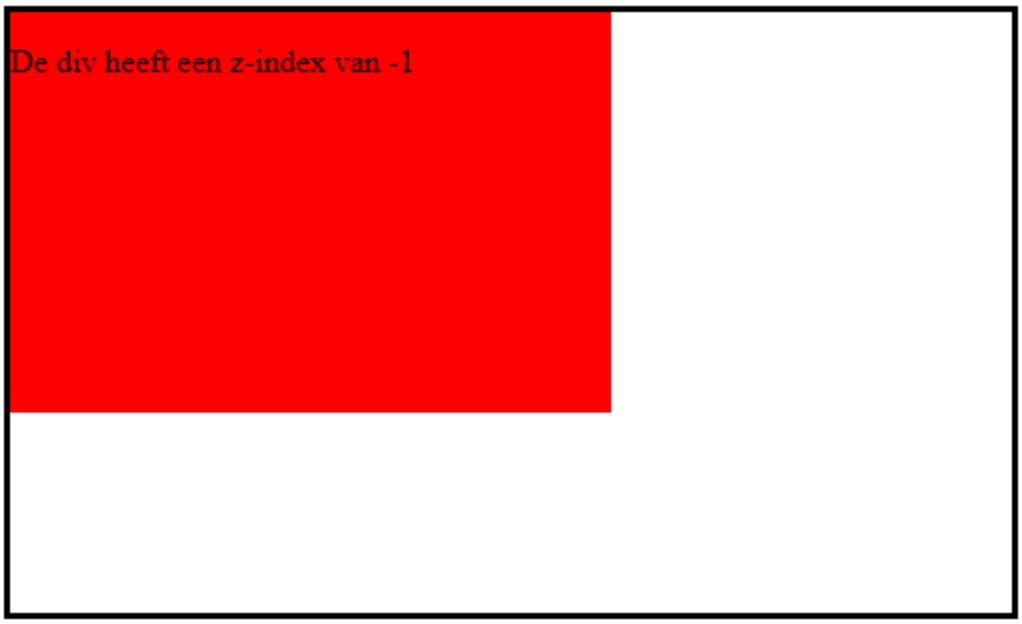
Bepaalt de afstand vanaf die kant, dus bij top: 10px betekent 10px vanaf de bovenkant. Werkt alleen bij gepositioneerde elementen, je kan gebruik maken van % of px.

Bijvoorbeeld:

```
#testid {
      position: relative;
      right: 60px;
}
```

Z-index:

Bepaalt welk object boven de ander staat, hoe hoger de z-index hoe hoger het staat in de visibiliteit rang:



Hier staat de tekst niet binnen de *div,* maar gewoon erbuiten. Vanwege de z-index staat de tekst boven de *div.*

Er geldt simpelweg het hoogste getal staat op voorgrond, dan des te lager des minder op de voorgrond. Werkt alleen als het om een gepositioneerd object gaat, dus niet bij een object met static positie

Bijvoorbeeld:

```
#testid {
     position: absolute;
     z-index: 999;
}
```

Display:

Bepaalt hoe en of een element wordt weergegeven. De standaard waarde is *block* (zichtbaar). Je kan hem ook de waarde *none* geven, dan wordt het niet meer zichtbaar.

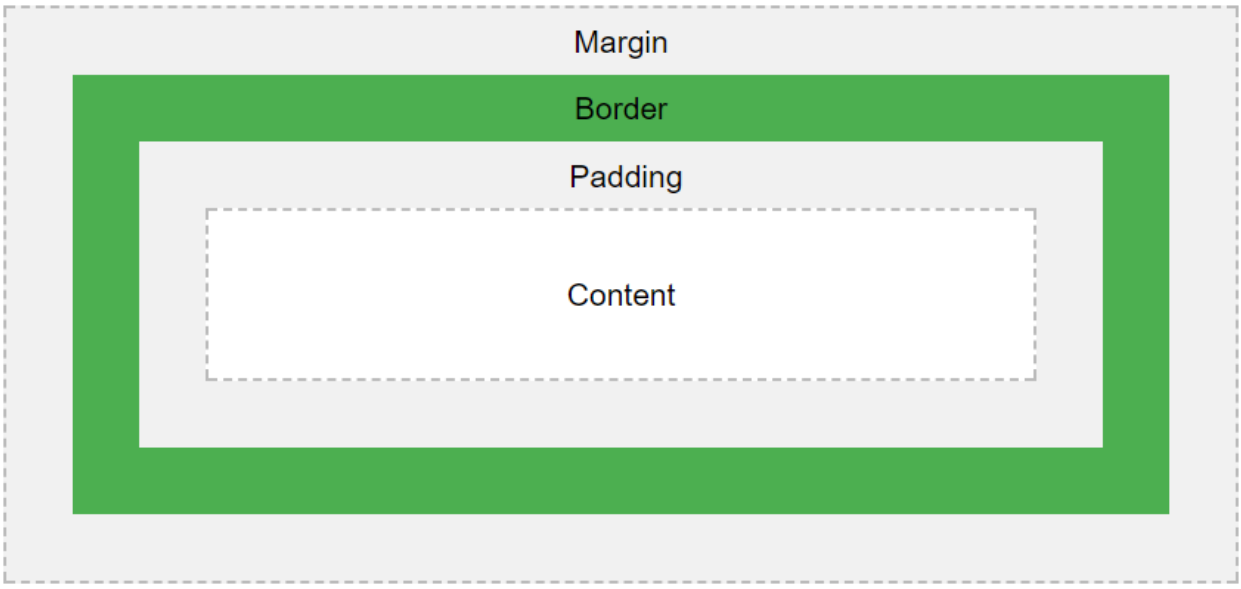
Een andere mogelijkheid is het gebruiken van de eigenschap *visibility* en daar de waarde op *hidden* zetten. Wat er dan gebeurt is dat het element inderdaad wordt verborgen, maar nog steeds ruimte inneemt. Bij *display: none* wordt er geen extra ruimte gebruikt.

Bijvoorbeeld:

```
#testid {
     display: none;
}
```

Het box-model

Het box-model wordt gebruikt wanneer het gaat om ruimte om inhoud heen. Box-model is gewoon de naam van de onderstaande illustratie:



In dit geval is de *content* de inhoud, de *padding* is de ruimte om deze inhoud heen, dan komt de *border*. De *border* is de buitenkant van de inhoud. Daar omheen zit nog *margin,* wat zorgt dat andere objecten niet in de buurt kunnen komen. Wanneer de *padding* bijvoorbeeld 0 is zal je zien dat de *border* direct om de inhoud heen zit.

Border is ook een CSS eigenschap, waarover je hier meer kan lezen.

Margin en padding

Margin en padding kan je met % en px aangeven. Wanneer je de pagina responsive wilt maken, wat inhoudt dat de pagina op de telefoon en PC goed te bekijken is, moet je gebruik maken van %.

Je kan het gebruiken als margin-top, margin-right, margin-bottom, margin-left, dus 4 aparte waardes, of als 1 waarde. Dan doe je het volgende; margin: ###px ###px ###px ###px. Dit gaat op de volgorde zoals net genoemd, top, right, bottom, left (Noord Oost Zuid West). Al ditzelfde geldt ook voor padding. Hierbij 2 voorbeelden die allebei exact hetzelfde doen:

```
#testid {
     margin: 40px 50px 60px 70px;
}

#testid {
     margin-top: 40px;
     margin-right: 50px;
     margin-bottom: 60px;
     margin-left: 70px;
}
```

Het voordeel aan het tweede voorbeeld is het feit dat je ook maar 1 waarde kan instellen, als dat is wat je wilt.

Kolommen met columns

Margin en padding kan je met % en px aangeven. Wanneer je de pagina responsive wilt maken, wat inhoudt dat de pagina op de telefoon en PC goed te bekijken is, moet je gebruik maken van %.

In moderne browsers kan je ook *kolommen met CSS3 columns maken.* Je zet al je kolommen (divs) tezamen in een container-div. Met *colums* geef je het aantal kolommen aan, en eventueel de minimale breedte voor de kolommen, in pixels. Als het venster kleiner wordt komen de kolommen onder elkaar. Het is meteen al responsive, zonder extra css.

```
.kol3 {
        columns: 3 200px;       /* 3 kolommen van minstens 200px breed */
        column-gap: 2em;        /* optioneel, default is 1em */
   }
```

De ruimte tussen de kolommen is die *column-gap*, maar die kun je ook weglaten. In de HTML staat er simpelweg dit:

```
<div class="kol3">
        <div> Inhoud van Kolom 1 </div>
        <div> Inhoud van Kolom 2 </div>
        <div> Inhoud van Kolom 3 </div>
</div>
```

Kolommen met flexbox

Een andere manier om moderne responsive kolommen met CSS3 te maken is *Flexbox.* Hier een *goed voorbeeld daarvan*. Resize de browser (of tik Cmd+) om het responsive gedeelte te zien.

Overig

Een aantal zaken zijn handig om te weten, namelijk parallax en hoe je een navigatiebalk maakt.

Navigatiebalk

Eerst maar de navigatiebalk . Dit is eigenlijk vrij makkelijk. Hier een aantal ideeën daarvoor.

Ten eerste is het handig om deze navigatiebalk de *fixed* positie te geven. Zo blijft deze altijd in het scherm staan en hoeft de gebruiker niet telkens terug te scrollen om de navigatiebalk te vinden. Dan zorg je voor een aantal links.
Je kan ook in de CSS *:hover* achter de element selector zetten, dat zorgt ervoor dat je het object een speciale opmaak kan geven wanneer er overheen wordt gehoverd.
Hier meer documentatie hierover.

Dit is eigenlijk alles wat nodig is om een simpele, duidelijke navigatiebalk te maken. Je zou ook animaties kunnen toevoegen mocht je dat willen, waar je hier weer meer over kan vinden.

Zie hier een voorbeeld van een basis horizontaal en verticaal menu.

Parallax

Ten tweede parallax, dat is wanneer 1 object beweegt en de andere stil blijft staan. Op deze website kan je dat goed zien. Dit is vrij eenvoudig te maken. Een belangrijk ding om bij stil te staan is het feit dat je een *div* maakt met een achtergrondfoto. Voor een achtergrondfoto bestaat er ook een CSS eigenschap; *background-image*. De achtergrondfoto maak je *fixed* en dan blijft de afbeelding hangen op dat punt en nergens anders. Hier is de code die je dus aan een *div* toe moet voegen:

```
background-image: url("linkjenaarbestand.jpg");
background-attachment: fixed;
background-position: center;
background-repeat: no-repeat;
background-size: cover;
```

Voor een volledige lijst met alle CSS eigenschappen kan je hier kijken, waar alle eigenschappen staan benoemd met hun mogelijke waarden. Wanneer je werkt met CSS, is het het handigst om gewoon te googlen wat je wilt bereiken. Er is genoeg te vinden met oplossingen voor je probleem (wel in het Engels googlen!).

Extra

Hamburger menu

Op mobiel zie je tegenwoordig vaak een hamburger menu, ook wel off-screen menu genaamd. Die kan je met pure CSS maken, of met Javascript. Hier nog een. Meer uitleg in het Nederlands.

Multiple background-images in css

In de css van een background kan je meerdere waardes zetten, gescheiden met een komma, zodat je meerdere backgrounds in 1 div kan zetten.
De eerste waarde is voor het eerste plaatje. Als er maar 1 waarde staat, geldt die waarde voor beide plaatjes.
Het eerste plaatje komt bovenop de tweede, qua z-index.

```
background-image:url(bg2.png), url(bg.jpg);
background-size:5px 5px, cover;
background-attachment:fixed;
```

jQuery laden vanaf Google server

jQuery is een bibliotheek met Javascript-functies. Je kan deze jQuery library op je server neerzetten en aanroepen, of je kan die van Google gebruiken door dit in de <head> te zetten:

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></sc
ript>
```

Css-filters: o.a. kleur, zwart-wit, blur

Je *kan css filters* gebruiken om een plaatje te filteren, bijvoorbeeld in zwartwit of met een blur tonen (hier geanimeerd met een transition). In dit voorbeeld wordt het plaatje pas in kleur bij een hover:

```
.zw             { filter: grayscale(1); }
.zw:hover       { filter: grayscale(0); }
```

Noot: ipv (1) of (0.5) of (.5) kun je ook % gebruiken (100%) of (50%)
Noot 2: in dit voorbeeld zit er ook nog een opacity:0.4 *filter bij*
Noot 3: vendor prefixes zoals -webkit- zijn nog steeds nodig

In de HTML is er bij de img het attribuut class="zw" toegevoegd. Je kan ook ALLE images in een bepaalde div het filter meegeven:

```
#divnaam img { filter: grayscale(100%); }
```

Het tweede voorbeeld met de blur is vergelijkbaar
(hier even met -webkit-prefix):

```
.blur          { -webkit-filter: blur(0); transition: all .5s; }
.blur:hover    { -webkit-filter: blur(5px);}
```

Google Webdesigner

Google heeft het *gratis programma Webdesigner* gemaakt om eenvoudig en snel animaties te maken in HTML5 en CSS3.

En als laatste nog wat humor…

```
#wife {
  right: 100%;
  margin: 0;
}
```

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

```
.ninja {
  color: black;
  visibility: hidden;
  animation-duration: 0.00001s;
}
```

```
.ghost {
  color: white;
  opacity: 0.1;
}
```

Hier is meer te vinden…