



# Inleiding

Web Frontend Basics

## INHOUD

|  |           |
|--|-----------|
| <b>1 INLEIDING</b>                             | <b>3</b>  |
| <b>1.1 World Wide Web</b>                      | <b>3</b>  |
| 1.1.1 Kort stukje geschiedenis                 | 3         |
| 1.1.2 Enkele ‘wist je datjes’                  | 3         |
| 1.1.3 World Wide Web Consortium                | 3         |
| 1.1.4 Super strak design?                      | 4         |
| <b>1.2 Browser?</b>                            | <b>5</b>  |
| 1.2.1 Algemeen                                 | 5         |
| 1.2.2 Gekende browsers                         | 5         |
| 1.2.3 Browser werking                          | 6         |
| 1.1.1.1 Stap een                               | 6         |
| 1.1.1.2 Stap twee                              | 7         |
| 1.1.1.3 Stap drie                              | 7         |
| 1.1.1.4 Stap vier                              | 7         |
| 1.1.1.5 Stap vijf                              | 7         |
| 1.1.1.6 Stap zes                               | 7         |
| 1.1.1.7 Stap zeven                             | 8         |
| <b>1.3 Terminologie</b>                        | <b>8</b>  |
| 1.3.1 Hypertext Markup Language - HTML         | 8         |
| 1.3.2 Van HTML naar HTML5                      | 9         |
| 1.3.3 Cascading Style Sheets - CSS             | 9         |
| <b>1.4 Elementen, Attributen &amp; Waarden</b> | <b>11</b> |
| 1.4.1 HTML Elementen                           | 11        |
| 1.4.2 Attributen en waarden                    | 12        |
| <b>1.5 Nesten van elementen</b>                | <b>13</b> |
| <b>1.6 Editoren</b>                            | <b>14</b> |
| 1.6.1 VS Code                                  | 14        |
| 1.1.1.8 Installatie                            | 14        |
| 1.1.1.9 Nuttige Extensies                      | 16        |
| 1.1.1.10 Aanbevolen instellingen               | 17        |
| <b>1.7 HTML Validatie</b>                      | <b>18</b> |

## 1 INLEIDING

### 1.1 WORLD WIDE WEB

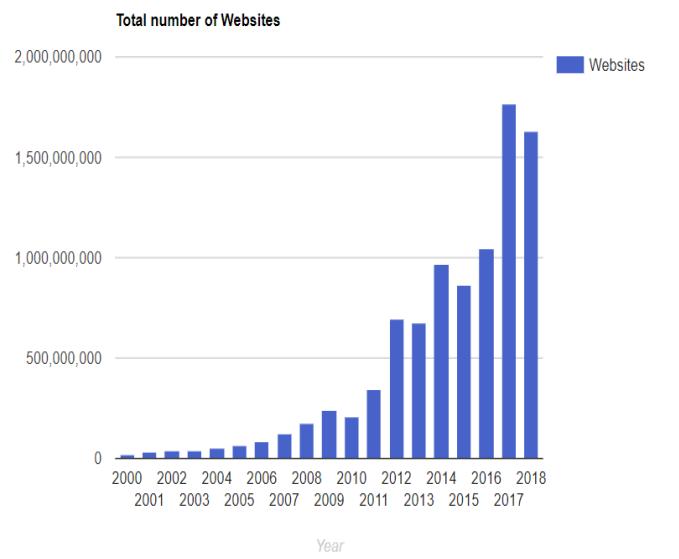
Het internet mag niet verward worden met de term “het wereld wijde web”. Het internet, een immense verzameling van aan elkaar gekoppelde computers, is slechts de basis voor vele diensten waaronder Usenet met zijn nieuwsgroepen, FTP voor bestandsoverdrachten, E-mail, Voice telefoon tussen computers, enzovoort. Het WWW is slechts een speler die gedragen wordt dankzij het bestaan van het internet.

#### 1.1.1 KORT STUKJE GESCHIEDENIS

Sinds de lancering van het world wide web, hiervoor gaan we ongeveer terug naar 1991, is het aantal beschikbare websites enkel maar exponentieel toegenomen.

De statistieken (figuur 1) brengen deze evolutie mooi in kaart van 2000 tot en met 2015. Vandaag de dag kan je via [www.internetlivestats.com](http://www.internetlivestats.com) de teller volgen van het aantal gevonden unieke websites.

Deze heeft inmiddels de kaap van 1.700.000.000 overschrijden. Kleine kanttekening hierbij is dat ongeveer 75% van de websites geen actieve sites zijn maar geparkeerde domeinen of gelijkaardige



1 bron: [internetlivestats.com](http://internetlivestats.com) ©

#### 1.1.2 ENKELE ‘WIST JE DATJES’

- De eerste website ([info.cern.ch](http://info.cern.ch)) werd gepubliceerd op 6 augustus 1991 door de Britse natuurkundige Tim Berners-Lee, werkzaam in CERN, Zwitserland.
- CERN maakte op 30 april 1993 de World Wide Web technologie gratis beschikbaar voor het grote publiek, waardoor het internet kon bloeien.
- In 2013 groeide het internet met meer dan een derde op een jaar tijd. Van ongeveer 630 miljoen websites aan het begin van het jaar tot meer dan 850 miljoen in december 2013.
- Meer dan 50 % van de websites van vandaag worden gehost op Apache of nginx, beide open source web servers. Sinds juni 2014 is Microsoft qua marktaandeel heel dicht bij Apache gekomen (*slechts een verschil van 0,15% scheidt de twee*). Als deze trend doorzet, zou Microsoft binnenkort voor het eerst in de geschiedenis de toonaangevende webserverontwikkelaar kunnen worden.

#### 1.1.3 WORLD WIDE WEB CONSORTIUM

In 1994 werd in samenwerking met het CERN het World Wide Web Consortium het ([W3C](http://www.w3c.org)) opgericht door Tim Berners-Lee. Deze organisatie heeft het doel om de ontwikkeling van webpagina's

eenduidig te maken voor alle browsers door een gemeenschappelijke standaard vast te leggen. Deze cursus houdt zich voornamelijk aan de regels van het W3C omdat ze streven naar een model waar een webpagina in alle browsers op dezelfde manier wordt weergegeven.

#### 1.1.4 SUPER STRAK DESIGN?

Sinds de lancering kende het internet niet alleen een grote groei maar evolueerde gaandeweg ook de factor ‘design’ mee met de tijd. Onderstaand een paar voorbeelden van hoe een aantal bekende websites er uitzagen toen ze voor het eerst gelanceerd werden.



Google ging live in 1998, Facebook in 2004. Zoals je zelf kan zien is er ondertussen heel wat veranderd aan het uitzicht van beide websites.



Als je zelf even wil rondneuzen hoe jouw favoriete website er vroeger uitzag, of je wil nog even voor nostalgische redenen terug in de tijd duiken?  
Dan kan je dit altijd even doen via [web.archive.org](http://web.archive.org)

## 1.2 BROWSER?

### 1.2.1 ALGEMEEN

Om al deze websites en hun webpagina's te bekijken, maken we gebruik van een browser. Nu, wat doet zo een browser precies? Een browser zet webpagina's, die door een webserver zijn aangeleverd, om in een vorm die voor mensen leesbaar is.

Een aantal elementen die je vaak terugvindt op een webpagina zijn bijvoorbeeld:

- Verschillende soorten opmaak van tekst
- Afbeeldingen
- Links, of doorverwijzingen, naar andere webpagina's
- Animaties
- Video's
- ...

Er zijn webbrowsers die documenten voorlezen, andere zetten ze om in puntjes op een braillemachine. Maar het grotendeel van de browsers geeft een webpagina weer op een computerscherm.

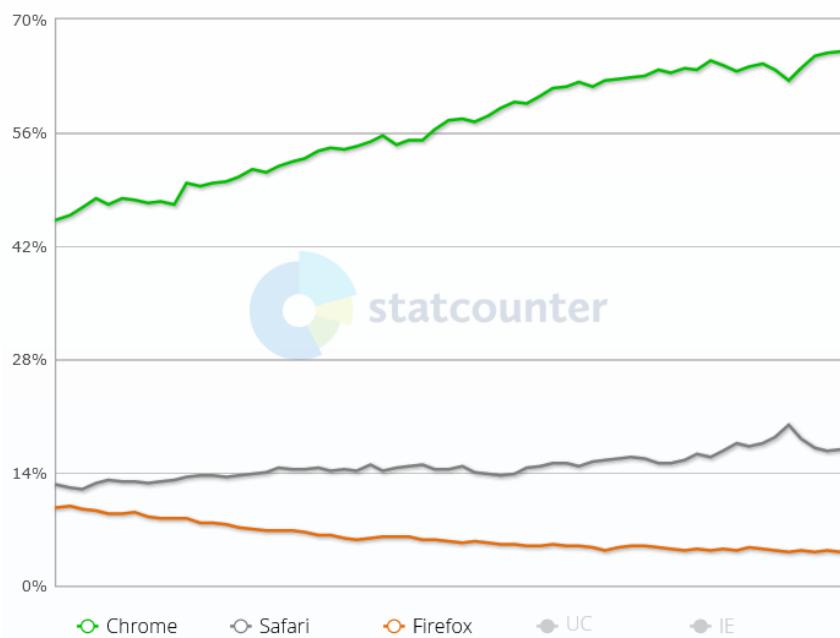
Vrijwel alle browsers hebben de mogelijkheid om weblocaties op te slaan (bladwijzers), bestanden te downloaden en een geschiedenis bij te houden van waar de gebruiker geweest is. Sommige browsers voegen hier nog een aantal extra's aan toe zoals meerdere tabbladen, pop-upblockers, advertentiefiltering en automatisch zoeken op een zoekmachine naar eigen voorkeur.

### 1.2.2 GEKENDE BROWSERS

Iedereen werkt ermee, maar staat veelal niet stil bij de keuze van een webbrowser. Vaak wordt gewoon genomen wat een vriend of kennis aanbeveelt of simpelweg wat er geïnstalleerd was op het toestel waarmee je aan de slag gaat.

Hieronder een overzicht van de top 3 gebruikte browsers wereldwijd.

Aug 2015 - Aug 2020



Wil je hier meer over weten, lezen, opzoeken?

We verwijzen je dan graag door naar de volgende website: [gs.statcounter.com](https://gs.statcounter.com)

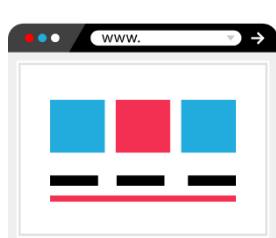
### 1.2.3 BROWSER WERKING

Om een correct beeld te hebben van hoe alles eigenlijk in zijn werk gaat, staan we toch even stil om iets dieper in te gaan op de werking hiervan.

Geen paniek, we gaan niet te diep in detail en beperken ons tot een minimum van technische termen en technologie. Het draait erom even duidelijkheid te scheppen wat er allemaal achter de schermen gebeurt op het moment dat je aan het surfen gaat.

We beginnen bij het begin, laat ons even de volgende situatie schetsen. Je opent een browser (naar keuze) en wil de facebook website openen.

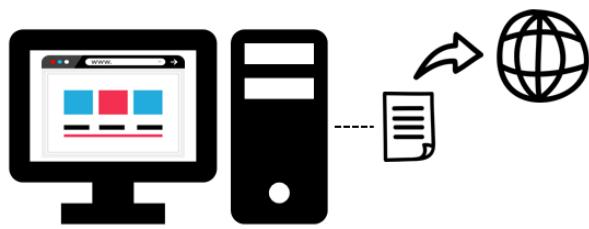
#### 1.1.1.1 STAP EEN



De invoer van de site waar je heen wil surfen. In dit geval wordt dit facebook.com

### 1.1.1.2 STAP TWEE

Jouw computer stuurt vervolgens een ‘bericht’ uit dat op zoek gaat naar deze informatie. De eerste stap waar jouw bericht zal passeren, is bij je provider. De bekendste providers bij ons zijn Telenet en Proximus. Deze ontvangt je bericht en kijkt of hij er mee aan de slag kan of niet.



### 1.1.1.3 STAP DRIE



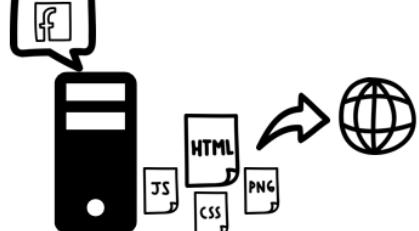
Je provider stuurt je bericht door naar de server van facebook en vraagt aan deze server of deze de gevraagde informatie beschikbaar heeft.



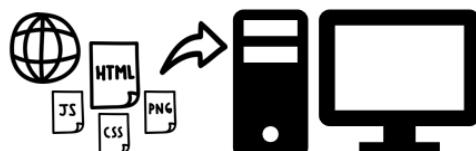
### 1.1.1.4 STAP VIER

De server van facebook gaat aan de slag om de gevraagde informatie op te halen. Dit kan je (weliswaar heel simpel) vergelijken met de mappen en folders op je eigen computer waarin je iets gaan zoeken.

### 1.1.1.5 STAP VIJF



De server van facebook stuurt de gevonden informatie terug naar je provider. Deze informatie bevat alles wat gerelateerd is aan je aanvraag.

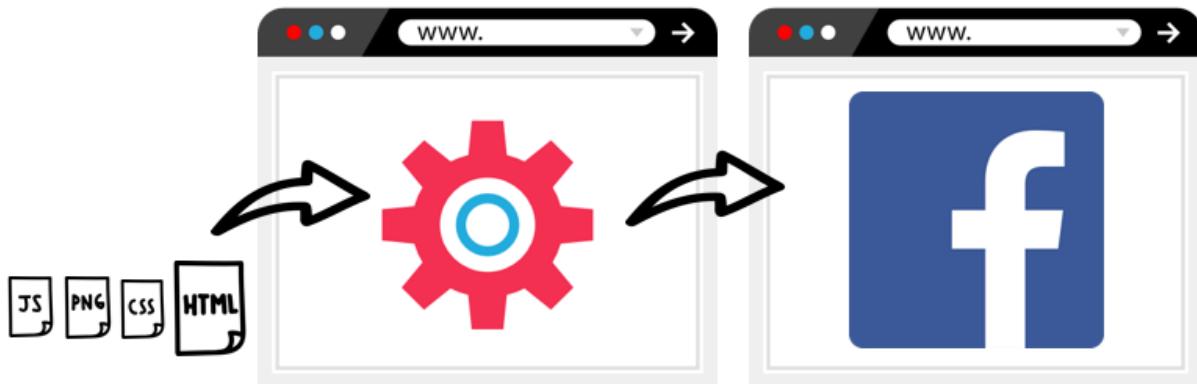


### 1.1.1.6 STAP ZES

Je had het zeker wel al zien aankomen. Je provider stuurt vervolgens dit pakketje door richting jouw computer.

### 1.1.1.7 STAP ZEVEN

Je browser gaat onmiddellijk aan de slag met deze informatie en zet de ontvangen informatie om naar voor jou leesbare en verstaanbare visuele informatie.



Ziezo, we hebben de volledige schakel even zeer simpel in kaart gebracht. Onthoud echter dat dit alles gepaard gaat met de nodige complexiteit en dat er nog wel iets meer bij kijken komt achter de schermen. Maar voor deze cursus volstaat het om inzicht te hebben in deze simpelere stappen.



Wil je hier meer over weten, lezen, opzoeken?

Check dan zeker even [HTML5 Rocks](#) voor wat meer informatie (*in detail*)

## 1.3 TERMINOLOGIE

### 1.3.1 HYPERTEXT MARKUP LANGUAGE - HTML

Dit is de taal en de structuur die we hanteren in de ontwikkeling van huidige webpagina's. Hoewel dit in een aantal (*verbeterde*) varianten voorkomt, is HTML nog steeds de basis voor de ontwikkeling van alle webpagina's.

Vandaag kan bijna elke computer een HTML pagina weergeven, maar elke computer zal deze pagina op een eigen manier weergeven. Het is de taak van de browsers (*zoals Chrome, Firefox, Opera, Safari, ...*) om de html correct te interpreteren en af te beelden, maar er zijn nog andere factoren: de snelheid van de computer, de internetverbinding, de instellingen van het beeldscherm (resoluties, etc) en zoveel meer waar de ontwikkelaar rekening mee dient te houden. De webdesigner staat altijd voor de uitdaging om goede en "**browser safe**" pagina's te kunnen maken, zodat die overal op dezelfde manier worden weergegeven, ongeacht de browser, computer, en andere factoren die de ervaring kunnen beïnvloeden.

< p > De letter p staat voor "paragraph" wat "alinea" betekent in het engels, niet paragraaf! Verwarrend niet? Een alinea is een verzameling zinnen terwijl een paragraaf meerdere alinea's kan bevatten. < /p >  
< p > Wanneer we echter paragraaf vertalen naar het engels dan krijgen we "section". < /p >

HTML Code voor de opbouw van twee alinea's.

HTML markeert tekst door middel van zogenaamde **tags**. Dit zijn de code objecten die tussen de bekjes < en > staan en ze vormen de basis voor de paginastructuur van een document, omdat ze de

weergave van de data bepalen. Een tag kan best omschreven worden als een etiket dat we rond een stuk tekst kleven om er een eigen betekenis aan te geven.

In zijn beginfase kon HTML ook de stijl van een pagina beïnvloeden: lettertypes, uitlijning, kleuren, enz. In HTML versie 4 heeft het W3C dergelijke stijlelementen afgekeurd (*deprecated*) en uit de officiële standaard gehaald en raadt ze het gebruik ervan af hoewel de meeste browsers ze nog steeds ondersteunen om alle sites te kunnen tonen. Deze tags worden hier niet meer behandeld.

Om de leegte van de afgekeurde stijlelementen te vullen, creëerde men een nieuw systeem om stijlen te kunnen instellen voor een HTML pagina. Deze instructieset noemt men Cascading Style Sheets of kortweg CSS en vervangt de meeste vroegere HTML opmaak tags, maar kan nog veel meer. Meer hierover verder in deze cursus.

### 1.3.2 VAN HTML NAAR HTML5

Sinds 2012 ondersteunen de meeste gebruikte browsers een nieuwe HTML specificatie, HTML5 genaamd. Deze versie introduceerde een hele reeks nieuwe elementen. Gelukkig is HTML5 geen letterlijke opvolger van HTML4, en kunnen we nog steeds **strikt volgens de HTML regels** coderen, en onze code zuiver houden.



Afgekeurde tags en attributen die niet langer deel uitmaken van HTML vind je hier:  
<http://htmlhelp.com/reference/html40/deprecated.html>



HTML5 is een **aanbevolen specificatie** bij W3C sinds oktober 2014. Het bevat heel wat elementen om een moderne webpagina's van inhoud te voorzien.

### 1.3.3 CASCADING STYLE SHEETS - CSS

Omdat sommige alinea's en andere tags meestal een bepaalde stijl , kleur en "look" moeten hebben op een webpagina worden deze voorzien van zogenaamde stijl-definities. Omdat HTML voornamelijk de opbouw van een pagina voorlegt - de blokken, paragrafen en andere zaken – was er nood aan een andere meer uniforme taal die deze blokken hun eigen stijl en karakter geeft. We moeten niet enkel de blokken bepalen door middel van **<p>-tags**, maar we kunnen deze ook een eigen stijl geven. Dit gebeurt dankzij de "stijl" taal genaamd CSS. We gaan gedurende deze cursus zowel HTML en CSS combineren om tot uniforme en presentabele webpagina's te komen. De benaming "cascading" komt nog duidelijk aan bod.

De volgende CSS-code zal bijvoorbeeld alle alinea's **<p>** steeds in het vet afdrukken.

```
p {  
    font-weight: bold;  
}
```

CSS Syntax voor de opmaak van alle alinea's.

Merk op dat de CSS taal sterk lijkt op een C of Java programmeertaal door het gebruik van accolades **{ }** en puntkomma's, maar toch gaat het hier slechts om instructies voor de te gebruiken opmaak. In dit geval heeft de eigenschap/property "font-weight" de instructie dat de tekst in elke "p" tag afgebeeld moet worden in het vet.



Cascading Stylesheets: [https://nl.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](https://nl.wikipedia.org/wiki/Cascading_Style_Sheets)



De meest recente implementatie van CSS is “CSS level 3”, dat een uitbreiding is bovenop CSS level 2.1.

## 1.4 ELEMENTEN, ATTRIBUTEN & WAARDEN

HTML heeft drie instructietypes waarmee we kunnen werken: elementen, attributen en waarden (*Elements, Attributes, Values*).

### 1.4.1 HTML ELEMENTEN

Elementen structureren de verschillende delen van een webpagina. Ze markeren bepaalde delen in de pagina zodat de inhoud een bepaalde functie krijgt. In het vorige hoofdstuk hebben we reeds kennismegemaakt met `<p>` elementen die aanduiden dat elk stuk tekst die ze omsluiten een alinea betreft. De elementen kunnen **tekst of andere elementen** bevatten, maar ze kunnen ook **leeg** zijn.

```
<h1>Deze tekst wordt beïnvloed en is nu een hoofding</h1>
```

Een typisch HTML element dat tekst bevat

Wanneer een element in HTML iets bevat, zoals tekst of andere elementen, dan moet het element **steeds afgesloten** worden met zijn sluitingstag. Het bovenstaande voorbeeld zorgt ervoor dat de ingesloten tekst als **hoofding (h1)** zal weergegeven worden. De openingstag is een `<h1>` en wordt ook opnieuw afgesloten met de sluitingstag `</h1>`.

Er bestaan ook elementen die **geen** inhoud kunnen bevatten. Deze elementen hebben dan ook **geen sluitingstag**, maar worden op een **alternatieve manier afgesloten**. Dergelijke elementen bestaan uit een zogenaamde **zelfsluitende (self-closing) tag** die zowel de openingstag als sluitingstag is. Een mooi voorbeeld is het **img** element, dat een afbeelding op de pagina plaatst.

```

```

Voorbeeld van een zelfsluitend element

Zoals je ziet is er niet echt een openingstag of een sluitingstag. **De tag sluit zichzelf** door middel van **een spatie, een slash en een bekje**.

Tot slot moet er nog vermeld worden dat in HTML **alle tags** in kleine letters geschreven moeten worden. Dit slaat uiteraard niet op de tekstuele inhoud ervan.



Een overzicht van alle mogelijke HTML elementen kan je terugvinden via onderstaande site:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>

## 1.4.2 ATTRIBUTEN EN WAARDEN

Attributen maken het mogelijk om **informatie over de data** van het element te beschrijven, in tegenstelling tot de **data zelf** die we terugvinden tussen de openings-en sluitingstags. Wanneer een attribuut geplaatst wordt in een tag dan heeft deze **altijd** een waarde in HTML. Deze waarde wordt steeds ingesloten tussen dubbele aanhalingstekens, zoals het voorbeeld hieronder toont.

```
<a href="http://www.google.be">Ga naar Google</a>
```

Een attribuut van het a-element. De aanhalingstekens zijn verplicht en omsluiten de **waarde** van het **attribuut href**.

Een a-element creëert een hyperlink op de pagina. In dit geval zal de hyperlink de tekst “Ga naar Google” als data gebruiken en weergeven. We hebben echter ook extra informatie nodig bij deze data, namelijk het adres van de website waar we naartoe willen navigeren bij het klikken op de link. Dit gebeurt door het attribuut **href**. Let goed op de plaatsing van de aanhalingstekens en het gelijkheidsteken.

Hetzelfde geldt voor attributen van een zelfsluitende tag:

```

```

Bij deze img-tag zijn er twee attributen gedefinieerd: waar de figuur gevonden kan worden (**src**) en de alternatieve tekst. Vergeet de spatie voor de eind-slash niet.

Er bestaan twee soorten attributen: **specifieke** en **globale** attributen. Globale attributen (ook wel standaardattributen genoemd) zijn van toepassing op **alle** HTML-elementen. Specifieke attributen zijn steeds afhankelijk van het element.



Attributen kunnen niet zo maar willekeurige waarden bevatten. Voor een goede referentie neem je dus best even een kijkje online.



Een overzicht van alle global attributes kan je terugvinden via onderstaande site:  
[https://developer.mozilla.org/en-US/docs/Web/HTML/Global\\_attributes](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes)

## 1.5 NESTEN VAN ELEMENTEN

Alle elementen die niet zelfsluitend zijn, m.a.w. elementen die een openingstag en sluitingstag hebben, kunnen andere elementen bevatten. Elementen die zelfsluitend zijn kunnen uiteraard geen tekst of andere elementen meer bevatten. De elementen die ingesloten zijn door een ander element noemen we **Child** elementen, en het omsluitende element heet het **Parent** element. We kunnen dus gerust spreken van een hierarchie in de structuur die sterk lijkt op een stamboom, de “document tree”.

In het onderstaande voorbeeld is het div element de **parent** van de img en de twee p elementen. De p elementen zijn **children** van de div. De eerste p is **parent** van het cite element, wat op zijn beurt een **child** element is van dit p element.

```
<div>
  <p>
    De haas en de schildpad is een fabel door <cite>Aesopus</cite>
  </p>
  <p>Hij was een grieke verteller uit de oudheid.</p>
  
</div>
```

Deze relaties tussen elementen zijn belangrijk bij het stylen van elementen. We gaan het hier nog uitgebreid over hebben wanneer we CSS onder de loep nemen.

Wanneer we nu een element **in** een element willen plaatsen dan is het belangrijk om deze hiërarchie te respecteren.

```
<p>De haas en de schildpad is een fabel door <cite>Aesopus
<mark>(Griek)</mark></cite></p>
```

*Geneste elementen. Juist afgesloten.*

Het p-element bevat nu wat tekst, aangevuld met een verwijzing naar de auteur via het **cite**-element en wat verderop ook een **mark**-element. We zeggen dat het mark-element **genest** is in het cite-element.

Aangezien we nog steeds **in** het cite-element zaten toen we de openingstag **<mark>** schreven, moet ook de eindtag **</mark>** zich in dit cite-element bevinden. De volgende code zou dus volledig **FOUT** zijn:

```
<p>De haas en de schildpad is een fabel door <cite>Aesopus</p></cite>
```

*Nesten van elementen. **FOUT** afgesloten.*

Zodra je een sluitingstag plaatst moet deze dus overeenstemmen met de dichtst voorgaande, niet gesloten openingstag. Als we dus eerst een element X openen en daarna een element Y, dan moeten we eerst Y sluiten en daarna pas X.

## 1.6 EDITOREN

Er zijn tal van Editoren op de markt waar je mee aan de slag kan om met HTML en CSS je webpagina's tot leven te brengen.

Je bent vrij om hier je eigen keuze in te maken.

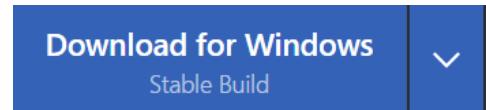
De voorbeelden en referenties in deze cursus zullen echter steeds verwijzen naar Visual Studio Code. We raden het gebruik van Visual Studio Code dan ook aan als je vlot wil meevolgen en niet te veel tijd wil verliezen om een andere editor onder de knie te krijgen.

### 1.6.1 VS CODE

In onderstaande stappen overlopen we kort de installatie van Visual Studio Code. Handige Extensions worden ook vermeld na de stappen van de installatie

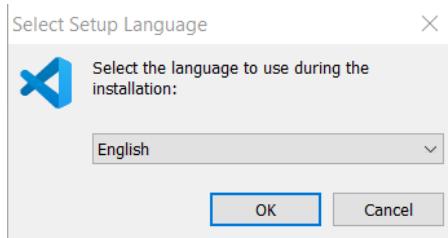
#### 1.1.1.8 INSTALLATIE

1. Download de laatste versie van Visual Studio Code via de **Download for Windows** knop op de website. Na enkele ogenblikken start je download automatisch.  
Indien je een ander platform draait, kies dan uiteraard voor de relevante versie.



Je kan de laatste versie van Visual Studio Code terugvinden via onderstaande link:  
<https://code.visualstudio.com>

2. Kies voor **English** bij de installatie taal.

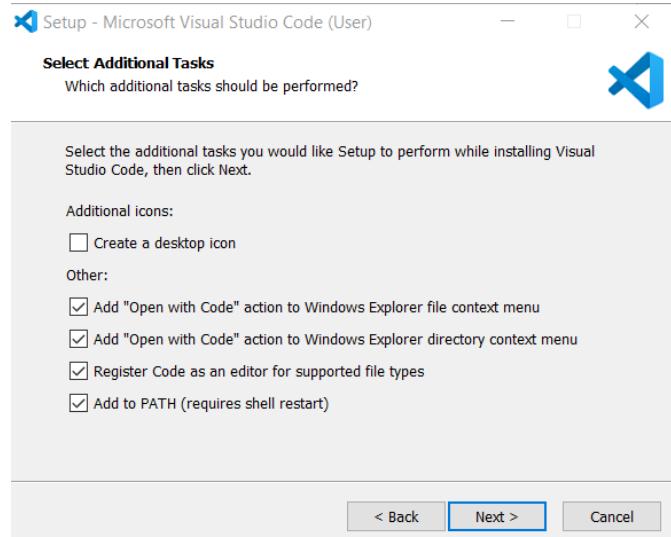


**3. Accepteer vervolgens de License Agreement en klik op Next.**

**4. Zorg er vervolgens voor dat de opties**  
zoals deze op de screenshot zichtbaar  
zijn **aangevinkt** staan. Deze zullen het  
je net iets makkelijker maken om met  
Visual Studio Code te werken.

Klik vervolgens op **Next** en **Install**.

Visual Studio Code wordt nu  
geïnstalleerd.



### 1.1.1.9 NUTTIGE EXTENSIES

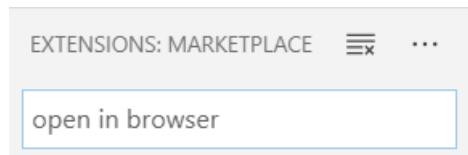
Er zijn tal van Extensies die beschikbaar zijn binnen Visual Studio Code om het leven van een developer makkelijker te maken. We kunnen ze uiteraard niet allemaal bespreken maar we pikken er wel even de twee meest interessante uit om meteen mee van start te gaan.

#### 1. Open in browser – TechER

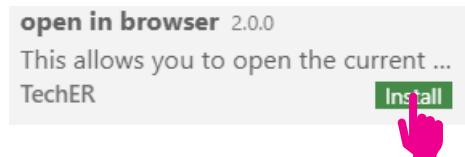
Deze extensie stelt je in staat om vanuit Visual Studio Code meteen je pagina door je default browser te laten openen door de toetsencombinatie **alt-b** te gebruiken. Je kan deze extensie rechtstreeks in je Visual Studio Code toevoegen door in de linker

navigatiebalk op het icoon voor extensions te klikken 

Voeg vervolgens bovenaan in de zoekbalk **open in browser** als zoekterm in.



In de gefilterde lijst klik je vervolgens op de **Install** button die bij de extension **open in browser** van de publisher **TechER**



Meer is er niet nodig om deze extensie te installeren.



Je kan deze extensie ook terugvinden in de Visual Studio Marketplace via onderstaande link:

[Marketplace - open in browser - TechER](#)

#### 2. Live Server – Ritwick Dey

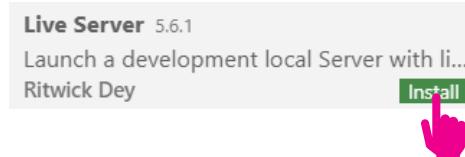
Deze extensie stelt je in staat om wijzigingen die je in Visual Studio Code maakt meteen door te voeren op de pagina in je browser. Je kan deze extensie rechtstreeks in je Visual Studio Code toevoegen door in de linker navigatiebalk op het icoon voor

extensions te klikken 

Voer vervolgens bovenaan in de zoekbalk **Live Server** als zoekterm in.



In de gefilterde lijst klik je vervolgens op de **Install** button die bij de extension **Live Server** van de



publisher **Ritwick Dey** staat.

Meer is er niet nodig om deze extensie te installeren.



Je kan deze extensie ook terugvinden in de Visual Studio Marketplace via onderstaande link:

[Marketplace - Live Server - Ritwick Dey](#)

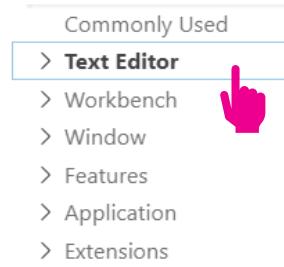
### 1.1.1.10 AANBEVOLEN INSTELLINGEN

Tot slot hebben we nog een instelling die best handig is om in te schakelen in Visual Studio Code. Dat is de optie om je wijzigingen automatisch op te slaan. Op deze manier worden je laatste wijzigingen steeds meegenomen en verlies je alvast geen tijd met het zoeken waarom die laatste aanpassing niet correct weergegeven wordt in de browser.

In je Visual Studio Code ga je naar:

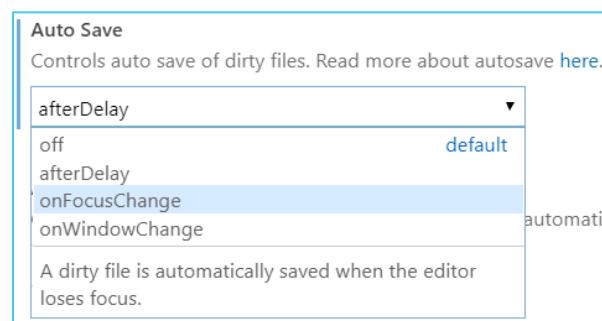
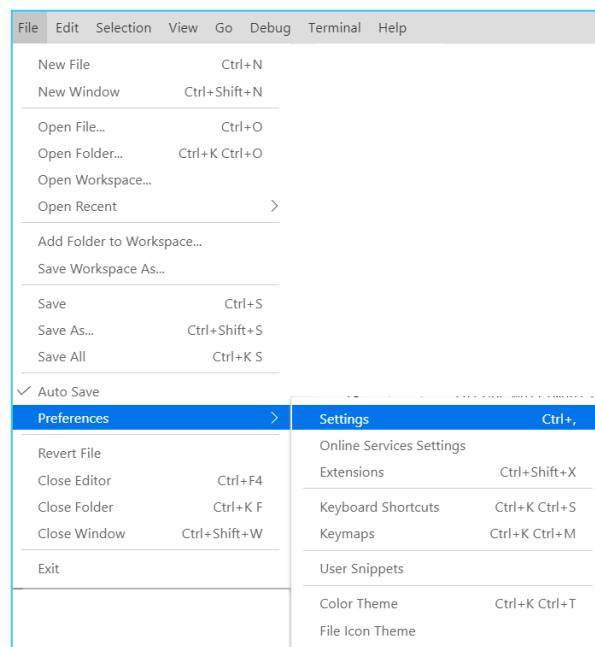
**File -> Preferences -> Settings**

Klik vervolgens de **Tekst Editor** node open



Na het openklappen selecteer je hier de menu optie **Files**

Daarin ga je opzoek naar de **Auto Save** setting en plaats deze op **onFocusChange**.



## 1.7 HTML VALIDATIE

---

Tijdens deze module werken we er naartoe om steeds geldige HTML code op te leveren. Om ervoor te zorgen dat je HTML code vrij van fouten is kan je hiervoor gebruik maken van een HTML Validator.



De officiële validatie is terug te vinden online via het W3C op volgende website:  
[Markup Validation Service](#)

Via deze website kan je HTML laten valideren op basis van een URL, een bestand dat je upload, of het copy pasten van je broncode. Deze verschillende opties zijn terug te vinden in de verschillende tabbladen op de website zelf.

Je bent vrij om gebruik te maken van andere validators die je online tegenkomt, of eventueel als plugin kan installeren in je browser zelf. Maar zorg ervoor dat je zelf dan ook even controleert als de door jou gekozen tool ook dezelfde errors toont als deze van de W3C. Daar deze manier van controleren voor opdrachten/examen gehanteerd zal worden door de lectoren.





# Van start met HTML en CSS

**Web Frontend Basics**

## INHOUD

|            |  |                              |
|------------|--|------------------------------|
| <b>1</b>   | <b>VAN START MET HTML EN CSS</b>         | <b>3</b>                     |
| <b>1.1</b> | <b>Je eerste webpagina maken</b>         | <b>3</b>                     |
| 1.1.1      | VS Code                                  | 3                            |
|            | Stap 1                                   | 3                            |
|            | Stap 2                                   | 3                            |
|            | Stap 3                                   | 3                            |
|            | Stap 4                                   | 3                            |
|            | Stap 5                                   | 4                            |
|            | Stap 6                                   | Error! Bookmark not defined. |
| 1.1.2      | HTML Code onder de loep                  | 6                            |
| 1.1.1.1    | Structuur van onze pagina                | 6                            |
| 1.1.3      | Commentaar in onze code                  | 8                            |
| 1.1.4      | Onze eerste elementen                    | 9                            |
| 1.1.1.2    | Hoofdingen h1 ... h6                     | 9                            |
| 1.1.1.3    | Alinea – P                               | 9                            |
| 1.1.1.4    | Toepassing                               | 10                           |
| 1.1.1.5    | Basiselementen span en div               | 12                           |
| <b>1.2</b> | <b>Het CSS Box Model</b>                 | <b>13</b>                    |
| 1.2.1      | Wat is het CSS Box Model?                | 13                           |
| 1.1.1.6    | Vakken in het CSS Box Model              | 14                           |
| 1.2.2      | Inline en Block elementen                | 14                           |
| <b>1.3</b> | <b>Onze eerste Cascading Style Sheet</b> | <b>15</b>                    |
| Stap 1     |  | 15                           |
| Stap 2     |  | 15                           |
| Stap 3     |  | 15                           |
| Stap 4     |  | 16                           |
| Stap 5     |  | 16                           |
| 1.3.1      | CSS Selector op Element                  | 17                           |

## 2 VAN START MET HTML EN CSS

### 2.1 JE EERSTE WEBPAGINA MAKEN

#### 2.1.1 VS CODE

##### STAP 1

Voor we VS Code openen, maken we eerst een **nieuwe folder** aan.

Doe dit bij voorkeur op een locatie op je harde schijf waar je ook je andere HTML oefeningen, uitwerkingen, ... zal plaatsen. Zo blijft alles alvast mooi gegroepeerd en moet je nooit lang op zoek naar waar je nu weer je bestanden geplaatst hebt.

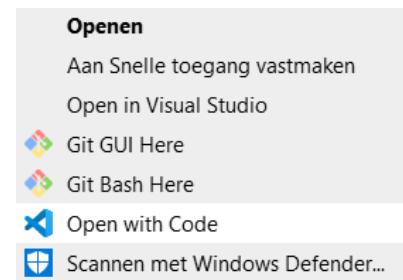
##### STAP 2

We openen deze folder in VS Code. Dit kan op twee manieren.

###### Manier 1

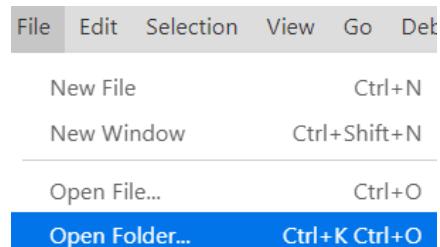
Rechtsklik op de folder die je net aanmaakte en klik op **Open with Code**. Zoals hiernaast te zien is.

Indien je deze optie niet hebt dan ben je bij de installatie van VS Code deze setting vergeten aanvinken. Niet erg, je kan dan ook gewoon met de tweede manier aan de slag.



###### Manier 2

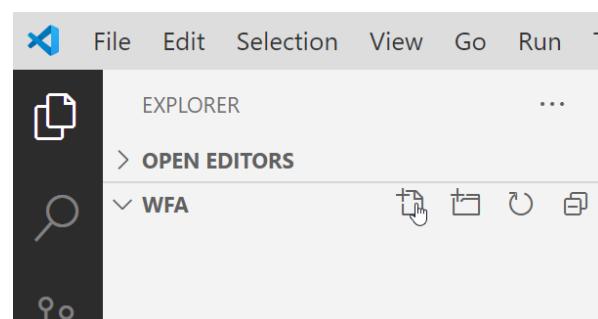
Open VS Code en ga hier dan naar **File -> Open Folder ...** Kies hierbij dan de correcte folder waar je mee aan de slag wil gaan.



##### STAP 3

Je ziet nu in de VS Code Explorer de naam van je gekozen folder staan (*in de screenshot heeft onze folder de naam voorbeeld*). We gaan meteen een **nieuw bestand**, een HTML pagina, **toevoegen** aan deze folder.

Dit doe je door op het icoontje **New File** te klikken naast de naam van je folder. Geef je nieuw bestand de naam **index.html**



##### STAP 4

Laten we wat HTML syntax toevoegen, selecteer het zopas aangemaakte bestand index.html.

Klik vervolgens op de eerste (*lege*) regel in het bestand om de nodige syntax in te voeren. In VS Code kan dit simpelweg door **html:5** in te typen en vervolgens de Tab toets in te drukken, of nog korder

door ! te typen en de tab toets in te drukken. De editor voorziet meteen wat nodig is aan code om een geldig HTML5 document te maken.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

De startcode die voorzien wordt in VS Code

## STAP 5

We passen onze pagina nog een klein beetje aan zodat we ook wat inhoud zien in onze browser als we deze zometeen gaan testen.

Tussen de **title** tags vervangen we **Document** door **Mijn Eerste Pagina**.

```
<title>Mijn Eerste Pagina</title>
```

Updaten van de inhoud in het title element

In het body element voegen we vervolgens wat inhoud toe.

```
<body>
<h1>Introductie HTML</h1>
<p>Introductie voor Graduaat Programmeren</p>

<h2>Hoe werkt een browser</h2>
<p>Basis werking van een browser</p>
</body>
```

Inhoud van het body element aangevuld met een **h1**, **h2** en twee **p** elementen

## 2.1.2 ENKELE HANDIGE TIPS

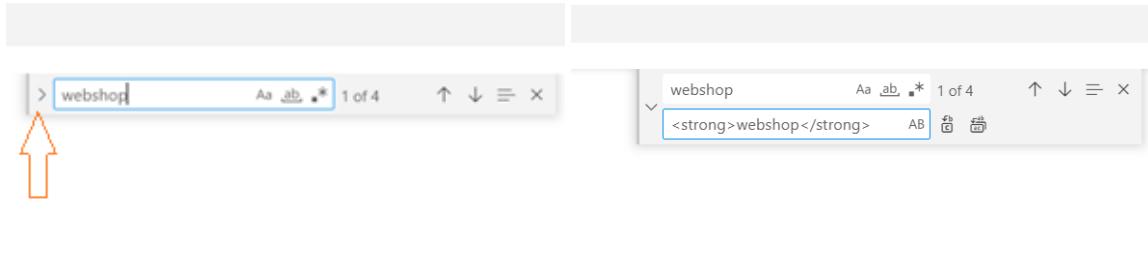
### 2.1.2.1 SHORTCUTS

Binnen VS Code heb je een aantal handige shortcuts die het leven van een ontwikkelaar aangenamer maken.

- Alt + ↑: Beweeg lijn naar omhoog
- Alt + ↓: Beweeg lijn naar beneden
- Alt + Shift + ↑: Kopieer lijn naar omhoog
- Alt + Shift + ↓: Kopieer lijn naar beneden
- Alt + Shift + F: Document opmaken

### 2.1.2.2 ZOEKEN EN AANPASSEN IN EEN BESTAND

Om gemakkelijk aanpassingen door te voeren in de tekst kan je gebruik maken van de zoekfunctie in een document. Als je op **ctrl + F** drukt opent zich een zoekvenster. Als je op de chevron links van de tekst klikt, kan je de gevonden tekst ook laten vervangen.



## 2.1.3 HTML CODE ONDER DE LOEP

Wat schreven we nu net allemaal en hoe werd dit vervolgens in onze browser verwerkt voor de weergave?

### 2.1.3.1 STRUCTUUR VAN ONZE PAGINA

Bovenaan vinden we als eerste regel de declaratie van het type document terug. Op basis hiervan krijgt je browser meteen te weten wat het type van het document is dat hij aan het openen is. Al je webpagina's zullen dus starten met deze declaratie. Ter verduidelijking deze regel is geen HTML tag.

```
<!DOCTYPE html>
```

*Declaratie van het type document*

Vervolgens komen we het **html** element tegen. Het **html** element staat voor de 'root' (het element op het hoogste niveau) van een HTML-document. Daarom dat je in sommige referenties **root-element** kan terugvinden als ze het over dit element hebben. Alle andere elementen die je dus zal gebruiken, moeten binnen dit element hun plaats krijgen.

Het **html** element heeft, net zoals alle andere elementen, de mogelijkheid om aangevuld te worden met **attributen**. Op onze eerste pagina hebben we reeds een language attribuut staan (*lang*) waar we verduidelijking kunnen geven aan de taal die hoofdzakelijk in het document gebruikt wordt. We kunnen dit dus gerust aanpassen naar **nl**, wat voor Nederlands staat.

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

*Het html element, met aangepast lang attribuut, ook gerefereerd als root-element.*

Vervolgens kunnen we twee grote blokken onderscheiden binnen ons html element zelf. Eerst hebben we het **head** element met net daaronder het **body** element.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Mijn Eerste Pagina</title>
</head>
```

*Het head element*

Het head element bevat 4 andere elementen waarvan er drie **meta** elementen zijn. Meta elementen komen later nog uitgebreider aan bod, maar we staan alvast toch even stil bij het eerste **meta** element.

Het **meta** element met attribuut **charset** stelt ons in staat om de browser duidelijk te maken welke

characterset er in het onderstaande document gebruikt wordt. Hier ga je standaard steeds UTF-8 invullen/tegenkomen.

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

Meta element met attribuut charset ingesteld op waarde UTF-8

Het **title** element stelt je in staat om je pagina een titel te geven. De inhoud van een paginatitel kan aanzienlijke gevolgen hebben voor zoekmachineoptimalisatie (*SEO*). Over het algemeen presteert een langere, beschrijvende titel beter dan korte of generieke titels.

De inhoud van de titel is één van de componenten die door algoritmen van zoekmachines worden gebruikt om de volgorde te bepalen waarin pagina's in zoekresultaten worden weergegeven. De titel is ook de eerste "hook" waarmee u de aandacht trekt van lezers die naar de pagina met zoekresultaten kijken.

```
<title>Mijn Eerste Pagina</title>
```

Het title element op onze eerste pagina



#### Enkele richtlijnen en tips voor het samenstellen van goede titels:

- Vermijd titels met één of twee woorden. Gebruik een beschrijvende zin of een term-definitie paar voor woordenlijst- of referentiestijlpagina's.
- Zoekmachines geven meestal ongeveer de eerste 55-60 tekens van een paginatitel weer. Tekst die verder gaat, kan verloren gaan, dus probeer je titels niet langer te maken. Als je dan toch een langere titel moet gebruiken, zorg er dan voor dat de belangrijke delen zich in de eerste 55-60 tekens bevinden.
- Vermijd speciale tekens wanneer mogelijk; niet alle browsers zullen ze op dezelfde manier weergeven. "<" Wordt bijvoorbeeld vaak weergegeven in de titelbalk van het venster als "&lt;" (de HTML kleiner dan entiteit).
- Gebruik geen 'trefwoord-blobs'. Als de titel slechts bestaat uit een woordenlijst, verminderen algoritmen vaak de positie van uw pagina in de zoekresultaten.
- Probeer ervoor te zorgen dat uw titels zo uniek mogelijk zijn binnen uw eigen site. Identieke of bijna identieke titels overeen verschillende paginas kunnen bijdragen aan onnauwkeurige zoekresultaten.

Bron: [Mozilla Developer Network - Title element](#)

Rest er nog het **body** element samen met de elementen die daar voor de nodige inhoud van onze pagina zorgen.

Volgens de HTML specificaties dient het **body** element steeds het tweede child element van een html element te zijn. Wat we aan elementen voorzien in het **body** element zal de inhoud van onze pagina gaan bepalen. De elementen die we gebruiken op onze eerste pagina in het **body** element zijn **h1**, **h2** en **p** elementen.

Alle drie deze elementen zijn van nature **block elementen** en nemen de volledig voor hen beschikbare ruimte in beslag. Geen paniek, we komen hier later nog op terug.

```
<h1>Introductie HTML</h1>
<p>Introductie voor Graduaat Programmeren</p>
```

```
<h2>Hoe werkt een browser</h2>
<p>Basis werking van een browser</p>
```

Inhoud van het body element aangevuld met een **h1**, **h2** en twee **p** elementen

#### 2.1.4 COMMENTAAR IN ONZE CODE

Niet alles wat we typen willen we door onze browser zichtbaar laten maken naar de bezoekers van onze website. Soms kan het handig zijn voor de developer zelf om op bepaalde plaatsen in de HTML code één of meerdere regels commentaar te voorzien. Commentaar wordt niet door de browser gerenderd, we kunnen commentaar plaatsen door middel van de tags **<!-- -->**

```
<!-- -->
```

Commentaar in onze code dient omringd worden door bovenstaande syntax

We kunnen gerust een of meerdere regels commentaar toevoegen op onze eerste pagina.

```
<body>
  <!-- Een regel commentaar toevoegen in onze HTML -->
  <h1>Introductie HTML</h1>
  <p>Introductie voor Graduaat Programmeren</p>

  <!--
    Commentaar spreiden overheen
    verschillende lijnen.
  -->
  <h2>Hoe werkt een browser</h2>
  <p>Basis werking van een browser</p>
</body>
```

Onze eerste pagina, aangevuld met commentaar



#### Opgelet

Commentaar wordt niet gerenderd maar kan wel geraadpleegd worden door de bezoeker van je site als deze de broncode bekijkt. Denk er dan ook altijd aan om **geen gevoelige informatie** in commentaar te plaatsen tijdens de uitwerking van je pagina(s).

## 2.1.5 ONZE EERSTE ELEMENTEN

### 2.1.5.1 HOOFDINGEN H1 ... H6

Laat ons met het echte werk beginnen en een pagina opmaken. Hiervoor hebben we een veelvoud van elementen tot onze beschikking die een pagina kunnen opdelen in hoofdingen, alinea's en andere secties. Om een hoofding te maken, kunnen we gebruik maken van een **h1** tag, die de tekst automatisch veel groter zal afbeelden. Je kan dit vergelijken met standaard hoofdingen die we terugvinden in tekstverwerkers. De elementen die we voor hoofdingen kunnen gebruiken lopen van **h1** tot en met **h6**.

```
<h1>Hoofding niveau 1</h1>
<h2>Hoofding niveau 2</h2>
<h3>Hoofding niveau 3</h3>
<h4>Hoofding niveau 4</h4>
<h5>Hoofding niveau 5</h5>
<h6>Hoofding niveau 6</h6>
```

De verschillende niveaus van hoofdingen

# Hoofding niveau 1

## Hoofding niveau 2

### Hoofding niveau 3

#### Hoofding niveau 4

##### Hoofding niveau 5

###### Hoofding niveau 6

Voorbeeld van de output in een browser



#### Enkele richtlijnen voor het gebruik van hoofdingen:

- Vermijd meer dan één `<h1>` hoofding op een pagina.
- Hoofdingen kunnen bijvoorbeeld door user-agents (*vb. een browser*) worden gebruikt om automatisch een inhoudsopgave voor een document te maken.
- Gebruik nooit een hoofding omdat je het formaat van de tekst wenst te wijzigen. Gebruik in plaats daarvan relevante CSS. Hoofdingen gebruiken grootte om hun relatieve belang aan te geven, CSS heeft steeds de voorkeur indien het over het louter aanpassen van formaat gaat.
- Vermijd het overslaan van niveaus: begin steeds vanaf `<h1>`, gebruik vervolgens `<h2>` enzovoort.

Bron: [Mozilla Developer Network - <h1>-<h6>: The HTML Section Heading elements](#)

### 2.1.5.2 ALINEA – P

Het **p** element vertegenwoordigt een alinea. Alinea's worden meestal in visuele media weergegeven als tekstblokken gescheiden van aangrenzende blokken door lege regels en / of inspringen op de eerste regel. In HTML kunnen alinea's eender welke groepering van inhoud zijn, zoals afbeeldingen of formuliervelden.

Dit element geeft automatisch een stuk witruimte zodat de alinea's duidelijk gescheiden blijven van elkaar.

### 2.1.5.3 TOEPASSING

Laten we gebruik maken van beide elementen om een stuk tekst wat meer vorm te geven. Maak hiervoor een nieuwe pagina aan in VS Code en neem onderstaande tekst over in het body element van je HTML pagina.

Leonardo da Vinci

Leonardo da Vinci (Italië), 15 april 1452 - Cloux (thans Clos-Lucé) (Frankrijk), (2 mei 1519) was een beroemde Italiaanse architect, uitvinder, ingenieur, filosoof, natuurkundige, scheikundige, beeldhouwer, schrijver, schilder en componist uit de Renaissance.

Kunst

Leonardo is beroemd vanwege schilderwerken als de Mona Lisa (ook "La Gioconda"), dat zich nu in het Parijse Louvre-museum bevindt en de muurschildering Het Laatste Avondmaal.

Slechts zeventien van zijn doeken en geen van zijn beelden zijn bewaard gebleven. Het merendeel van de tekeningen wordt bewaard in de Koninklijke Collectie van het Windsor Castle in Engeland. Zij zijn eigendom van Koningin Elisabeth.

Wetenschap en techniek

Leonardo's wetenschappelijke werk was gebaseerd op empirisch onderzoek, niet op wetenschappelijke experimenten of theoretische verklaringen: hij probeerde een verschijnsel te doorgronden door het zo gedetailleerd mogelijk te beschrijven en te tekenen. Gedurende zijn hele leven had hij het plan om een grote encyclopedie te maken op basis van tekeningen.

Zijn boeken bevatten tekeningen van verschillende nieuwe apparaten en machines, zoals een helikopter, machinegeweren, een bepantserde tank, een onderzeeboot en een apparaat met tandwielen waarvan men vermoedt dat het een mechanische rekenmachine voorstelt.

In 1502 maakte Leonardo een ontwerp voor een 240 m lange brug zonder tussenliggende pijlers. Het ontwerp was bedoeld om een inham van de Bosporus die bekend stond als De Gouden Hoorn te overbruggen. De brug werd daar nooit gebouwd, maar in 2001 werd er in Noorwegen een brug geconstrueerd volgens hetzelfde principe.

Brontekst, Leonardo da Vinci

Bekijk je webpagina al even door de **alt+b** toetsencombinatie in VS Code, of via de rechtermuisknop in de Explorer met de keuze '**Open With Live Server**'.

Werken deze niet in jouw VS Code? Herneem dan even het hoofdstuk omtrent Editoren en de installatie van VS Code met aanbevolen extensies.

Je zal zien dat, ondanks een kleine basis opmaak in de tekst, de browser hier totaal geen rekening mee zal houden. De browser baseert zich immers op wat er aan HTML elementen werd voorzien en kijkt niet naar de reguliere enters, meervoudige spaties, etc. tijdens het verwerken van een document.

Leonardo da Vinci Leonardo da Vinci (Italië), 15 april 1452 - Cloux (thans Clos-Lucé) (Frankrijk), (2 mei 1519) was een beroemde Italiaanse architect, uitvinder, ingenieur, filosoof, natuurkundige, scheikundige, beeldhouwer, schrijver, schilder en componist uit de Renaissance. Kunst Leonardo is beroemd vanwege schilderwerken als de Mona Lisa (ook "La Gioconda"), dat zich nu in het Parijse Louvre-museum bevindt en de muurschildering Het Laatste Avondmaal. Slechts zeventien van zijn doeken en geen van zijn beelden zijn bewaard gebleven. Het merendeel van de tekeningen wordt bewaard in de Koninklijke Collectie van het Windsor Castle in Engeland. Zij zijn eigendom van Koningin Elisabeth. Wetenschap en techniek Leonardo's wetenschappelijke werk was gebaseerd op empirisch onderzoek, niet op wetenschappelijke experimenten of theoretische verklaringen: hij probeerde een verschijnsel te doorgronden door het zo gedetailleerd mogelijk te beschrijven en te tekenen. Gedurende zijn hele leven had hij het plan om een grote encyclopedie te maken op basis van tekeningen. Zijn boeken bevatten tekeningen van verschillende nieuwe apparaten en machines, zoals een helikopter, machinegeweren, een bepantserde tank, een onderzeeboot en een apparaat met tandwielen waarvan men vermoedt dat het een mechanische rekenmachine voorstelt. In 1502 maakte Leonardo een ontwerp voor een 240 m lange brug zonder tussenliggende pijlers. Het ontwerp was bedoeld om een inham van de Bosporus die bekend stond als De Gouden Hoorn te overbruggen. De brug werd daar nooit gebouwd, maar in 2001 werd er in Noorwegen een brug geconstrueerd volgens hetzelfde principe.

Voorbeeld weergave van de tekst in een browser

We voegen even een aantal elementen rondom de tekst heen om onze pagina van wat meer structuur te voorzien.

We starten met de hoofdingen toe te voegen rondom de passende regels in de tekst. De hoofdtitel voorzien we van een **h1** element, alle volgende tussentitels voorzien we van een **h2** element.

```
<h1>Leonardo da Vinci</h1>
...
<h2>Kunst</h2>
...
<h2>Wetenschap en techniek</h2>
...
```

Toevoegen van h1 en h2 elementen

Vervolgens transformeren we even de blokjes tekst om in alinea's. Dit doen we door deze te omringen door het **p** element. We krijgen dan vervolgens onderstaand resultaat in onze HTML code.

```
<h1>Leonardo da Vinci</h1>
<p>Leonardo da Vinci ... uit de Renaissance.</p>

<h2>Kunst</h2>
<p>Leonardo is beroemd ... Het Laatste Avondmaal.</p>
<p>Slechts zeventien ... van Koningin Elisabeth.</p>

<h2>Wetenschap en techniek</h2>
<p>Leonardo's wetenschappelijke ... op basis van tekeningen.</p>
<p>Zijn boeken ... mechanische rekenmachine voorstelt.</p>
<p>In 1502 maakte ... volgens hetzelfdeprincipe.</p>
```

Toevoeging van p elementen om de nodige alinea's te maken.

Het resultaat ziet er dan ook meteen iets anders uit in onze browser.

## Leonardo da Vinci

Leonardo da Vinci (Italië), 15 april 1452 - Cloux (thans Clos-Lucé) (Frankrijk), 2 mei 1519) was een beroemde Italiaanse architect, uitvinder, ingenieur, filosoof, natuurkundige, scheikundige, beeldhouwer, schrijver, schilder en componist uit de Renaissance.

### Kunst

Leonardo is beroemd vanwege schilderwerken als de Mona Lisa (ook "La Gioconda"), dat zich nu in het Parijse Louvre-museum bevindt en de muurschildering Het Laatste Avondmaal.

Slechts zeventien van zijn doeken en geen van zijn beelden zijn bewaard gebleven. Het merendeel van de tekeningen wordt bewaard in de Koninklijke Collectie van het Windsor Castle in Engeland. Zij zijn eigendom van Koningin Elisabeth.

### Wetenschap en techniek

Leonardo's wetenschappelijke werk was gebaseerd op empirisch onderzoek, niet op wetenschappelijke experimenten of theoretische verklaringen: hij probeerde een verschijnsel te doorgronden door het zo gedetailleerd mogelijk te beschrijven en te tekenen. Gedurende zijn hele leven had hij het plan om een groot encyclopedie te maken op basis van tekeningen.

Zijn boeken bevatten tekeningen van verschillende nieuwe apparaten en machines, zoals een helikopter, machinegeweren, een bepantserde tank, een onderzeeboot en een apparaat met tandwielen waarvan men vermoedt dat het een mechanische rekenmachine voorstelt.

In 1502 maakte Leonardo een ontwerp voor een 240 m lange brug zonder tussenliggende pijlers. Het ontwerp was bedoeld om een inham van de Bosporus die bekend stond als De Gouden Hoorn te overbruggen. De brug werd daar nooit gebouwd, maar in 2001 werd er in Noorwegen een brug geconstrueerd volgens hetzelfde principe.

Voorbeeld weergave van de aanpassingen in een browser.

#### 2.1.5.4 BASISELEMENTEN SPAN EN DIV

Er bestaan twee elementen die geen voorgedefinieerde opmaak of witruimte hebben, namelijk het **div** en **span** element. We passen beiden ook even onmiddellijk toe om dit te illustreren. Voeg rondom elke **blok** tekst een div element toe.

```
<div>
    <h1>Leonardo da Vinci</h1>
    <p>Leonardo da Vinci ... uit de Renaissance.</p>
</div>
<div>
    <h2>Kunst</h2>
    <p>Leonardo is beroemd ... Het Laatste Avondmaal.</p>
    <p>Slechts zeventien ... van Koningin Elisabeth.</p>
</div>
<div>
    <h2>Wetenschap en techniek</h2>
    <p>Leonardo's wetenschappelijke ... op basis van tekeningen.</p>
    <p>Zijn boeken ... mechanische rekenmachine voorstelt.</p>
    <p>In 1502 maakte ... volgens hetzelfdeprincipe.</p>
</div>
```

Toevoeging van **div** elementen rondom de reeds bestaande elementen.

Bekijk je webpagina al even door de **alt+b** toetsencombinatie in VS Code, of via de rechtermuisknop in de Explorer met de keuze '**Open With Live Server**'.

Zoals we zonet dus zagen is een div een element dat niet onmiddellijk een zichtbare opmaak heeft in tegenstelling tot **p** en **h1..h6** elementen en dergelijke. Een **div** element (*of division*) is net als het **p** en de **h1..h6** elementen een **block-level element** en deze zullen steeds de volledig beschikbare ruimte innemen.

Een **span** element heeft ook geen zichtbare opmaak, in tegenstelling tot de elementen die we reeds gebruikten. Maar het grote verschil tussen een **div** en een **span** element zit hem in het standaard

gedrag. Een **span** element is nu eenmaal een **inline element** van nature en zal de huidige regel dus niet onderbreken. We voegen op onze pagina ook enkele **span** elementen toe.

```
<div>
    <h1>Leonardo da Vinci</h1>
    <p>Leonardo da Vinci ... uit de Renaissance.</p>
</div>
<div>
    <h2>Kunst</h2>
    <p><span>Leonardo</span> is beroemd ... Het Laatste Avondmaal.</p>
    <p>Slechts zeventien ... <span>Engeland</span>. Zij zijn eigendom van Koningin Elisabeth.</p>
</div>
<div>
    <h2>Wetenschap en techniek</h2>
    <p>Leonardo's wetenschappelijke ... op basis van tekeningen.</p>
    <p>Zijn boeken ... mechanische rekenmachine voorstelt.</p>
    <p>In 1502 maakte ... volgens hetzelfde principe.</p>
</div>
```

Toevoeging van **span** elementen rondom de reeds bestaande elementen.

Bekijk je webpagina al even door de **alt+b** toetsencombinatie in VS Code, of via de rechtermuisknop in de Explorer met de keuze ‘**Open With Live Server**’.

We zien dat er wat opmaak betreft ook hier niets gewijzigd is door het toevoegen van de **span** elementen.

Beide elementen hebben als doel om bepaalde tekst te bevatten en er **nadien met CSS** een opmaak aan toe te kennen. Dit is de werkwijze die het meest zal toegepast worden in de cursus. We definiëren met andere woorden of een bepaald stuk tekst een blok of een stuk in een regel moet zijn. Nadien bepalen we de opmaak ervan zoals fontgrootte, lettertype, witruimte, uitlijning, enz... met behulp van CSS.



### Oefening

Aansluitend op het eerste deel kan je **oefening 01 a** uit **oefenreeks 01** uitwerken om zelf even aan de slag te gaan met alle elementen die we tot hiertoe besproken hebben.



[Oefening 01 a via Github](#)

## 2.2 HET CSS BOX MODEL

Voor we overschakelen naar ons eerste stukje CSS staan we eerst even stil bij het CSS Box Model.

### 2.2.1 WAT IS HET CSS BOX MODEL?

Alles in CSS heeft eigenlijk een ‘**box**’ om zich heen. Deze ‘**box**’ bestaat uit een viertal **vakken**, waar het begrijpen van hoe deze vakken werken essentieel is om lay-outs met CSS te kunnen maken of items uit te lijnen met elkaar. In dit stukje zullen we het **CSS Box model** even onder de loep nemen

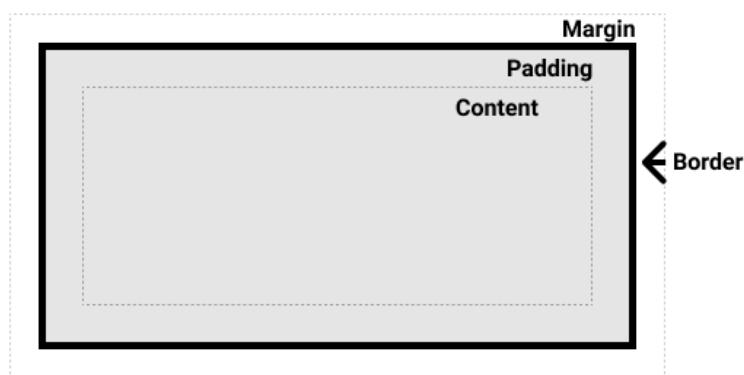
zodat je van start kan met de standaard en later ook de meer complexere lay-outtaken. We bekijken even de werking en de terminologie die hiermee gepaard gaat.

### 2.2.1.1 VAKKEN IN HET CSS BOX MODEL

In een CSS Box Model kunnen we vier vakken gaan onderscheiden:

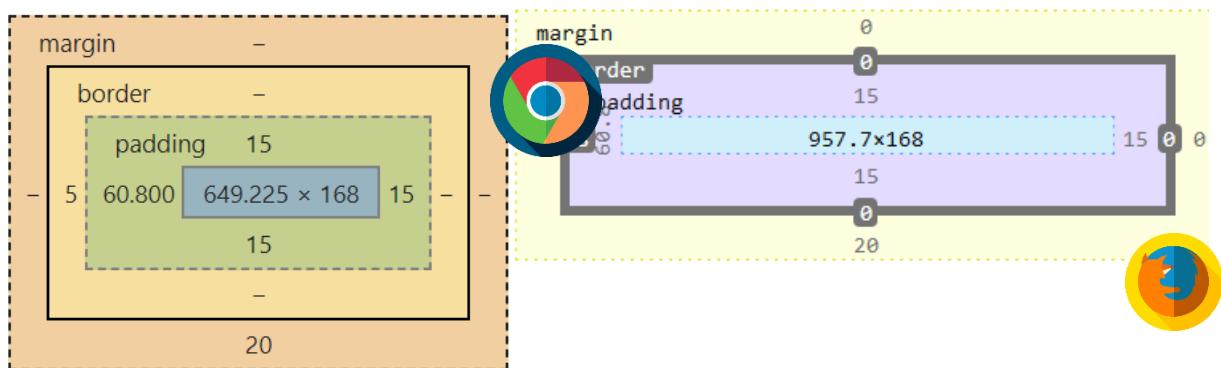
- **Content** (*inhoud*): het gebied waar de inhoud van het element wordt weergegeven. Aanpasbaar met eigenschappen zoals **width** en **height**.
- **Padding** (*opvulling*): de opvulling die zich bevindt rondom de inhoud als witruimte. Aanpassing kunnen hierop gebeuren door de property **padding** met bijbehorende eigenschappen aan te spreken.
- **Border** (*rand*): de border zal de content en padding omringen. De grootte en stijl kunnen worden geregeld met behulp van de **border** property met bijhorende eigenschappen aan te spreken.
- **Margin** (*marge*): de marge is de buitenste laag, waarbij de **content**, **padding** en **border** met witruimte tussen dit vak en andere elementen worden omsloten. De grootte kan worden geregeld met behulp van de **margin** property en bijhorende eigenschappen.

Het onderstaande diagram toont deze lagen:



1 Bron: [MDN developer.mozilla.org](https://developer.mozilla.org)

Bij het inspecteren van een element via je browser ziet het CSS Box Model er meestal als volgt uit:



### 2.2.2 INLINE EN BLOCK ELEMENTEN

Er bestaan hoofdzakelijk twee weergavetypes voor elementen: **Block** en **Inline**. **Block elementen** (zoals **p** en **div**) zullen steeds de volledig beschikbare ruimte innemen en geen elementen naast zich tolereren. Het is dus '**onmogelijk**' om twee **p** of **div** elementen **naast** elkaar te plaatsen. **Inline**

**elementen** blijven netjes in dezelfde regel staan en nemen enkel de plaats in die ze nodig hebben op basis van hun inhoud. Inline elementen die we reeds gezien hebben zijn de **cite**, **mark**, **a** en **img**.

In het onderstaand stukje code zijn de **block elementen** in het geel en de **inline elementen** in het groen gemerkeerd.

```
<div>
  <p>
    De haas en de schildpad is een fabel door <cite>Aesopus</cite>
  </p>
  <p>Hij was een griekse verteller uit de oudheid.
    
  </p>
</div>
```

Het is heel belangrijk om hiermee rekening te houden aangezien een element steeds Inline of Block is. Het gedrag hiervan heeft grote gevolgen voor het uitzicht van de pagina.

Verder in deze cursus zullen we geregeld in CSS aan de slag gaan om de properties van onze elementen en het CSS Box Model aan te passen naar onze behoeften.

## 2.3 ONZE EERSTE CASCADING STYLE SHEET

Tot slot gaan we in dit hoofdstuk ook al even van start met CSS. Verder in de cursus komen we hier uiteraard ook geregeld op terug en verdiepen we ons ook in meer detail in het CSS verhaal.

### STAP 1

We maken een nieuwe pagina aan in VS Code en zorgen voor de standaard html 5 syntax als startpunt. Eenmaal je pagina klaarstaat voorzien we deze van wat HTML code om zo meteen op zijn minst een aantal elementen te hebben waar we een stijl op kunnen toepassen.

### STAP 2

Voeg onderstaande HTML toe in het **body** element van je HTML pagina.

```
<h1>Introductie van CSS</h1>
<p>CSS wordt gebruikt voor de opmaak en het stijlen van onze pagina.</p>
<h2>CSS is tof!</h2>
<p>CSS is <span>super tof</span> om te gebruiken.</p>
```

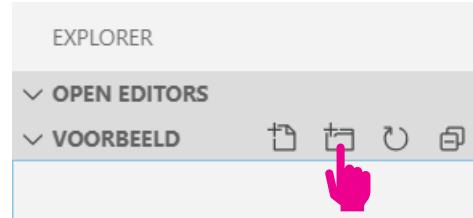
Basis html invulling voor je pagina

### STAP 3

Vervolgens maken we een nieuwe folder aan waar we onze Cascading Stylesheet in onder zullen brengen.

Voeg via de explorer een **nieuwe folder** toe.

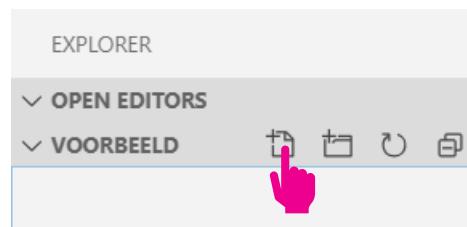
Dit doe je door op het icoontje **New Folder** te klikken naast de naam van je folder. Geef je nieuwe folder de naam **css**



#### STAP 4

Selecteer je nieuwe folder en klik vervolgens op **nieuw bestand**, deze keer een css bestand, **toevoegen** aan deze folder.

Dit doe je door op het icoontje **New File** te klikken naast de naam van je folder. Geef je nieuw bestand de naam **mijnStylesheet.css**



Denk aan de extensie van het bestand! Dit is deze keer niet .html maar .css

#### STAP 5

Maak de relatie van je stylesheet op de pagina waar je deze wenst te gebruiken duidelijk. Dit doen we door in het **head** element een regel toe te voegen.

```
<head>
  ...
  <title>CSS Stylesheet introductie</title>
  <link rel="stylesheet" type="text/css" href="css/mijnStylesheet.css">
</head>
```

Toevoegen van een link element die de relatie naar de te gebruiken stylesheet verduidelijkt

Even verduidelijken wat we hier net aan onze code toevoegden.

Een **link** element specificert relaties tussen het huidige document en een externe bron. Dit element wordt meestal gebruikt om te linken naar stylesheets, maar wordt ook gebruikt om onder andere sitepictogrammen (*zowel "favicon" -stijlpictogrammen als pictogrammen voor het startscherm en apps op mobiele apparaten*) vast te stellen. Op dit laatste komen we later in deze cursus nog terug.

Het **link** element bevat een aantal attributen die de link die we willen maken verduidelijken aan onze browser.

- **rel:** dit attribuut verduidelijkt de relatie tussen wat we linken en de pagina waar we dit linken. In dit geval vermelden we dat het om een relatie van het type **stylesheet** gaat.
- **type:** wat we als inhoud in deze stylesheet zullen voorzien zal van het type **text/css** zijn.
- **href:** een absolute of relatieve link naar **de locatie** van deze stylesheet. Met andere woorden, waar kan de browser deze vinden? (*later komen we nog terug op het concept absolute en relatieve paden*)

Nu onze stylesheet aangemaakt is en we de link tussen onze stylesheet en onze pagina gelegd hebben, is het tijd om over te gaan naar onze eerste regels CSS syntax en meteen ook een eerste manier te zien over hoe we nu iets kunnen selecteren.

### 2.3.1 CSS SELECTOR OP ELEMENT

Wat CSS syntax betreft zullen er nog heel wat zaken de revue passeren in het verdere verloop van deze cursus. Maar voor deze introductie beginnen we met de meest eenvoudige selector die we in CSS hebben. En dat is namelijk de selector op een element.

We gaan van start in onze css stylesheet door een selectie te declareren die een stijl zal toepassen op al onze **p** elementen. Dit doen we door onderstaande code toe te voegen aan onze stylesheet.

```
p {  
    margin-top: 0;  
    margin-right: 25px;  
    margin-bottom: 50px;  
    margin-left: 20px;  
}
```

Een eerste selectie op **p** element met een aanpassing op de margin property van het Box Model van het element

Zoals je ziet heeft onze css een ietwat andere stijl dan onze HTML wat syntax betreft. Een stijlregel bestaat uit twee delen:

- **de selector:** deze bepaalt op welke elementen de regel van toepassing is
- **de stijldeclaratie:** deze bevat de eigenlijke opmaak instructies, die telkens bestaan uit een **eigenschap** en een **waarde (property/value)**, **gescheiden door een dubbele punt** en elke regel wordt afgesloten door een **puntkomma**.

In bovenstaande CSS spreken we de eigenschap aan van de margin (marge) van een **p** element. Specifiek elke mogelijke zijde van de marge. We zijn echter niet verplicht om steeds alle waarden te gebruiken. We kunnen hier bijvoorbeeld enkel de marge aan de rechterzijde vergroten of verkleinen zonder alle andere zijden te specifiëren.

Bekijk het eerste resultaat gerust even via de inspector van je browser om te zien wat je nu net aangepast hebt.

Je ziet meteen dat een selectie op elementnaam **alle** elementen met bijhorende naam zal aanpassen.

Maar dit kan ook korter, de properties van ons Box Model hebben allen ook een syntax die ons toestaan om de wijzigingen op 1 regel te schrijven. Pas gerust de code aan naar onderstaande, en zo zien we ook meteen wat de syntax voor commentaar is in CSS.

```
p {  
    /* margin-top: 0;  
    margin-right: 25px;  
    margin-bottom: 50px;  
    margin-left: 20px; */  
    margin: 0 25px 50px 20px;  
}
```

Een verkorte versie van een aanpassing op de margin property van het Box Model van het element

In CSS zal **/\*** commentaar openen en **\*/** zal de commentaar sectie terug gaan sluiten. In bovenstaande verwerken we meteen alle gewenste waarden van onze margin op één regel. Hier is steeds de volgorde belangrijk in gedachten te houden, de eerste waarde die je typt start bovenaan, vervolgens volg je wijzerszin mee. Dus boven, rechts, onder en links zijn de waarden van de posities.

Wens je alle zijden een gelijke waarde te geven dan kan je dit zonder herhaling gaan doen, dit door enkel een waarde mee te geven.

```
p {  
    margin: 50px;  
}
```

*Alle zijden van de margin instellen op 50 pixels breed.*

Laten we een tweede property uit onze Box Model uitproberen. Stel de padding van een **p** element in op 20 pixels.

```
p {  
    margin: 0 25px 50px 20px;  
    padding: 20px;  
}
```

*CSS syntax om de padding van het p element aan te passen naar 20 pixels overeen alle zijden.*

Nu we zien dat dit ook prima werkt, kunnen we ook even een andere property op een ander element gaan testen. We voegen een selector toe die ons op het **h1** element een **border** zal geven. Voeg deze CSS selector onder de selector van het **p** element op een nieuwe regel toe.

```
p {  
...  
}  
  
h1{  
    border: 3px solid black;  
}
```

*CSS syntax om de h1 elementen met een zwarte rand, 3 pixels dik te voorzien.*

Bekijk je webpagina even door de **alt+b** toetsencombinatie in VS Code, of via de rechtermuisknop in de Explorer met de keuze ‘**Open With Live Server**’.

We kunnen niet alleen de properties die in onze Box Model voorkomen gaan manipuleren met CSS, maar ook alle andere zaken op onze elementen kunnen we gaan beïnvloeden. In onderstaande CSS syntax selecteren we alle **span** elementen en we voorzien deze van wat opmaak op de tekst zelf. Ter illustratie ook meteen drie mogelijkheden om een kleur te definiëren in CSS, hier komen we later in de cursus ook nog op terug.

```
span {  
    font-weight: bold;  
    font-style: italic;  
    color: red;           /* optie 1 */  
    color: rgb(255,0,0);  /* optie 2 */  
    color: #ff0000;       /* optie 3 */  
}
```

*CSS syntax om de span elementen hun tekst inhoud van de nodige opmaak te voorzien.*

Als laatste voegen we nog een selector toe, om aan te tonen dat we eender welk bestaand element kunnen gebruiken om in onze CSS selectoren mee aan de slag te gaan, zelfs **html** en **body** als dit nodig is.

```
body{  
    background-color:orange;  
}  
  
h2{  
    background-color:greenyellow;  
}
```

CSS syntax om de achtergrondkleur van het **body** en **h2** element aan te passen

Onderstaand een screenshot van hoe je scherm er ongeveer zal uitzien na het volgen van bovenstaande stappen.

## Introductie van CSS

CSS wordt gebruikt voor de opmaak en het stylen van onze pagina.

**CSS is tof!**

CSS is *super tof* om te gebruiken.



### Oefening

Aansluitend op het eerste deel kan je **oefening 01 b** uit **oefenreeks 01** uitwerken om zelf even aan de slag te gaan met alle elementen die we tot hiertoe besproken hebben.

 [Oefening 01 b via Github](#)





# Basis HTML elementen

Web Frontend Basics

## INHOUD

|                                |           |
|--------------------------------|-----------|
| <b>1 BASIS HTML ELEMENTEN</b>  | <b>3</b>  |
| <b>1.1 b element</b>           | <b>3</b>  |
| <b>1.2 strong element</b>      | <b>3</b>  |
| 1.2.1 b of strong?             | 4         |
| <b>1.3 i element</b>           | <b>5</b>  |
| <b>1.4 em element</b>          | <b>5</b>  |
| <b>1.5 u element</b>           | <b>5</b>  |
| <b>1.6 mark element</b>        | <b>7</b>  |
| <b>1.7 cite element</b>        | <b>7</b>  |
| <b>1.8 small element</b>       | <b>8</b>  |
| <b>1.9 code element</b>        | <b>8</b>  |
| <b>1.10 samp element</b>       | <b>8</b>  |
| <b>1.11 kbd element</b>        | <b>8</b>  |
| <b>1.12 pre element</b>        | <b>9</b>  |
| <b>1.13 sup en sub element</b> | <b>9</b>  |
| <b>1.14 blockquote element</b> | <b>10</b> |
| <b>1.15 q element</b>          | <b>10</b> |

## 3 BASIS HTML ELEMENTEN

In dit deel van de cursus nemen we een aantal basis HTML-elementen onder de loep en bespreken we hun doel en werking.

### 3.1 B ELEMENT

Type: *inline element*

Het **b** element (*Bring Attention To element*) wordt gebruikt om de aandacht van de lezer te trekken, zonder extra betekenis toe te voegen.

De *user-agent* zal in de meeste gevallen de opmaak van de inhoud van het **b** element in het vet weergeven. Gebruik dit element echter niet om snel een stukje tekst van stijl te voorzien. Hiervoor ga je aan de slag met **CSS** en de *font-weight* property.

Het **b** element wordt hoofdzakelijk gebruikt om sleutelwoorden in een tekst of productnamen in een recensie te omsluiten. Het wordt ook vaak toegepast om de eerste zin van een artikel in te plaatsen om er de nodige aandacht van de lezer mee te trekken.

```
<p>In dit hoofdstuk zullen wat meer leren over <b>HTML</b> elementen.</p>
```

Voorbeeld gebruik van het **b** element



#### Enkele richtlijnen en tips voor het gebruik:

- Verwar het **b** element niet met de elementen **strong**, **em** of **mark**.
  - Het **strong** element geeft tekst weer met een bepaald belang
  - Het **em** element legt enige nadruk op de tekst
  - Het **mark** element geeft tekst aan met een zekere relevantie.
 Het **b** element geeft geen betekenis mee, gebruik het daarom alleen als de andere elementen niet passen. Zie verder voor de verduidelijking van bovenvermelde elementen.
- Gebruik het **b** element niet om koppen aan te geven. Hiervoor zijn de **h1** tot en met **h6** elementen ter beschikking.

Bron: [Mozilla Developer Network - b element](#)

### 3.2 STRONG ELEMENT

Type: *inline element*

Het **strong** element is bedoeld voor inhoud die van groot belang is, zaken van grote ernst of urgentie (*zoals waarschuwingen*). Dit kan zelfs een zin zijn die van groot belang is voor de hele pagina, of je kan erop wijzen dat sommige woorden van groter belang zijn dan inhoud in de buurt.

```
<p>
  Voor je verder gaat, <strong>zorg ervoor dat je je veiligheidsbril aan hebt</strong>.
</p>
```

Voorbeeld gebruik van het **strong** element

De *user-agent* zal in de meeste gevallen de opmaak van de inhoud van het **strong** element in het vet weergeven. Gebruik dit element echter niet om snel een stukje tekst van stijl te voorzien. Hiervoor ga je aan de slag met **CSS** en de *font-weight* property.

Een ander geaccepteerd gebruik van het **strong** element is om de labels van alinea's aan te duiden die opmerkingen of waarschuwingen in de tekst van een pagina vertegenwoordigen.

```
<p><strong>Belangrijk:</strong> Wees voorzichtig tijdens het rijden in de mist.</p>
```

Voorbeeld gebruik van het *strong* element

### 3.2.1 B OF STRONG?

Het kan vaak verwarrend zijn als je van start gaat met HTML waarom er zoveel verschillende manieren zijn om iets op het scherm te plaatsen wat er voor de gebruiker uiteindelijk precies hetzelfde uitziet. Het **b** en **strong** element zijn hier beiden misschien een van de meest voorkomende bronnen van verwarring. Waardoor al vaak de vraag komt: "Moet ik nu **b** of **strong** gebruiken? Doen ze niet allebei hetzelfde?"

Wel, niet precies. Het **strong** element is voor inhoud die van groter belang is, terwijl het **b** element wordt gebruikt om de aandacht op tekst te vestigen zonder aan te geven dat het belangrijker is.

Het kan helpen om te beseffen dat beide geldige en semantische elementen zijn en dat het toeval is dat ze in de meeste browsers dezelfde standaardstijl (*vetgedrukt*) hebben. Elk element heeft echter zijn plaats om gebruikt te worden gebruikt in bepaalde scenario's. En het is net het doel, de betekenis wat bepalen zal welk element je gebruikt en niet het visuele resultaat in de browser.

### 3.3 I ELEMENT

---

Type: *inline element*

Het **i** element wordt gebruikt om inhoud te omsluiten die om de een of andere reden afwijkt van de normale tekst. Voorbeelden hiervan zijn technische termen, uitdrukkingen in een vreemde taal of gedachten van fictieve personages. We zien dan ook vaak het **lang** attribuut op dit element opduiken om verdere verduidelijking te geven. Zorg ervoor dat de tekst in kwestie niet beter geschikt is voor een ander element.

De *user-agent* zal in de meeste gevallen de opmaak van de inhoud van het **i** element cursief weergeven. Gebruik dit element echter niet om snel een stukje tekst van stijl te voorzien. Hiervoor ga je aan de slag met **CSS** en de *font-style* property.

```
<p>In het Frans noemen ze dit <i lang="fr">à l'improviste</i>.</p>
<p><i lang="la">Taraxacum officinale</i> is
door ons beter gekend als paardenbloem.</p>
```

Voorbeeld gebruik van het *i* element

### 3.4 EM ELEMENT

---

Type: *inline element*

Het **em** element (*Emphasis*) legt op de inhoud een zekere nadruk. **em** elementen kunnen genest worden in elkaar waarbij elk niveau van nesten een grotere mate van nadruk aangeeft. Dit element is bedoeld om een nadruk op woorden te leggen ten opzichte van de omringende tekst. Vaak beperkt de inhoud van dit element zich tot een woord of woorden van een zin en heeft invloed op de betekenis van de zin zelf.

De *user-agent* zal in de meeste gevallen de opmaak van de inhoud van het **em** element cursief weergeven. Gebruik dit element echter niet om snel een stukje tekst van stijl te voorzien. Hiervoor ga je aan de slag met **CSS** en de *font-style* property.

```
<div>
  <h2>EM element</h2>
  <p>Laten dit <em>nu</em> doen!</p>
  <p>We moeten hier <em>dringend</em> iets aan doen.</p>
</div>
```

Voorbeeld gebruik van het *em* element

### 3.5 U ELEMENT

---

Type: *inline element*

Het **u** element (*Unarticulated Annotation element*) gebruik je voor om tekst te markeren als een vorm van niet-tekstuele annotatie. Wat hiermee concreet bedoeld wordt, is dat dit een verklarende visuele 'opmerking' toevoegt ter verduidelijking van de tekst die omsloten werd. Geldige scenario's waarin je

dit element kan gebruiken zijn onder meer het annoteren van spelfouten, het toepassen van een **proper name mark** om eigen namen in Chinese tekst aan te duiden (*zal je hoogstwaarschijnlijk nooit tegenkomen, maar even meegeven om volledig te zijn*) en andere vormen van annotatie.

De *user-agent* zal in de meeste gevallen de opmaak van de inhoud van het **u** element onderlijnen. Gebruik dit element echter niet om snel een stukje tekst van stijl te voorzien. Hiervoor ga je aan de slag met **CSS** en passende properties.

In vele gevallen zal je ook nog de nodige aanpassingen maken via CSS om de gebruiker een visueel herkenbare onderlijning weer te geven. Op deze manier onderscheid je ook meteen de interpretatie van een gebruiker dat iets wat onderlijnd is veelal een hyperlink voorstelt.

In onderstaand voorbeeld wordt hiervoor gebruik gemaakt van een class attribuut om de nodige bijkomende opmaak te voorzien op het element. Het class attribuut en hoe je hiermee omgaat in CSS wordt in het volgend hoofdstuk besproken.

```
<p>Deze alinea bevat een woord dat <u class="spelling">verkreed</u> gespeld werd.</p>
```

Voorbeeld gebruik van het **u** element samen met een class attribuut

```
.spelling {
    text-decoration: red wavy underline;
}
```

Voorbeeld van de CSS selector op class attribuut

**Deze alinea bevat een woord dat verkreed gespeld werd.**

Voorbeeld output in een browser

Indien je enkel maar op zoek bent naar een manier om te onderlijnen, maar het **u** element niet relevant is om te gebruiken dan kan je op soortgelijke manier aan de slag gaan. Maak dan gebruik van een niet semantisch element omheen de inhoud van de tekst die je wil onderlijnen. We bekijken even een voorbeeld.

```
<span class="underline">Dit willen we onderlijnen:</span>
<br />
```

Vervolgens komt dit op een nieuwe lijn.

Voorbeeld gebruik van het **span** element samen met een class attribuut om stijl te voorzien

```
.underline {
    text-decoration: underline;
}
```

Voorbeeld van de CSS selector op class attribuut

**Dit willen we onderlijnen:**  
Vervolgens komt dit op een nieuwe lijn.

Voorbeeld output in een browser

## 3.6 MARK ELEMENT

Type: *inline element*

Het **mark** element vertegenwoordigt tekst die is gemarkerd voor referentie- of notatiedoeleinden.

Dit kan zijn vanwege de relevantie, of het belang van de gemarkerde passage in de omsluitende context.

Als output zal het **mark** element de inhoud in het fluo aanduiden. Dit element wordt voornamelijk gebruikt in een dynamische context waar men bijvoorbeeld - aan de hand van javascript - woorden uit een zoekopdracht zal markeren in een stuk tekst.

```
<p>In deze tekst kunnen we gaan markeren dat het <mark>correct gebruik van HTML Elementen </mark> zeker en vast van belang is!</p>
```

Voorbeeld gebruik van het mark element

In deze tekst kunnen we gaan markeren dat het correct gebruik van **HTML Elementen** zeker en vast van belang is!

Voorbeeld output in een browser

## 3.7 CITE ELEMENT

Type: *inline element*

Het **cite** element (*Citation*) element wordt gebruikt om een verwijzing naar een geciteerd creatief werk te beschrijven en moet de titel van dat werk bevatten.

In de context van het **cite** element zou een creatief werk bijvoorbeeld een van de volgende kunnen zijn:

- Een boek
- Een onderzoeksrapport
- Een verhandeling
- Een gedicht
- Een muzikale score
- Een lied
- Een toneelstuk of filmscript
- Een film
- Een televisieprogramma
- Een spel
- Een sculptuur
- Een schilderij
- Een theatrale productie
- Een toneelstuk
- Een opera
- Een musical
- Een tentoonstelling
- Een juridisch dossier
- Een computerprogramma
- Een website
- Een webpagina
- Een blogbericht of reactie
- Een forumbericht of reactie
- Een tweet
- Een Facebook-bericht
- Een schriftelijke of mondelijke verklaring
- ...

De *user-agent* zal in de meeste gevallen de opmaak van de inhoud van het **cite** element cursief weergeven. Wens je dit liever niet, dan kan je uiteraard altijd aan de slag met **CSS** en de *font-style* property om dit op te heffen of je eigen opmaak te voorzien.

```
<p>Een aan te raden boek voor alle toekomstige developers is <cite>Clean Code</cite>  
geschreven door Uncle Bob.</p>  
<p>Meer informatie kan je terugvinden in <cite>[ISO 20000-1]</cite>.</p>
```

Voorbeeld gebruik van het cite element

## 3.8 SMALL ELEMENT

Type: inline element

Het **small** element vertegenwoordigt rand commentaar en ‘de kleine lettertjes’, zoals copyright, auteursrecht en juridische tekst(en). Standaard maakt het tekst binnenden één lettergrootte kleiner (bv. van *big* naar *medium* of van *small* naar *x-small*).

```
<small>Een of meerdere regels copyright en/of disclaimers</small>
```

Voorbeeld gebruik van het *small* element

## 3.9 CODE ELEMENT

Type: inline element

Het **code** element heeft tot doel om een kort fragment van computercode te omsluiten. De inhoud wordt meestal weergegeven met het standaard monospaced lettertype van de *user agent* (zoals Courier of Lucida Console).

```
<p>Roep vervolgens de functie <code>updateTextElements()</code> aan via javascript.</p>
```

Voorbeeld gebruik van het *code* element



### Aandachtspunt

Als je meerdere coderegels wenst weer te geven is het de afspraak om het **code** element in een **pre** element (*zie verder*) te plaatsen. Het **code** element vertegenwoordigt op zichzelf slechts een fragment of een enkele coderegel.

## 3.10 SAMP ELEMENT

Type: inline element

Het **samp** element (*Sample*) wordt gebruikt om tekst in te sluiten die een voorbeeld-output (*of citaat*) van een computerprogramma weergeeft. De inhoud wordt meestal weergegeven met het standaard monospaced lettertype van de *user agent* (zoals Courier of Lucida Console).

```
<p>
Na uitvoeren van de applicatie krijg je in de console <samp>Update Successful</samp> te zien.
</p>
<p><samp>Keyboard not found <br /> F1 to continue</samp></p>
```

Voorbeeld gebruik van het *samp* element

## 3.11 KBD ELEMENT

Type: inline element

Het **kbd** element (*Keyboard Input element*) geeft een gebruikersinvoer aan die gerelateerd kan zijn aan een toetsenbord, spraakinvoer of een ander tekstinvoerapparaat. De inhoud wordt meestal weergegeven met het standaard monospaced lettertype van de *user agent* (zoals Courier of Lucida Console).

```
<p>Om iets te kopiëren, dien je <kbd>CTRL</kbd> + <kbd>C</kbd> in te drukken.</p>
```

Voorbeeld gebruik van het *kbd* element

### 3.12 PRE ELEMENT

Type: *block element*

Het **pre** element (*Preformatted*) staat voor voorgeformatteerde tekst. Met andere woorden tekst die exact moet worden gepresenteerd zoals geschreven in het HTML-bestand. De inhoud wordt meestal weergegeven met het standaard monospaced lettertype van de *user agent* (zoals Courier of Lucida Console). Witruimte binnen dit element wordt, in tegenstelling tot wat er normaal gebeurt bij het renderen, wel weergegeven.

```
<pre>
body {
    color:red;
}
</pre>
```

Voorbeeld gebruik van het *pre* element

```
body {
    color:red;
}
```

Voorbeeld output in een browser



#### Aandachtspunt

Misbruik dit element niet om alles een plaats (bv.: aliniëren, spatiëren, e.d.) te geven op je pagina. Gebruik dit enkel voor kleine fragmenten waar dit nodig/passend is. Bijvoorbeeld in combinatie met het **code** element.

### 3.13 SUP EN SUB ELEMENT

Type: *inline elementen*

Het **sup** element (*Superscript*) en het **sub** element (*Subscript*) geeft inline-tekst aan die uitsluitend om typografische redenen een andere weergave vereist. Superscript wordt meestal weergegeven met een verhoogde basislijn met kleinere tekst en subscript wordt meestal weergegeven met een verlaagde basislijn met kleinere tekst. Veelal gebruikt voor wiskundige notaties.

```
<p>x<sup>2</sup> = y en x<sub>1</sub> ... x<sub>n</sub></p>
```

Voorbeeld gebruik van het sup en sub element

$$x^2 = y \text{ en } x_1 \dots x_n$$

Voorbeeld output in een browser

### 3.14 BLOCKQUOTE ELEMENT

Type: block element

Het **blockquote** element (*Block Quotation element*) geeft aan dat de omsloten tekst een uitgebreid citaat is. Meestal wordt dit visueel ook weergegeven door het inspringen van de betreffende tekst. Een URL voor de bron van het citaat kan worden gegeven met behulp van het **cite attribuut** (*niet verwarren met het cite element*), terwijl een tekstweergave van de bron dan weer kan worden meegegeven met behulp van het **cite element**.

```
<blockquote cite="https://cleancoders.com/">
  <p>
    You should name a variable using the same care with which you name a first-born child.
  </p>
  <cite>Clean Coders</cite>
</blockquote>
```

Voorbeeld gebruik van het blockquote element, samen met het cite attribuut en het cite element

### 3.15 Q ELEMENT

Type: inline element

Het **q** element (*Quotation*) geeft aan dat de ingesloten tekst een kort citaat is. De meeste moderne browsers implementeren dit door de tekst tussen aanhalingstekens te plaatsen. Dit element is dus bedoeld voor korte citaten waarvoor geen alinea-eindes nodig zijn. Gebruik voor lange(re) citaten steeds het **blockquote** element wat we hier net voor zagen.

```
<p>
  <q cite="http://www.caesar.com" lang="la">Veni, vidi, vici</q> is een gekende
  uitspraak
  van Caesar.
</p>
```

Voorbeeld gebruik van het q element

"Veni, vidi, vici" is een gekende uitspraak van Caesar

Voorbeeld output in een browser

## abbr element

Type: *inline element*

Het **abbr** element (*Abbreviation element*) laat ons toe om een afkorting aan te duiden en geeft ons bijkomend de mogelijkheid om de volledige omschrijving ter verduidelijking toe te voegen. Indien het **title** attribuut wordt benut in het element, hoort deze de volledige omschrijving te bevatten en niets anders.

```
<p>Maak gebruik van <abbr title="Cascading Style Sheets">CSS</abbr> om je <abbr title="HyperText Markup Language">HTML</abbr> stijl te geven.</p>
```

Voorbeeld gebruik van het **abbr** element



### Oefening

Aansluitend kan je met **oefening 01 c** uit **oefenreeks 01** uitwerken om zelf even aan de slag te gaan met alle elementen die we tot hiertoe besproken hebben.

[Oefening 01 c via Github](#)





# Hyperlinks

Web Frontend Basics

## INHOUD

|  |          |
|--|----------|
| <b>4. HYPERLINKS</b>                               | <b>3</b> |
| <b>4.1 A element</b>                               | <b>3</b> |
| <b>4.2 Relatief en absoluut</b>                    | <b>3</b> |
| 4.2.1 Bestandslocatie en mappenstructuur           | 3        |
| 4.2.2 Absoluut linken                              | 5        |
| 4.2.3 Relatief linken                              | 5        |
| <b>4.3 Verwijzing binnen dezelfde pagina</b>       | <b>6</b> |
| <b>4.4 Mogelijke waarden in het href attribuut</b> | <b>7</b> |

## 4. HYPERLINKS

### 4.1 A ELEMENT

Type: *inline element*

Het **a** element (*Anchor element*) stelt ons in staat om onze gebruikers te laten navigeren doorheen onze website, naar andere pagina's. Maar we kunnen deze ook gebruiken om links te leggen naar andere websites of zelfs om binnen dezelfde pagina de gebruiker door te verwijzen zonder dat deze moet scrollen.

Het **a** element gaat steeds gepaard met het **href** attribuut. De waarde van dit **href** attribuut bevat de nodige informatie over waar we de gebruiker precies heen willen sturen als hij op het **a** element zal klikken.

```
<a href="https://google.com">Ga naar Google</a>  
<a href="een-pagina.html">Ga naar deze pagina</a>
```

Voorbeeld gebruik van het **a** element in combinatie met het **href** attribuut

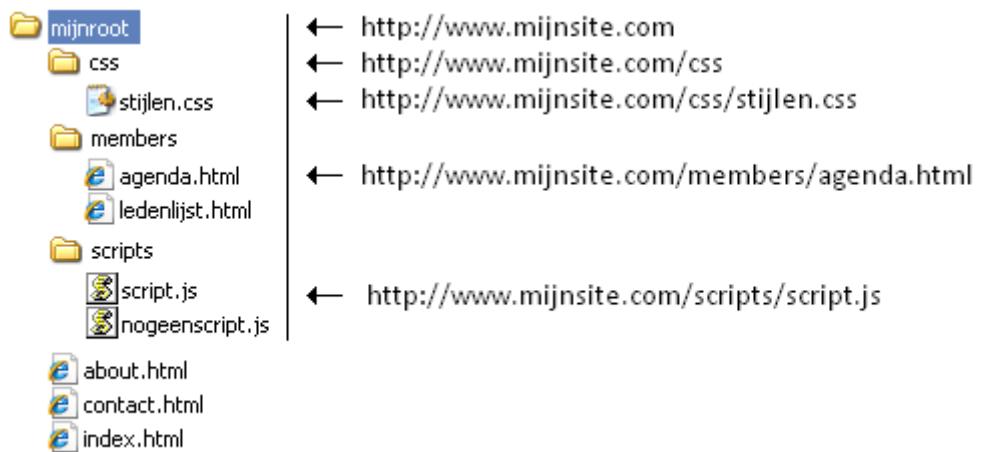
### 4.2 RELATIEF EN ABSOLUUT

Er bestaan twee manieren om te linken naar andere webpagina's: relatief en absoluut. Vooraleer we dit aanleren bekijken we eerste de algemene mappenstructuur van documenten op een website.

#### 4.2.1 BESTANDSLOCATIE EN MAPPENSTRUCTUUR

De manier waarop de verschillende bestanden op een website georganiseerd zijn, is meestal naargelang de smaak en de logica van de ontwikkelaar. Het is echter belangrijk om het nut in te zien van een logische structuur, zodat er vervolgens op een eenduidige wijze gerefereerd kan worden naar eender welk bestand in de mappenstructuur op een website.

De organisatie van bestanden op een site is identiek als de ordening van mappen en bestanden op je eigen harde schijf. De belangrijkste map op een website is de "Root" map. Hierin staat meestal de hoofdpagina van een site. Er kunnen mappen aanwezig zijn die pagina's in categorieën onderbrengen, mappen die CSS bestanden bevatten en mappen die javascript bestanden bevatten. Een bestandsstructuur is bepalend voor de benodigde URL om dat bestand te downloaden vanaf een website zoals blijkt uit het volgende diagram:



## 4.2.2 ABSOLUUT LINKEN

Een absolute hyperlink betekent dat we de volledige URL van het webdocument refereren. Bijvoorbeeld: <https://www.howest.be/nl/opleidingen/graduaat/programmeren> is een geldige waarde voor het **href** attribuut van een **a** element en deze hyperlink zal doorverwijzen naar de betreffende pagina.

```
<a href="https://www.howest.be/nl/opleidingen/graduaat/programmeren">Onze  
opleiding</a>
```

Voorbeeld gebruik van het *a* element in combinatie met het *href* attribuut

Deze manier is de enige manier om te linken naar een pagina/resource op een externe website. Het is echter uit den boze om op deze manier (*absoluut linken*) te linken naar pagina's, figuren, etc,... binnen dezelfde website. Daarvoor gebruiken we relatieve verwijzingen.

## 4.2.3 RELATIEF LINKEN

Een relatieve link vertrekt steeds van de locatie (*map*) waarin het **huidige html-bestand** zich bevindt. Hierbij is het belangrijk om de mappenstructuur van je website te kennen. Bekijk nogmaals het diagram met de mappenstructuur en bekijk de volgende link die in onze **index.html** aanwezig zou kunnen zijn:

```
<a href="members/agenda.html">Link vanuit index.html naar de agenda pagina</a>
```

Voorbeeld gebruik van het relatief linken van een pagina

Indien we nu van **agenda.html** terug willen linken naar de index pagina, dan moeten we in principe een mapje 'hoger' gaan. Telkens we naar een bovenliggende map willen verwijzen, voegen we een **..**/ in.

```
<a href="../index.html">Link vanuit agenda.html naar de index pagina</a>
```

Voorbeeld met het terugkeren naar een bovenliggende map

## 4.3 VERWIJZING BINNEN DEZELFDE PAGINA

We kunnen ook binnen dezelfde pagina verwijzen met behulp van het **a** element. In dat geval moeten we ergens op onze pagina iets ‘markeren’, zodat we er ook effectief naar kunnen linken. Dit kunnen we doen door gebruik te maken van een attribuut dat op elk element toegepast kan/mag worden, namelijk het **id** attribuut.

```
<h2 id="markering">Deze hoofding is gemarkerd</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed hendrerit euismod dui
pellentesque venenatis. Nulla tortor ante, pellentesque at lacinia at, commodo at nunc
. Etiam facilisis tempus eros, vitae blandit orci pulvinar nec.</p>
```

Voorbeeld van de toevoeging van een **id** attribuut aan een **h2** element

Nu we een id voorzien hebben, kunnen we ernaar ‘linken’ met behulp van ons **a** element. Dit doen we door het **href** attribuut de volgende waarde toe te kennen: het spoorwegteken (*beter gekend als hashtag, #*) gevolgd door de naam van het **id** attribuut waarheen we willen linken

```
<h2 id="markering">Deze hoofding is gemarkerd</h2>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed hendrerit euismod dui
pellentesque venenatis. Nulla tortor ante, pellentesque at lacinia at, commodo at nunc
. Etiam facilisis tempus eros, vitae blandit orci pulvinar nec.</p>

<p>Klik <a href="#markering">hier</a> om terug naar het begin van de paragraaf te gaan
</p>
```

Voorbeeld van het plaatsen van een verwijzing op een pagina

Om bovenstaand voorbeeld goed te kunnen testen, zal je uiteraard wat meer tekst of witregels moeten voorzien, zodat de hyperlink buiten het bereik van je browserscherm staat. Je kunt je de hoogte van je browser verminderen.

Merk op dat we ook rechtstreeks naar een gemarkerd deel op een andere pagina kunnen linken. Aansluitend op de **URL** die we willen gebruiken, plaatsen we de naam van het gewenste **id** attribuut waarnaar we willen linken voorafgegaan door het spoorwegteken (#).

```
<a href="http://en.wikipedia.org/wiki/HTML_element#anchor" target="_blank">Klik hier</a>
```

Voorbeeld van het linken naar een specifieke plaats op een pagina van een andere website

Het spreekt uiteraard voor zich dat je wel even in de broncode zal moeten kijken van de website waarheen je wil linken om de correcte naam van het anchor terug te vinden.

Er is wel reeds ondersteuning in de browser voorzien bij het gebruik van **#top** of **#** om steeds naar het begin van de pagina terug te gaan zonder dat je hiervoor expliciet een id dient te voorzien.

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

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

Voorbeeld van een link naar de bovenzijde van de pagina

## 4.4 MOGELIJKE WAARDEN IN HET HREF ATTRIBUUT

Bovenop de standaard functionaliteit kunnen we ook gebruik maken van het **a** element om meteen een e-mail of een telefoonnummer te linken.

Dit kunnen we doen door de waarde die we toevoegen in het href attribuut te laten voorafgaan door **mailto:** of **tel:**.

Hieronder vind je hiervan enkele voorbeelden.

```
<a href="mailto:voornaam.naam@voorbeeld.com">E-mail naar ...</a><br />
<a href="tel:+123456789">Telefoon nummer</a><br />
```

Voorbeeld van een invulling voor e-mail of telefoonnummer in het href attribuut



### Oefening

Aansluitend kan je **oefening 01 d** uit **oefenreeks 01** uitwerken om zelf even aan de slag te gaan met alle elementen die we tot hiertoe besproken hebben.

 [Oefening 01 d via Github](#)





# Basis CSS

Web Frontend Basics

## INHOUD

|   |          |
|---|----------|
| <b>5 BASIS CSS</b>                        | <b>3</b> |
| <b>5.1 CSS Selectors</b>                  | <b>3</b> |
| <b>5.2 Element selector</b>               | <b>3</b> |
| <b>5.3 Context selector</b>               | <b>3</b> |
| <b>5.4 Selecteren met klassen en id's</b> | <b>3</b> |
| <b>5.5 De border instellen</b>            | <b>4</b> |
| 5.5.1 Afgeronde hoeken                    | 5        |
| <b>5.6 Opmaak met CSS</b>                 | <b>5</b> |
| 5.6.1 Fonts, tekstdecoratie en tekststijl | 5        |
| 5.6.2 Uitlijning van tekst                | 6        |
| 5.6.3 Witruimte en spatiëring             | 6        |
| 5.6.4 Kleuren                             | 7        |

## 5 BASIS CSS

### 5.1 CSS SELECTORS

De **CSS selector** bepaalt op welke elementen de stijlcode van toepassing is. We hebben reeds gezien dat we met CSS de opmaak van elementen kunnen bepalen door de naam van het element (*bvb p of div*) te gebruiken als selector. In dit deel behandelen we de meest voorkomende manieren om elementen te selecteren met de bedoeling er stijlcode op toe te passen.

### 5.2 ELEMENT SELECTOR

De meest eenvoudige selector is de element selector die de naam van het element gebruikt.



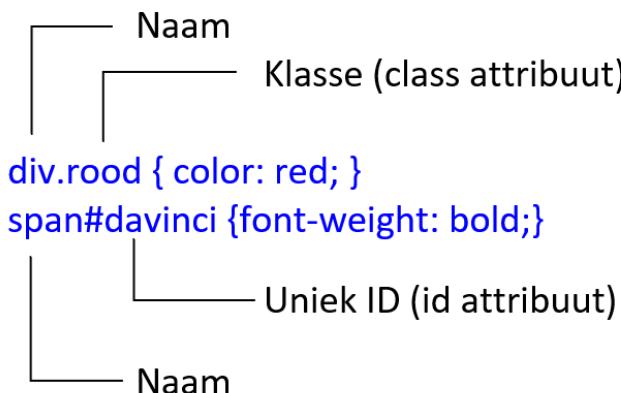
### 5.3 CONTEXT SELECTOR

Deze selector gebruikt de context van het gewenste element. In dit geval wordt de stijl enkel toegepast op **em** elementen die zich in **h1** elementen bevinden. Andere **em** elementen worden niet geselecteerd.



### 5.4 SELECTEREN MET KLASSEN EN ID'S

De eerste selector kiest alle **div** elementen die tot een bepaalde **klasse** behoren (*instellen m.b.h. het class attribuut*). De tweede selector zal van toepassing zijn op het **span** element waarvan het **id** attribuut gelijk is aan ‘*davinci*’.



## 5.5 DE BORDER INSTELLEN

We kunnen de rand (*border*) van het boxmodel instellen met een breedte, stijl en kleur. De rand op zich heeft net als de andere eigenschappen van het boxmodel vier delen (*boven, rechts, onder en links*). Deze delen kunnen we ook afzonderlijk instellen. We deden dit reeds bij de koppeling van onze eerste CSS stylesheet.

| CSS Property        | Beschrijving  |
|---------------------|---|
| <b>border-style</b> | Stijl van de randlijn: <i>none, dotted, dashed, solid, double, groove, ridge, inset of outset</i> |
| <b>border-width</b> | Dikte van de rand in pixels.  |
| <b>border-color</b> | Kleur van de rand   |
| <b>border</b>       | Algemene eigenschap om alles in 1 lijn in te stellen  |

*Border eigenschappen in css*

Indien we een van de vier delen van onze **border** afzonderlijk willen instellen, kunnen we dit door de positie toe te voegen aan de property die we nodig hebben. Bij deze border eigenschappen kunnen dus steeds gebruik maken van; *-top, -right, -bottom, -left* om naar eigen behoefte de waarde te specifiëren op een bepaalde plaats.

```
border-left-width: 5px;  
border-top-color: #000;
```

*Enkele voorbeelden met specifieke positie in de property*



Enkele bijzonderheden van de border:

- Border wordt **niet overgeërfd**
- Je moet minstens de "style" instellen van een border om ze te zien.  
De standaardwaarde is immers *none*.
- Border kan ook gebruikt worden voor *<hr />* elementen en tabellen (zie hoofdstuk over Tables).

Randen konden vroeger verschillen in uiterlijk tussen browsers en besturingssystemen, maar tegenwoordig hebben de meeste browsers dit probleem weggewerkt.

Bron: [Mozilla Developer Network: CSS Border](#)

Wens je een **afbeelding** te gebruiken **als rand**? Dat kan uiteraard ook. Meer informatie hierover kan je ook terugvinden via de [Mozilla Developer Network: border-image](#) pagina.

### 5.5.1 AFGERONDE HOEKEN

Een vaak gehanteerde stijl om de immer vierkante vormen van HTML elementen aan te passen zijn de zogenaamde ‘rounded corners’. Via CSS kunnen we dit bekomen door de property border-radius te manipuleren.

```
/* Radius voor alle 4 hoeken */
border-radius: 10px;

/* top-left | top-right | bottom-right | bottom-left */
border-radius: 1px 0 3px 4px;

/* top-left en bottom-right | top-right en bottom-left */
border-radius: 10px 5%;
```

Voorbeeld gebruik van de radius property voor borders in CSS

## 5.6 OPMAAK MET CSS

### 5.6.1 FONTS, TEKSTDECORATIE EN TEKSTSTIJL

Eén van de meeste belangrijke keuzes voor je website is die van het lettertype voor de body, hoofdingen en andere stukken tekst. Verder willen we ook tekst laten opvallen: schuin of vetgedrukt, onderlijning, enz...

Hieronder vind je enkele belangrijke CSS eigenschappen die tekst een eigen karakter kunnen geven:

| CSS Property           | Beschrijving  |
|------------------------|---|
| <b>font-family</b>     | Specificeert één of meerdere lettertypes, gescheiden door komma's. Indien het eerste lettertype niet beschikbaar is voor de browser wordt de volgende gebruikt.<br>Namen die bestaan uit meerdere woorden moeten <b>tussen dubbele quotes</b> geplaatst worden! |
| <b>font-size</b>       | De tekstgrootte met eenheid<br><b>absoluut:</b> in pixels (px), points (pt), centimeter (cm)<br><b>relatief:</b> percentage (%), ems (em) [1.5em = 150% → <a href="#">meer</a> ], large, small, xx-large, ...   |
| <b>font-weight</b>     | Maat voor vetgedrukte tekst.  |
| <b>font-style</b>      | Toont de tekst schuingedrukt of niet  |
| <b>font-variant</b>    | Wanneer de waarde op ‘small-caps’ is ingesteld zal het lettertype in kleinere hoofdletters weergegeven worden.  |
| <b>font</b>            | Biedt de mogelijkheid om alle voorgaande eigenschappen op 1 lijn in te stellen  |
| <b>text-decoration</b> | Stelt lijnen in voor de tekst: <a href="#">underline</a> , <a href="#">overline</a> , <a href="#">line-through</a>  |

## 5.6.2 UITLIJNING VAN TEKST

Sommige HTML elementen (*met name block elementen*) kunnen hun **tekst** (*en/of inhoud*) een uitlijning geven: left, right, center of justified met behulp van de CSS property **text-align**. Justified zal de zinnen proberen te spreiden over de breedte van het element, net zoals we dat dikwijls vinden in de krantkolommen.

| CSS Property      | Beschrijving   |
|-------------------|--|
| <b>text-align</b> | Stelt de uitlijning in voor de tekst in een element: <b>right, left, center, justified</b> |

`text-align:justify;`

Voorbeeld van het gebruik van de `text-align` property

## 5.6.3 WITRUIIMTE EN SPATIËRING

Om de witruimte en spatiëring van alinea's, zinnen en woorden in te stellen, hebben we ook een aantal CSS eigenschappen tot onze beschikking:

| CSS Property          | Beschrijving  |
|-----------------------|---|
| <b>white-space</b>    | Standaard zal de tekst steeds teruglopen (een nieuwe lijn nemen) op het einde van de regel. Deze terugloop kan voorkomen worden door volgende waarde:<br><b>nowrap</b> : er wordt geen terugloop toegestaan.<br>De zin zal nooit automatisch naar de volgende lijn teruglopen.<br><b>normal</b> : standaardinstelling met automatische terugloop. |
| <b>text-indent</b>    | De <b>eerste regel</b> van een tekstblok krijgt een insprong a.d.h.v. de waarde die hier ingesteld wordt  |
| <b>word-spacing</b>   | Stelt de afstand in pixels (px) of em (em). in tussen woorden van een tekstblok.  |
| <b>letter-spacing</b> | Stelt de afstand in pixels (px) of em (em) in tussen letters van een woord.   |

## 5.6.4 KLEUREN

Elementen kunnen duidelijker of mooier tot hun recht komen op een webpagina door er kleur op toe te passen. We moeten een onderscheid maken tussen de **kleur van de inhoud** van een element (*bvb. de tekst in een p element*) en de **achtergrondkleur** van het element.

| CSS Property            | Beschrijving   |
|-------------------------|--|
| <b>color</b>            | Specificeert de kleur van de tekst zelf, volgens één van de drie notaties.   |
| <b>background-color</b> | Specificeert de achtergrondkleur van het element, volgens één van de drie notaties.<br>We kunnen ook <b>transparent</b> gebruiken als waarde voor de background-color. |
| <b>border-color</b>     | Specificeert een kleur indien er een rand ingesteld is voor het element  |

Kleuren kunnen ingesteld worden volgens drie notaties: voorgedefinieerde kleurnamen, de hexadecimale waarde, of de RGB waarde ervan. Een kleur op het scherm wordt steeds gemaakt door verschillende niveau's van de basiskleuren rood, groen en blauw. Bijvoorbeeld: naarmate we minder groen en intenser rood en blauw hebben krijgen we een magenta kleur.

Elke basiskleur heeft een mogelijke waarde van 0 tot 255, wat 256 mogelijke waarden oplevert. Hierdoor kunnen we met CSS 16 miljoen ( $256 \times 256 \times 256$ ) verschillende kleuren maken. Een vaak voorkomende notatie van een kleur is hexadecimaal, (0 = 00, 255 = FF) waarbij 6 hexadecimale tekens genoteerd worden, voorafgegaan door een spoorwegteken (#).

#rrggbb  


- └ blauw component (00-ff)
- └ groen component (00-ff)
- └ rood component (00-ff)

rgb(r,g,b)  


- └ blauw component (0-255)
- └ groen component (0-255)
- └ rood component (0-255)

Met **rgb()** kunnen we de decimale waarden opgeven van elke basiskleur (0 – 255) of zelfs een percentage (0% - 100%).





# Afbeeldingen

Web Frontend Basics

## INHOUD

|   |           |
|---|-----------|
| <b>6 WERKEN MET AFBEELDINGEN</b>              | <b>3</b>  |
| <b>6.1 Figuren voor het Web</b>               | <b>3</b>  |
| 6.1.1 Formaat                                 | 3         |
| 6.1.2 Kleur                                   | 3         |
| 6.1.3 Afmetingen en resolutie                 | 4         |
| 6.1.4 Snelheid                                | 4         |
| 6.1.5 Transparantie en animatie               | 4         |
| <b>6.2 Afbeeldingen vinden en/of bewerken</b> | <b>5</b>  |
| <b>6.3 Het Image element</b>                  | <b>6</b>  |
| 6.3.1 Figuren uitlijnen                       | 6         |
| 1.1.1.1 alignen                               | 7         |
| 1.1.1.2 Floaten                               | 7         |
| <b>6.4 Figuren als Hyperlinks</b>             | <b>8</b>  |
| <b>6.5 Achtergrondfiguren</b>                 | <b>9</b>  |
| <b>6.6 Een Favicon toevoegen</b>              | <b>10</b> |

## 6 WERKEN MET AFBEELDINGEN

### 6.1 FIGUREN VOOR HET WEB

Figuren maken voor het web verschilt van figuren maken die afgedrukt moeten worden. Hoewel de karakteristieken dezelfde blijven, moeten we met meer zaken rekening houden; de figuur moet in elke browser weergegeven kunnen worden en ze mag ook niet te groot zijn.

Er zijn zes factoren waarmee we rekening moeten houden: formaat, kleur, resolutie, snelheid, transparantie en animatie.

#### 6.1.1 FORMAAT

Wanneer je figuren maakt die bedoeld zijn voor het web is het belangrijk om te weten of de figuur eigenlijk wel weergegeven kan worden voor de bezoekers. Dagelijks bevinden er zich miljoenen Windows PC's, Macs, Linux, en andere computers op het web en elke systeem heeft zijn eigen grilletjes. De taak als webdesigner is (*alweer*) om de grootste gemene deler te zoeken zodat de figuren steeds op dezelfde manier weergegeven kunnen worden.

De meest ondersteunde formaten zijn:

- **GIF** (.gif / Graphics Interchange Format)
  - Gebruikt voor figuren die **weinig kleuren** hebben, bijvoorbeeld logo's of computer getekende figuren.
  - Transparantie mogelijk.
  - Animatie mogelijk.
- **JPEG** (.jpg of .jpeg / Joint Photographic Experts Group)
  - Gebruikt voor figuren die **veel kleuren** hebben, bijvoorbeeld foto's.
  - Transparantie niet mogelijk
  - Animaties niet mogelijk
- **PNG** (.png / Portable Network Graphics)
  - Patentvrij formaat ter vervanging van GIF.
  - Meestal betere compressie dan GIF met dezelfde kwaliteit (kleiner)
  - Zeer precieze transparantie mogelijk (24-bit PNGs)
  - Animaties niet mogelijk (wel met de zgn. "APNG" die weinig ondersteund is)

#### 6.1.2 KLEUR

Tegenwoordig zijn 8-bit monitors nagenoeg uitgestorven en zijn we redelijk zeker dat de monitor van onze gebruikers de meeste kleuren op dezelfde manier weergeeft. Het zogenaamde Browser-Safe kleurpalet is zo langzamerhand in onbruik geraakt (*hoewel er nog redenen zijn om deze te gebruiken*).

Voor figuren kunnen we stellen dat we **GIF** of **PNG** gebruiken voor figuren met 256 kleuren of minder. Voor figuren met meer kleuren, zoals een foto, gebruiken we het **JPEG** formaat voor maximale compressie en kwaliteitsbehoud.

### 6.1.3 AFMETINGEN EN RESOLUTIE

Digitale figuren worden gemeten in **pixels**. Dit zegt in principe nog niets over hoe groot deze overkomt voor de lezer omdat dit afhankelijk is van de gebruikte resolutie. Wanneer we een foto van 1600 x 1200 (breedte x hoogte) afdrukken met een printer die een **resolutie** van 200ppi (pixels per inch) dan meten we 8 bij 6 inch ( $1600/200=8$  en  $1200/200=6$ ) of 20,32 cm bij 15,24 cm. Monitors hebben meestal een lagere resolutie, rond de 86ppi (ergens tussen 72 tot 100ppi) en zullen de figuur veel te groot weergeven (18 x 14 inch of 46 x 36cm) om ze op het scherm te laten passen.

Vroeger hadden de meeste monitors een resolutie van 640px bij 480px. Tegenwoordig is dit vaak Full HD (1920 x 1080) of zelfs meer. Het is dus belangrijk om er voor te zorgen dat de tekst en/of figuren niet verdwijnen aan de rechterkant van de pagina en rekening te houden met de resolutions die we tegenwoordig mogen verwachten van de gemiddelde monitor. Hou ook rekening met mobiele apparaten!

### 6.1.4 SNELHEID

Een krant zou weinig succes hebben indien lezers zouden moeten wachten op de figuren die erbij horen. In het geval van webpagina's is er steeds een wachttijd om alle data over te brengen, en van al deze data gebruiken de figuren meestal de meeste bandbreedte. Het is dus immens belangrijk om figuren zo klein mogelijk te houden met minimaal kwaliteitsverlies.

De gemakkelijkste manier om een figuur zo klein mogelijk te houden qua grootte is deze te herschalen. We zullen in elk geval onze foto van 1600 x 1200 moeten herschalen naar een werkbaar formaat, bijvoorbeeld 533 x 400, wat al veel minder bandbreedte zou verbruiken. De figuur kan tevens volledig weergegeven worden in een browserscherm zonder dat we moeten scrollen.

Via een verhoogde compressie kunnen we de bestandsgrootte van de figuur verkleinen met behoud van de afmetingen. Compressie is sterk afhankelijk van het formaat: GIF en PNG doen dit beter voor illustraties en figuren met weinig kleuren, JPEG compressie is specifiek ontwikkeld voor kleurrijke foto's. De meeste programma's voor figuurbewerking (GIMP, Photoshop, ...) hebben opties om de compressie in te stellen voor een optimale grootte/kwaliteitsverhouding.

### 6.1.5 TRANSPARANTIE EN ANIMATIE

Alle figuren die we weergeven zijn rechthoekig in afmeting. Met transparante figuren kunnen we de layout van onze webpagina echter complexer maken (ronde of grillige randen, de ene figuur boven een andere plaatsen, enz.). Onze pagina kan hierdoor veel aantrekkelijker worden. Enkel GIF en PNG ondersteunen transparantie.

Geanimeerde figuren worden **best niet in overmaat gebruikt**. Het kan werkelijk irritant zijn voor de bezoeker wanneer bijna elke afbeelding beweegt of de aandacht probeert te trekken. Meer gebruikt zijn de "loading" animaties die we meer en meer tegenkomen op sites die javascript of AJAX gebruiken om inhoud in te laden. Figuuranimatie voor webpagina's is enkel mogelijk met het GIF formaat.

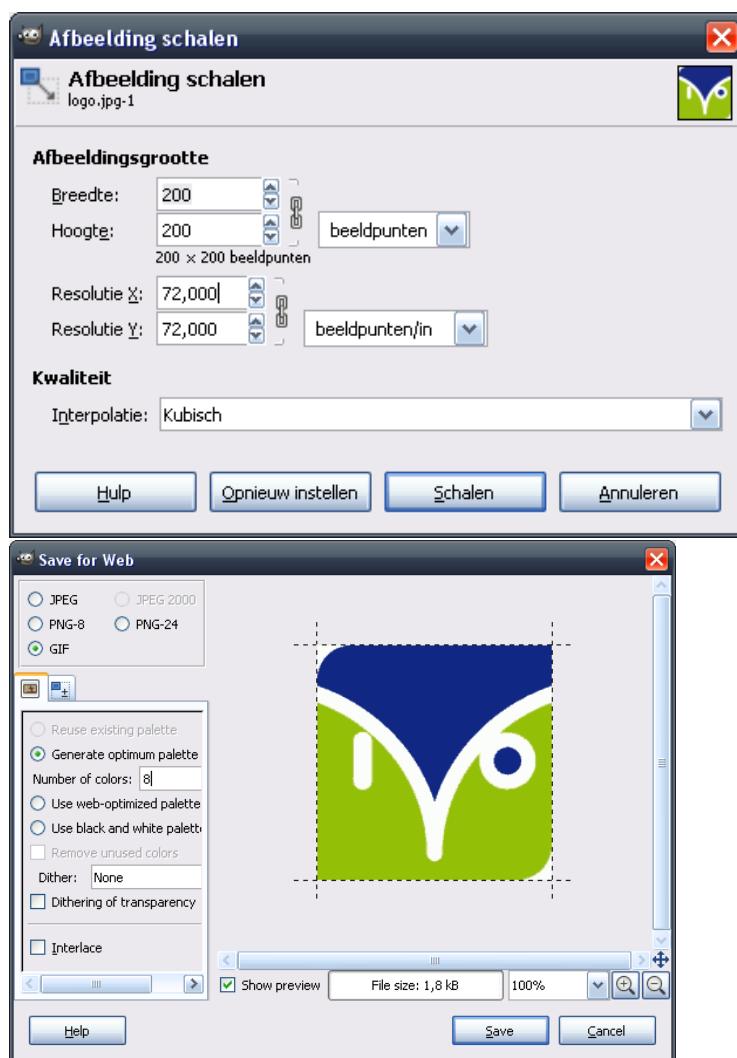
## 6.2 AFBEELDINGEN VINDEN EN/OF BEWERKEN

Creatievelingen kunnen zelf hun afbeeldingen en icoontjes die afkomstig zijn van een scanner, digitale camera, illustraties met software of pen, enz... maken en bewerken. Wanneer een nieuwe site ontworpen wordt, zal deze meestal getekend worden met een programma (*Photoshop, GIMP, Illustrator,...*) en vervolgens in verschillende afbeeldingen gesneden voor gebruik in de HTML code.

We kunnen ook veel figuren vinden op het web:

- Google (*opgelet voor copyrights voor die figuren!*)
- Gratis figuren downloaden van bepaalde sites (*zgn. "royalty free"*)
- Kwaliteit "stock" illustraties/foto's kopen en downloaden (*vanaf \$1 per foto*)

Meestal zullen we deze figuren verder willen bewerken voordat we ze gebruiken, of tenminste de grootte aanpassen zodat ze snel gedownload worden naar de browser. Er bestaan talloze programma's om figuren te bewerken en op te slaan in een zuinig formaat voor webpagina's. GIMP is een gratis alternatief voor Photoshop met gelijkaardige functionaliteiten.



GIMP Opties voor herschalen van het IVO logo en een "Save for Web" plugin met preview.

Wanneer je zelf figuren bewerkt, is het belangrijk om vertrouwd te raken met je gekozen programma. De meeste bevatten basisopties om de compressie in te stellen en om de figuur te herschalen.

## 6.3 HET IMAGE ELEMENT

---

We zijn het **img** element reeds tegengekomen in de loop van de cursus. Dit hoofdstuk besteedt hieraan bijzondere aandacht en beschrijft alle mogelijkheden ervan.

```

```

Voorbeeld van een img element met passend alt attribuut

Het **src** attribuut bevat de locatie van de figuur (*het pad*). Wanneer de figuur in een map op onze eigen website staat, gebruiken we best een **relatieve** verwijzing, bij een externe figuur kunnen we enkel **absolute** verwijzingen gebruiken (zie hoofdstuk over *relatieve en absolute verwijzing*).

Het gebruik van het **alt** attribuut is **verplicht** in HTML waarin we een beschrijvend stukje tekst plaatsen over de figuur. Dit is noodzakelijk indien de figuur niet beschikbaar is in de browser of voor personen (*bvb slechtzienden*) die speciale tekst naar audio apparatuur gebruiken om met de PC te werken.

Een img element is steeds **zelfsluitend**.

### 6.3.1 FIGUREN UITLIJNEN

Image elementen zijn **inline** elementen. Alle tekst die er voor of na geplaatst wordt zal op dezelfde lijn weergegeven worden, maar we kunnen dit manipuleren met behulp van CSS.

### 1.1.1.1 ALIGNEN

```
<p>
    Een afbeelding is steeds


    <strong>inline</strong> met onze tekst
</p>
<p>
    De tekst


    is top-gealinieerd
</p>
<p>
    De tekst


    is bottom-gealinieerd
</p>
<p>
    Onze tekst is middle-gealinieerd


</p>
```

HTML syntax voorbeeld voor uitlijnen

```
/*Element aan de bovenzijde alignen*/
.topAlign {
    vertical-align:top;
}

/*Element aan de onderzijde alignen*/
.bottomAlign {
    vertical-align:bottom;
}

/*Element aan in het midden alignen*/
.middleAlign{
    vertical-align:middle;
}
```

CSS syntax voorbeeld voor uitlijnen

### 1.1.1.2 FLOATEN

Een andere manier om een afbeelding te positioneren naast je tekst zelf is door gebruik te maken van het floaten.

Float is een CSS property die je op elke welk element kan toepassen. Het vaakst komt deze property voor in toepassingen met afbeeldingen. Float heeft als mogelijke waarden **left**, **right** en

**none** (*de default waarde*). Door gebruik te maken van deze property ‘drijft’ het element als het ware aan de zijde die je specificeert.

`float: none;`

CSS syntax voor `float none`

Float  
me

As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

MDN Voorbeeld van de `float` property met value `none`

`float: left;`

CSS syntax voor `float left`

Float  
me

As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

MDN Voorbeeld van de `float` property met value `left`

`float: right;`

CSS syntax voor `float right`

Float  
me

As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

MDN Voorbeeld van de `float` property met value `right`

We kunnen bepalen waar de tekst stopt met langs de zijkant van de figuur te lopen (*wrappen*) door gebruik te maken van de css property **clear** op eenzelfde manier zoals we **float** aanroepen. Clear heeft als mogelijke waarden **none**, **left**, **right** en **both**.

## 6.4 FIGUREN ALS HYPERLINKS

In grafische omgevingen zijn we het gewoon om op figuren te klikken om bepaalde acties uit te voeren. In HTML kunnen we een figuur zonder probleem ook interactief maken door deze bijvoorbeeld in een hyperlink (*a element*) te plaatsen. Op deze manier creëren we een navigatie “knop” zodat de gebruiker naar de betreffende URL of pagina wordt doorverwezen (*zie hoofdstuk Hyperlinks*).

```
<p>U bekijkt de Da Vinci pagina <em>2</em> van 3</p>
<a href="pagina1van3.html" title="Naar vorige pagina (1/3)">

</a>
<a href="pagina3van3.html" title="Naar volgende pagina (3/3)">

</a>
```

Voorbeeld van het omsluiten van een `img` element met een `a` element om een hyperlink te maken

De figuur gedraagt zich nu als een hyperlink. Via het **title** attribuut van de hyperlink kunnen we een tooltip instellen om verwarring te vermijden. Vergeet echter het **alt** attribuut van je figuur niet!

Elke figuur die zich in een **a** element bevindt, wordt standaard voorzien van een border. We kunnen de pixelbreedte van deze rand instellen met behulp van het CSS indien de border ongewenst is.

```
border:none;
```

CSS syntax voor het verwijderen van de default border

## 6.5 ACHTERGRONDFIGUREN

---

In de plaats van een element enkel een achtergrondkleur te geven, kunnen we ook figuren instellen als achtergrond. Dit stelt ons in staat om mooie ontwerpen te maken voor webpagina's zonder het gebruik van een **img** element (*voorgrond-figuur!*).

Deze achtergrondfiguren kunnen herhaald worden langs de x of y as, uitgelijnd volgens verschillende instellingen. Een achtergrondfiguur is als een laag boven de originele achtergrondkleur (*indien deze is ingesteld*).

| CSS Property                 | Beschrijving  |
|------------------------------|---|
| <b>background-attachment</b> | Bepaalt of de achtergrondfiguur mee volgt wanneer er gescreld wordt ( <i>we kunnen soms scrollen wanneer de inhoud van een bloklement te lang is, bij de body is dit automatisch zo</i> )   |
| <b>background-image</b>      | Specificeert de url van de weer te geven figuur. We doen dit met behulp van de css notatie:<br>een online figuur. Bvb.: <b>url(<a href="http://www.mijn.figuur.com/figuur.jpg">http://www.mijn.figuur.com/figuur.jpg</a>);</b><br>een figuur die lokaal opgeslagen is in een map images bvb:<br><b>url(img/mijnfiguur.jpg);</b> |
| <b>background-position</b>   | Positioneert de figuur links, rechts, midden, boven, onder...<br>We kunnen ook X Y Coordinaten opgeven in pixels of %.  |
| <b>background-repeat</b>     | Specificeert of de figuur herhaald zal worden langs de X of Y as, beide assen of geen enkele.   |

## 6.6 EEN FAVICON TOEVOEGEN

---

Op vele sites vinden we icoontjes in de adresbalk, en wanneer we een site toevoegen aan de favorieten krijgt de snelkoppeling hetzelfde icoontje. Deze icoontjes zijn steeds 16 x 16 figuren, meestal van het type **.ico** (sommige browsers ondersteunen ook **.gif**). In de nieuwste browsers vinden we ook vaak **tabbladen**. Het is belangrijk om het icoon een transparante achtergrond te geven voor maximale compatibiliteit met de vensterkleur van de browser.

Om een favoriet icoon toe te voegen moeten we een **link** element toevoegen in de **head** sectie van onze HTML pagina.

```
<head>
...
<link rel="icon" href="img/html5.ico" type="image/x-icon" />
<link rel="shortcut icon" href="img/html5.ico" type="image/x-icon" />
<link rel="icon" type="image/png" href="img/html5_16.png" sizes="16x16">
<link rel="icon" type="image/png" href="img/html5_32.png" sizes="32x32">
<link rel="icon" type="image/png" href="img/html5_96.png" sizes="96x96">
...
</head>
```

Voorbeelden van de mogelijkheden om een favicon toe te voegen

We gebruiken het **rel** attribuut om mee te delen dat het hier over het website icoon gaat. Gebruik beide bovenstaande notaties om compatibiliteit tussen browsers te garanderen. Het **href** attribuut is het pad naar het eigenlijke icoon. **Type** is in dit geval van het type **image/x-icon**. Het is belangrijk het juiste formaat mee te delen. Indien je een gif gebruik als icoon dan is dit: **image/gif**





# Het list element

Web Frontend Basics

## INHOUD

|  |           |
|--|-----------|
| <b>7 HET LIST ELEMENT</b>                        | <b>3</b>  |
| 7.1.1 Elementen                                  | 3         |
| 7.1.2 Attributes                                 | 4         |
| 7.1.2.1 reversed                                 | 4         |
| 7.1.2.2 start                                    | 4         |
| 7.1.2.3 type                                     | 5         |
| 7.1.2.4 value                                    | 5         |
| 7.1.3 lijsten nesten                             | 6         |
| 7.1.4 CSS properties                             | 7         |
| 7.1.4.1 list-style                               | 7         |
| 7.1.4.2 list-style-type                          | 7         |
| 7.1.4.3 list-style-image                         | 7         |
| 7.1.4.4 list-style-position                      | 8         |
| <b>7.2 Van Unordered List naar Navigatiemenu</b> | <b>10</b> |
| 7.2.1 Van start gaan met de lijst                | 10        |
| 7.2.2 Aanpassingen in CSS                        | 11        |

## 7 HET LIST ELEMENT

### 7.1.1 ELEMENTEN

Lijsten, zoals opsommingen of genummerde items, zijn verwant met tabellen in die zin dat ze ook meestal dienen om data op een gestructureerde manier weer te geven. Lijsten zijn, gelukkig, veel minder complex om te schrijven dan volledige tabellen aangezien ze slechts 1 kolom hebben. We hoeven dus enkel de data van elke rij te definiëren.

We kunnen ofwel geordende lijsten definiëren (*met nummers of letters per item*) of niet-geordende lijsten met symbolen per item (*opsommingstekens*). De items in een lijst zelf definiëren we met behulp van het **li** element.

Geordende lijsten plaatsen we tussen **ol** tags (*ordered list*) en niet-geordende lijsten plaatsen we tussen **ul** tags (*unordered list*).

We bekijken eerst even onze geordende lijst.

```
<h2>Lijsten: stappenplan</h2>
<ol>
    <li>Open de lijst met een <strong>ol</strong> tag.</li>
    <li>
        Plaats elk item tussen een <strong>li</strong> tag,
        je mag hier uiteraard ook andere html in plaatsen.
    </li>
    <li>Wanneer je klaar bent met alle items, sluit de <strong>ol</strong> tag.</li>
    <li>Definieer nu opmaak voor de <strong>ol</strong> of
        <strong>li</strong> elementen.
    </li>
</ol>
```

Voorbeeld van een ordered list met 4 list items

## Lijsten: stappenplan

1. Open de lijst met een **ol** tag.
2. Plaats elk item tussen een **li** tag, je mag hier uiteraard ook andere html in plaatsen.
3. Wanneer je klaar bent met alle items, sluit de **ol** tag.
4. Definieer nu opmaak voor de **ol** of **li** elementen.

Voorbeeld output in een browser

Een niet-geordende lijst kunnen we maken met behulp van het **ul** element.

```
<h2>Boodschappenlijstje</h2>
<ul>
  <li>Melk</li>
  <li>Paprika's</li>
  <li>Bloem</li>
  <li>Vers gehakt</li>
  <li>Eieren</li>
</ul>
```

Voorbeeld van een unordered list met 5 list items

## Boodschappenlijstje

- Melk
- Paprika's
- Bloem
- Vers gehakt
- Eieren

Voorbeeld output in een browser

### 7.1.2 ATTRIBUTES

Voor het **ol** element kunnen we gebruik maken van volgende attributes. Hou er rekening mee dat deze attributes **niet** geldig zijn voor gebruik op het **ul** element.

#### 7.1.2.1 REVERSED

Door toevoeging van dit attribute zal de volgorde van de nummering van hoog naar laag zal gebeuren overeen alle **li** elementen binnen het **ol** element.

```
<ol reversed>
```

#### 7.1.2.2 START

Met dit attribute kun je via een value (*type integer*) het startpunt van de nummering te specifiëren.

```
<ol start="4">
```

**Let op!** Zelfs bij de keuze van een ander type van nummering zal de gebruikte value van het start attribute steeds een integer getal blijven. Je zal dan enkel een waarde moeten kiezen in relatie met het gekozen type.

### 7.1.2.3 TYPE

Door het attribute type in combinatie met een van onderstaande values te gebruiken, kan je het opsommings type van de nummering bepalen.

- 'a' duidt op het gebruik van kleine letters (*lowercase*)
- 'A' duidt op het gebruik van hoofdletters (*uppercase*)
- 'i' duidt op het gebruik van kleine Romeinse cijfers (*lowercase*)
- 'I' duidt op het gebruik van Romeinse cijfers met hoofdletters (*uppercase*)
- '1' duidt op het gebruik van cijfers, welke de standaardwaarde is (*default*)

Het gekozen type zal doorheen alle **li** elementen binnen het **ol** element gebruikt worden, tenzij er een ander type op een **li** element zelf gedefinieerd wordt.

```
<ol type="A">
```

Ook het **li** element laat het gebruik van een attribute toe. Dit is in HTML5 echter beperkt tot 1 attribute.

### 7.1.2.4 VALUE

Via het attribute value kunnen we een waarde (*type integer*) toe kennen aan een li element. De nummering van de lijst wordt dan verdergezet vanaf deze waarde.

```
<li value="3">
```

**Let op!** Zelfs bij de keuze van een ander type van nummering zal de gebruikte value van het start attribute steeds een integer getal blijven. Je zal dan enkel een waarde moeten kiezen in relatie met het gekozen type.

```
<h3>Top 20 Programmeertalen 2019</h3>
<ol start="20" reversed>
  <li>R</li>
  <li>Perl</li>
  <li>Swift</li>
  <li value="4">C#</li>
  <li>Python</li>
  <li>C</li>
  <li>Java</li>
</ol>
```

Voorbeeld van een ordered list, omgekeerde volgorde, custom startwaarde en override van een list item positie

## Top 20 Programmeertalen 2019

20. R
19. Perl
18. Swift
4. C#
3. Python
2. C
1. Java

### 7.1.3 LIJSTEN NESTEN

We kunnen bij lijsten ook perfect de verschillende elementen die we net zagen in combinatie met elkaar gebruiken.

Met andere woorden: de **ol** en **ul** elementen kunnen genest worden. Dit kan door hetzelfde element nog eens te gebruiken binnen een **li** element.

```
<h2>Geneste lijsten</h2>
<ol>
  <li>Ons eerste puntje</li>
  <li>Tweede punt
    <!-- zonder het <li> element te sluiten! -->
    <ul>
      <li>Extra item onder punt twee</li>
      <li>Nog een extra item onder punt twee</li>
      <li>Een laatste extra item onder punt twee</li>
    </ul>
  </li>
  <!-- Hier pas zullen we ons <li> element sluiten -->
  <li>Derde punt in de ordered list</li>
</ol>
```

*Voorbeeld van het maken van geneste lijsten*

## Geneste lijsten

1. Ons eerste puntje
2. Tweede punt
  - Extra item onder punt twee
  - Nog een extra item onder punt twee
  - Een laatste extra item onder punt twee
3. Derde punt in de ordered list

*Voorbeeld output in een browser*

## 7.1.4 CSS PROPERTIES

Voor de opmaak van onze lijsten kunnen we uiteraard ook CSS-properties gebruiken. Zo kunnen we bvb. het opsommingsteken bepalen door middel van CSS.

Onderstaand overlopen we even enkele specifieke CSS-properties voor onze lijsten.

### 7.1.4.1 LIST-STYLE

Met behulp van de property `list-style` kunnen we volgende zaken beheren op onze list item `<li>` elementen;

### 7.1.4.2 LIST-STYLE-TYPE

Standaard heeft een `ol` element een cijfer als opsommingsteken en een `ul` heeft een bolletje, maar we kunnen dit beïnvloeden met de CSS-eigenschap **list-style-type**. Die laat toe een groot aantal opsommingstekens in te stellen zoals letters, Romeinse cijfers en nog veel meer. Het **list-style-type** kan op een aantal manieren ingesteld worden. We bekijken deze even in onderstaande code voorbeelden.

```
/* Door middel van gebruik te maken van een van de gedefinieerde types in CSS */
list-style-type: circle;
list-style-type: disc;
list-style-type: square;
```

*Voorbeeld van enkele mogelijkheden in de list-style-type property*



#### Mogelijkheden

Mogelijke values van de types kan je onder andere terugvinden op het [Mozilla Developer Network](#).

### 7.1.4.3 LIST-STYLE-IMAGE

We kunnen ook een figuur instellen i.p.v. van de standaard symbolen. In dat geval geven we de URL naar het figuurbestand, vertrekend vanuit de locatie van de CSS.

```
/* Door zelf een verwijzing te maken naar de te gebruiken afbeelding */
list-style-image: url('../img/list_bullet_gen.gif');
```

#### 7.1.4.4 LIST-STYLE-POSITION

Uiteraard kunnen we ook de positie van de opsommingstekens bepalen. Hier hebben we slechts twee opties: **inside** of **outside**. Dit zal de opsommingstekens ofwel binnen ofwel buiten de *content* van ons *box-model* plaatsen.

```
/* De positie van onze opsommingstekens bepalen */
list-style-position: inside;
list-style-position: outside;
```

Zoals we reeds zagen bij andere CSS properties kan alles ook hier terug in 1 regel geschreven worden.

```
/* We kunnen zoals in onderstaand voorbeeld ook bij het gebruik van een afbeelding nog steeds ook een waarde voor type meegeven. Dit zal dan als back-up gebruikt worden indien de afbeelding niet gevonden wordt. */
list-style: square inside url('../images/list_bullet_gen.gif');
```

We bekijken even een paar voorbeelden van de toepassing van de list-style property.

```
<h2>eerste lijst</h2>
<ul class="een">
  <li>Melk</li>
  <li>Paprika's</li>
  <li>Bloem</li>
</ul>
<h2>tweede lijst</h2>
<ul class="twee">
  <li>Hout</li>
  <li>Lucifers</li>
  <li>Marshmallows</li>
>
</ul>
<h2>derde lijst</h2>
<ul class="drie">
  <li>Hout</li>
  <li>Lucifers</li>
  <li>Marshmallows</li>
>
</ul>
```

HTML syntax

```
.een {
  list-style-type: square;
}

.twee {
  list-style-image:
    url('../images/bonfire.png');
  list-style-position: inside;
}

.drie {
  list-
style: square outside url('../images/bonfire.png'
);
}
```

CSS syntax

### eerste lijst

- Melk
- Paprika's
- Bloem

### tweede lijst

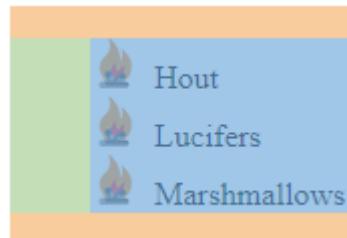
-  Hout
-  Lucifer
-  Marshmallows

### derde lijst

-  Hout
-  Lucifer
-  Marshmallows

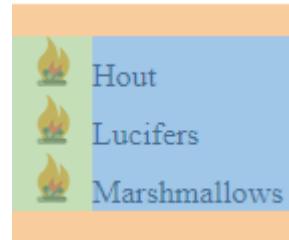
Als we bovenstaande voorbeeldcode bekijken in een browser dan zullen we het resultaat zien zoals hiernaast wordt weergegeven.

We gaan nog even iets dieper in op de **list-style-position** property om aan te tonen dat onze opsommingstekens effectief samen met onze content geplaatst worden (*inside*) of anderzijds er net buiten geplaatst worden (*outside*).



Als we via de inspector (bij de meeste browsers beschikbaar via F12) kijken naar het box model van onze tweede lijst, dan krijgen we het model te zien zoals hiernaast. De opsommingstekens worden binnen de content

geplaatst.



Bekijken we de derde lijst op dezelfde manier, dan zien we duidelijk dat de opsommingstekens zich net buiten de

content in het box-model bevinden. Meer bepaald in de padding van het box-model in kwestie.

## 7.2 VAN UNORDERED LIST NAAR NAVIGATIEMENU

---

Lijsten worden veelal gebruikt om een opsomming weer te geven, maar ze zijn ook handig om om te toveren naar een navigatie menu. Op het internet kan je talrijke voorbeelden terugvinden die je doorheen de nodige stappen zullen looden. In de cursus staan wij alvast even stil bij een mogelijke benadering.

### 7.2.1 VAN START GAAN MET DE LIJST

We hebben uiteraard een lijst nodig als basis in onze HTML-code. We beginnen met de volgende code in ons **body** element op de HTML-pagina te plaatsen. We voorzien meteen ook de basis CSS-code om onze pagina te ontdoen van *margin* en *padding* en deze een achtergrondkleur te geven.

```
<!-- Semantisch header element -->
<header>
    <!-- Semantisch nav element -->
    <nav>
        <!-- Onze lijst als basis voor onze navigatie -->
        <ul>
            <li><a href="#">Start</a></li>
            <li><a href="#">Cursussen</a></li>
            <li><a href="#">Oefeningen</a></li>
            <li><a href="#">Dropbox</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
</header>
```

HTML syntax

```
body {
    margin: 0;
    padding: 0;
    background-color: lightgray;
}
```

CSS syntax

Het viel je waarschijnlijk al op dat we onze lijst in een **nav** element plaatsten en dat **nav** element binnen een **header** element. Het is steeds een goede oefening om de semantische elementen waar mogelijk toe te passen en hiermee vertrouwd te raken. De inhoud van onze **li** elementen voorzien we reeds van een **a** element. Aan het href attribute kennen we voorlopig een value toe van een lege id (*namelijk #*). Op deze manier zien we reeds onze links staan, maar deze zullen nog nergens heen leiden, enkel terug naar de start van onze pagina.

## 7.2.2 AANPASSINGEN IN CSS

We zijn klaar voor onze volgende stap, waar we ons lijst element zullen ontdoen van zijn list-style. Door een andere achtergrond kleur onderscheiden we de navigatiebalk visueel van de pagina.

```
nav ul {  
    list-style: none;  
    background-color: darkslategrey;  
    text-align: center;  
    padding: 0;  
    margin: 0;  
}
```

Toevoegen van een selectie op een unordered element dat zich in een nav element bevindt



The screenshot shows a dark grey navigation bar with five blue underlined links: "Start", "Cursussen", "Oefeningen", "Dropbox", and "Contact". The links are centered and have a uniform appearance due to the CSS styles applied to the parent **ul** element.

Start  
Cursussen  
Oefeningen  
Dropbox  
Contact

Output in een browser

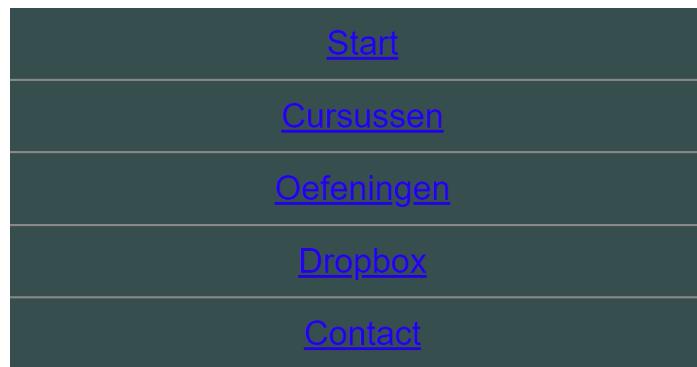
Door onze list-style property een value te geven van none zullen de gebruikte opsommingstekens verwijderd worden. De inhoud per **li** element zal vervolgens gecentreerd worden over de volledige breedte van het **ul** element. Een klein stapje, maar we zijn er natuurlijk nog niet.

Je merkte al op dat we onze CSS selector toepassen op alle **ul** elementen die zich binnen een **nav** element bevinden. Dit is een goede manier om ervoor te zorgen als er later ergens op je pagina's gebruik wordt gemaakt van **ul** elementen dat hun stijl los staat van de **ul** elementen die gebruikt worden voor de navigatie.

We gaan verder met onze navigatiebalk.

```
nav li {  
    font-family: sans-serif;  
    font-size: 1.2em;  
    line-height: 40px;  
    height: 40px;  
    border-  
    bottom: 1px solid #888;  
}
```

Een selector toepassen op list item elementen die zich in een nav element bevinden



Output in een browser

Onze **li** elementen krijgen al een klein beetje meer stijl en worden door gebruik te maken van een relatieve eenheidsmaat (*dit zien we nog in een later hoofdstuk*) als value van de font-size property een grootte die aangepast is aan de schermgrootte van het gebruikte apparaat.

Laten we eerst even werk maken van de stijl van onze tekst, zodat onze ogen al wat minder pijn gaan doen.

Door onderstaande CSS maken we de tekst binnen onze **a** elementen die zich in een **nav** element bevinden wit van kleur. De lijn onder deze tekst verwijderen we door gebruik te maken van de text-decoration property. We passen meteen ook de kleur aan van de achtergrond van het **a** element als we er met onze cursor overheen gaan (*:hover*)

```
nav a {  
    text-decoration: none;  
    color: White;  
}  
  
nav a:hover {  
    background-color: #005f5f;  
}
```

Toevoeging van CSS voor de navigatie



Output in een browser

Dit begint nu al iets meer op een degelijke navigatie te lijken. Maar om aan het concept van een navigatiebalk te voldoen, is het natuurlijk aan te raden om onze navigatie opties mooi op een horizontale lijn naast elkaar krijgen.

We maken hiervoor gebruik van de *display* property op onze **li** elementen. Van nature uit hebben **li** elementen de *display* value van *list-item* wat als standaard gedrag een block als display type zal aannemen. We maken een toevoeging aan onze reeds eerder aangemaakte selector voor **li** elementen in een **nav** element. We geven hier ook een vaste breedte mee per item in onze navigatie.

```
nav li {  
    ...  
    display: inline-block;  
    width: 120px;  
}
```

Toevoeging van CSS aan de reeds bestaande `nav li` selector

Start      Cursussen      Oefeningen      Dropbox      Contact

Output in een browser

Dit begint nu toch al iets meer op een navigatiemenu te lijken! Maar als we over onze menu items heen gaan zien we toch nog iets vervelends. De ruimte waar we met onze cursor kunnen klikken is beperkt tot de tekst die we als inhoud van ons `a` element geplaatst hebben. Om dit duidelijk te maken tonen we in onderstaand voorbeeld wat er geselecteerd wordt door gebruik te maken van een gele achtergrond in de `:hover` selector.

Start      Cursussen      Oefeningen      Dropbox      Contact

Als laatste stap zullen we dit probleem ook door middel van CSS van de baan helpen.

We voegen in onze selector voor de `a` elementen binnen een `nav` element onderstaande lijn toe. Deze zal ervoor zorgen dat onze `a` elementen zich over de volledige breedte en hoogte zullen weergeven.

```
nav a {  
    ...  
    display: block;  
}
```

Start      Cursussen      Oefeningen



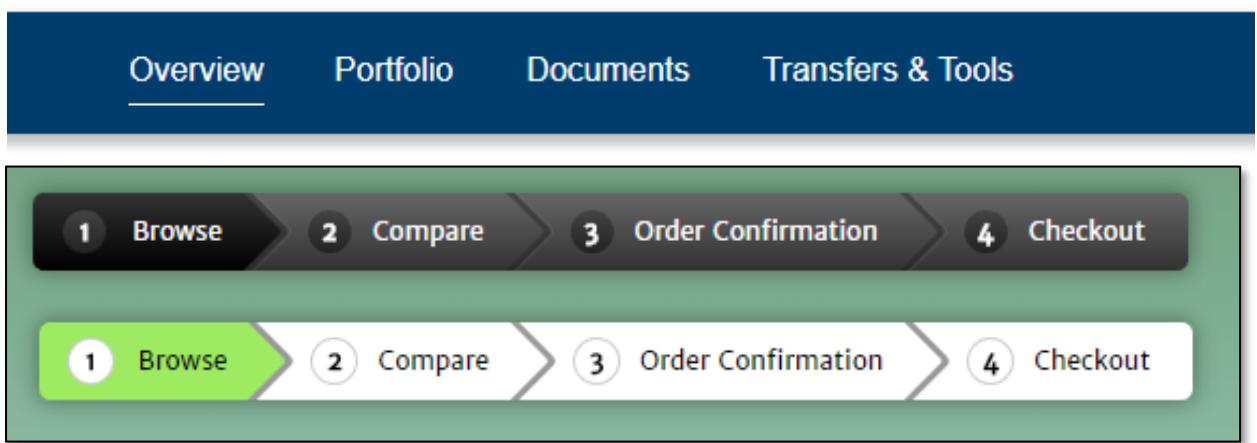
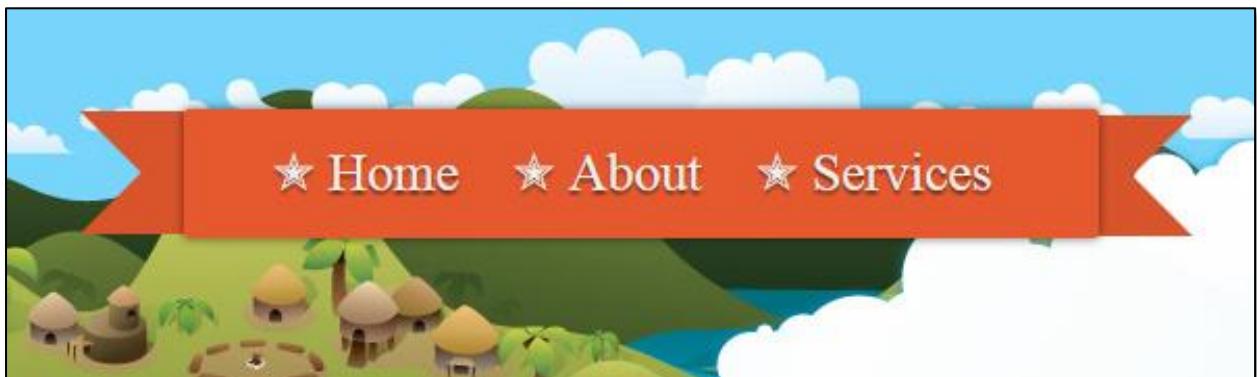
Output in een browser

Toevoeging van CSS aan de reeds bestaande `nav a` selector

Hiermee is de basis om een navigatiemenu via een lijst element te maken, afgerond. Er zijn natuurlijk nog een aantal belangrijke punten die in dit voorbeeld niet meteen aan bod kwamen. Denk bijvoorbeeld aan het responsive maken van je navigatiemenu, een andere layout voorzien voor kleinere schermen (bv.: gsm), het positioneren van andere elementen binnen je navigatiemenu in combinatie met je lijst, enz.

Nu je de basis gezien hebt, staat je natuurlijk niks in de weg om hierop verder te bouwen. Of zelfs één van de honderden andere uitwerkingen die online te vinden zijn te gaan bekijken en implementeren.

Alvast nog een paar voorbeelden ter inspiratie.







# Tabellen

Web Frontend Basics

## INHOUD

|                                |          |
|--------------------------------|----------|
| <b>8 TABELLEN IN HTML</b>      | <b>3</b> |
| <b>8.1 Eenvoudige tabellen</b> | <b>3</b> |
| 8.1.1 Tabellen definiëren      | 3        |
| 8.1.2 Tabelrijen en cellen     | 3        |
| 8.1.3 Caption                  | 4        |
| <b>8.2 Cellen groeperen</b>    | <b>4</b> |
| 8.2.1 Groeperen per rij        | 4        |
| 8.2.2 Groeperen per kolom      | 6        |
| <b>8.3 Cellen samenvoegen</b>  | <b>7</b> |

## 8 TABELLEN IN HTML

### 8.1 EENVOUDIGE TABELLEN

#### 8.1.1 TABELLEN DEFINIËREN

Met tabellen kunnen we data overzichtelijk weergeven in rijen en kolommen. In HTML moeten we dergelijke rijen en kolommen definiëren in het **table** element. Tussen de **table** tags gaan we aan de slag met **tr** en **td** elementen die de rijen en cellen van onze tabel vorm zullen geven.

```
<table>
  <tr>
    <td>Cel 1</td>
    <td>Cel 2</td>
    <td>Cel 3</td>
  </tr>
  <tr>
    <td>Cel 4</td>
    <td>Cel 5</td>
    <td>Cel 6</td>
  </tr>
</table>
```

Voorbeeld van een tabel met twee rijen en drie kolommen

De cellen (**td** elementen) horen enkel thuis in een rij (**tr** elementen). We moeten dus steeds eerst de rij definiëren en dan pas de cellen.

#### 8.1.2 TABELRIJEN EN CELLEN

Om een tabel te maken, moeten we minstens 1 rij hebben met 1 cel erin. Dit creëert een tabel met 1 kolom. Het aantal cellen per rij bepaalt het aantal kolommen dat de tabel hoort te hebben. Je moet er dus altijd voor zorgen dat het aantal cellen identiek is voor elke rij (*behalve wanneer we cellen samenvoegen, zie verder*).

Tabelrijen definiëren met we het **tr** element. Dit maakt een “onzichtbare” rij aan waarin we cellen kunnen plaatsen. Deze cellen maken we aan met **th** element of het **td** element. Het grootste verschil tussen deze twee elementen is dat een **td** element standaard zonder opmaak op de pagina staat, terwijl **th** element specifieker gemaakt is om kolomkoppen te definiëren.

Binnen **th** elementen staat de tekst standaard vetgedrukt en gecentreerde in de cel.

```

<table>
  <tr>
    <th>Cel 1</th>
    <th>Cel 2</th>
    <th>Cel 3</th>
  </tr>
  <tr>
    <td>Cel 4</td>
    <td>Cel 5</td>
    <td>Cel 6</td>
  </tr>
</table>

```

Merk op dat we steeds evenveel **td's** als **th's** definiëren als er kolommen zijn.

### 8.1.3 CAPTION

Behalve rijen en cellen bestaan er nog een aantal elementen die binnenin tabel gedefinieerd kunnen worden, waaronder de **caption**. Dit element laat ons toe een beschrijving te geven aan de tabel. Standaard staat deze tekst boven de tabel, buiten de border. Onderstaand voorbeeld illustreert een tabel met een caption:

```

<table class="myTable">
  <caption>Beschikbare Hardware</caption>
  <tr>
    <th>Id</th>
    <th>Merk</th>
    <th>Type</th>
  </tr>
  <tr>
    <td>#4039</td>
    <td>Kanon</td>
    <td>Printer/Scanner</td>
  </tr>
  <tr>
    <td>#4959</td>
    <td>Linksas</td>
    <td>Wireless Router</td>
  </tr>
</table>

```

Voorbeeld van een **table** element met **caption** element

```

table.myTable {
  border: solid 1px blue;
  background-color: #cccccc;
  width: 50%;
}

table.myTable > caption {
  text-align: left;
  font-style: italic;
  font-variant: small-caps;
}

```

CSS op basis van *element en class selector*, en direct *child selector* van het **caption** element

## 8.2 CELLEN GROEPEREN

### 8.2.1 GROEPEREN PER RIJ

Hoewel rijen duidelijk gedefinieerd staan als **tr** elementen, moeten we ze steeds individueel benaderen. Wanneer we in staat zijn om een aantal rijen te groeperen, dan kunnen we deze een duidelijker stijl geven, of afzonderen van de rest met randen. Hierin onderscheiden we 3 verschillende soorten groepen. We kunnen drie tags gebruiken die we *rond de te groeperen tr elementen plaatsen*. Deze drie elementen zijn allen optioneel:

| Element      | Beschrijving   |
|--------------|--|
| <b>thead</b> | Groepeert rijen die deel uitmaken van de tabelhoofding |
| <b>tfoot</b> | Groepeert rijen die deel uitmaken van de tabelvoet     |
| <b>tbody</b> | Groepeert rijen die deel uitmaken van een gewone groep |

Deze elementen zorgen voor nuttige semantische informatie die gebruikt kan worden bij het renderen (*scherm of printer*) en voor toegankelijkheidsdoeleinden. Aangezien deze tags enkel van toepassing zijn op tabellen mogen deze enkel voorkomen binnenin een **table** element, en moeten ze steeds een groep **tr** elementen omsluiten.



### tbody element

- Als een tabel een **thead** element bevat (*om headerrijen semantisch te identificeren*), moet het **tbody** element er net na komen.
- Als je gebruik maakt van het **tbody** element mag je in je **table** element geen rechtstreeks onderliggende **tr** element meer gebruiken.  
Bij het afdrukken van een document geven de elementen **thead** en **tfoot** informatie die op elke pagina van een tabel met meerdere pagina's dezelfde kan zijn. Dit in tegenstelling tot de inhoud van het **tbody** element die over het algemeen zal verschillen van pagina tot pagina.
- Je mag meer dan één **tbody** element per tabel gebruiken, zolang ze allemaal maar opeenvolgend zijn. Hiermee kan je grote tabellen makkelijk in secties verdelen, die elk afzonderlijk kunnen worden opgemaakt indien nodig.

```
<table>
  <thead>
    <tr>
      <th>Eigenschap</th>
      <th>IE 6</th>
      <th>IE 7</th>
      <th>IE 8</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>max-width</td>
      <td>Nee</td>
      <td>Ja</td>
      <td>Ja</td>
    </tr>
    <tr>
      <td>min-width</td>
      <td>Nee</td>
      <td>Ja</td>
      <td>Ja</td>
    </tr>
  </tbody>
  <tbody>
    <tr>
      <td>max-height</td>
      <td>Nee</td>
      <td>Ja</td>
      <td>Ja</td>
    </tr>
    <tr>
      <td>min-height</td>
```

```

        <td>Nee</td>
        <td>Ja</td>
        <td>Ja</td>
    </tr>
</tbody>
<tfoot>
    <tr>
        <td>compatibiliteit</td>
        <td>30%</td>
        <td>100%</td>
        <td>100%</td>
    </tr>
</tfoot>
</table>

```

Voorbeeld van een **table** element met bijhorende **thead**, **tbody** en **tfoot** elementen

```

table
{
    border: solid 1px #000000;
}
table thead
{
    background-color: #add8e6; /*lightblue */
}
table tfoot
{
    background-color: #ffffe0; /* lightyellow */
    font-weight: bold;
}

```

Aanvullende CSS voor de opmaak op **thead** en **tfoot** elementen

### 8.2.2 GROEPEREN PER KOLOM

Er is geen duidelijke definitie voor kolommen in onze tabellen: ze worden immers bepaald door het aantal cellen: de **td** of **th** elementen binnen de rijen. Als we een groep voor kolommen willen definiëren, gebruiken we het **colgroup** element, die onder de openingstag van **table** gedefinieerd worden (net zoals **caption** dus). Tussen de tags van een **colgroup** element plaatsen we een aantal **col** elementen, die aangeven hoeveel kolommen tot de kolomgroep behoren.

We zullen nu de voorgaande tabel opdelen in twee kolomgroepen: de eerste groep bestaat enkel uit de eerste kolom die de CSS-eigenschappen bevat, de tweede kolomgroep zijn de drie andere kolommen die de waarden bevatten.

```

<table>
    <colgroup>
        <col>
        <col span="3" class="browsers">
    </colgroup>
    <thead>
    ...

```

Toevoeging van het **colgroup** element

```
.browsers {
    background-color: #d7d9f2;
}
```

Opmaak van de laatste 3 kolommen gebruik makend van class selector

Met het **span** attribuut van het **col** element geven we aan dit col element de **drie volgende** kolommen moet vertegenwoordigen. Voegen we het **span** attribuut niet toe, dan krijgt dit standaard de waarde van 1.



### Beperkingen CSS voor colgroup

Wanneer je kolommen groepeert via **colgroup** en **col** dan kan je hiermee slechts een beperkte selectie van CSS properties toepassen:

- **border**
- **background**
- **width**
- **visibility**

Zoals in het voorbeeld hierboven kan je ook de meer specifieke versies van **border** en **background** gebruiken, zoals **background-color**, **border-width**, etc. Wil je andere properties instellen voor een volledige kolom in een tabel, dan zal je beroep moeten doen op meer geavanceerde CSS die in een later hoofdstuk besproken wordt.

## 8.3 CELLEN SAMENVOEGEN

Met groeperen kunnen we een aantal cellen of rijen als 1 logisch geheel zien. Het is echter ook mogelijk om een cel uit te breiden zodat ze met de cellen ernaast (of er onder) wordt samengevoegd. In dat geval definiëren we 1 **td** (of **th**) element met een attribuut **rowspan** of **colspan** dat aangeeft hoeveel cellen er overlapt moeten worden.

| Attribuut      | Beschrijving                                      |
|----------------|---|
| <b>colspan</b> | Het aantal kolommen dat deze cel moet overlappen. |
| <b>rowspan</b> | Het aantal rijen dat deze cel moet overlappen.    |

Wanneer we een col-of rowspan instellen op een waarde groter dan 1, dan betekent dit dat de cellen die normaal op deze plaats staan gewoon wegvalLEN. We mogen deze dus niet meer definiëren in de tabelrij! Zoals we boven reeds hebben vermeld en bekijken in de voorbeeld moet elke rij **hetzelfde**

aantal cellen bevatten. Zodra er 1 cel wordt samengevoegd met enkele andere, dan mogen deze andere cellen niet meer voorkomen.

```
<table>
  <thead>
    <tr>
      <td colspan="2">Cel 1</td>
      <td>Cel 3</td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Cel 4</td>
      <td rowspan="2">Cel 5</td>
      <td>Cel 6</td>
    </tr>
    <tr>
      <td>Cel 4b</td>
      <td>Cel 6b</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Cel 7</td>
      <td>Cel 8</td>
      <td>Cel 9</td>
    </tr>
  </tfoot>
</table>
```

Voorbeeld van het gebruik van de attributen **colspan** en **rowspan**





# Semantische elementen

Web Frontend Basics

## INHOUD

|   |          |
|---|----------|
| <b>9 SEMANTISCHE ELEMENTEN</b>              | <b>3</b> |
| <b>9.1 Semantiek</b>                        | <b>3</b> |
| 9.1.1 Wat is “semantiek”                    | 3        |
| 9.1.2 Semantiek in HTML                     | 3        |
| 9.1.3 Voordelen                             | 3        |
| <b>9.2 Een aantal Semantische elementen</b> | <b>4</b> |
| 9.2.1 Header element                        | 4        |
| 9.2.2 Nav element                           | 4        |
| 9.2.3 Main element                          | 5        |
| 9.2.4 Article element                       | 5        |
| 9.2.5 Section element                       | 7        |
| 9.2.6 Aside element                         | 7        |
| 9.2.7 Footer element                        | 8        |
| 9.2.8 Time element                          | 8        |
| 9.2.9 Figure en Caption element             | 8        |
| <b>9.3 To div or not to div?</b>            | <b>9</b> |

## 9 SEMANTISCHE ELEMENTEN

### 9.1 SEMANTIEK

#### 9.1.1 WAT IS “SEMANTIEK”

De semantiek houdt zich bezig met de betekenis van symbolen. Vergelijk dit met de woorden in een boek: we kennen werkwoorden, zelfstandige naamwoorden, enz. die de betekenis van een zin duidelijk maken. De zinnen geven betekenis aan de boodschap die de auteur duidelijk wil maken. Zinnen die bij elkaar horen worden gegroepeerd in alinea's, die op hun beurt weer hoofdstukken vormen. Losstaand van de eigenlijke inhoud van een boek is er vaak ook nog tekstmateriaal zoals een indexopgave, bibliografie, of een voorwoord.

#### 9.1.2 SEMANTIEK IN HTML

Met de komst van HTML5 is het mogelijk om de inhoud van webpagina's een gelijkaardige semantiek te geven. Dit gebeurt aan de hand van onze HTML elementen. In plaats van een weinigzeggend **div** element te gebruiken, kan je bijvoorbeeld gebruik maken van een **article** element om duidelijk te maken dat de inhoud van dit element met een artikel te maken heeft.

Een menselijke bezoeker heeft er op het eerste gezicht weinig aan, maar door het gebruik van deze semantische elementen is het voor zoekrobots en applicaties duidelijker wat er op je pagina's te vinden is (*en dat verhoogt je bezoekersaantal*).

Je vindt een overzicht van de semantische tags op volgende [MDN pagina](#). Ondertussen zijn er ongeveer een honderdal semantische element beschikbaar.

#### 9.1.3 VOORDELEN

Zoals hierboven al even aangehaald zijn er een aantal voordelen gekoppeld aan het gebruik van semantische elementen. We sommen er even een aantal op.

- Zoekmachines beschouwen de inhoud ervan als belangrijke zoekwoorden om de ranking van de pagina te bepalen.  
*(hier spreken we van SEO)*
- Schermlezers kunnen het gebruiken als een wegwijzer om gebruikers met een visuele beperking te helpen doorheen een pagina te navigeren.
- Het vinden van blokken met betekenisvolle code is aanzienlijk eenvoudiger dan zoeken door eindeloze divs met of zonder semantische elementen of betekenisvolle classes.
- Geeft een indicatie aan de developer over het type gegevens die hij daar zal terugvinden.

Wanneer je nadenkt over welk(e) element(en) je dan precies moet gaan gebruiken, stel je je dus best een aantal vragen.

- Welk(e) element(en) beschrijven het beste de inhoud die ik wil presenteren?
  - Is dit bijvoorbeeld een lijst met data?
  - Is deze data geordend of niet geordend?
  - Past deze data beter in een tabel?
- Werk ik met een artikel dat alinea's nodig heeft en al dan niet randinformatie bevat?
- Heb ik een figuur of afbeelding die een onderschrift nodig heeft?
- Dient mijn element een afzonderlijke header en/of footer te hebben of is de pagina header en/of footer voldoende?
- ...

Verder in dit hoofdstuk geven we nog een kleine flowchart mee om je keuze misschien wel wat makkelijker te maken kan.

## 9.2 EEN AANTAL SEMANTISCHE ELEMENTEN

### 9.2.1 HEADER ELEMENT

Type: *block element*

Het **header** element vertegenwoordigt een inleidende inhoud. In vele gevallen is de inhoud van dit element een geheel van navigatie en inleidende informatie. Dit element bevat vaak een of meerdere **h** elementen, een logo, een zoekformulier, de naam van de auteur, ... . Dit element is niet te verwarren met het **head** element!

```
<header>
    
    <h1>Website Title/Company Name</h1>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
</header>
```

Voorbeeld van het gebruik van een **header** element

### 9.2.2 NAV ELEMENT

Type: *block element*

Het **nav** element vertegenwoordigt het gedeelte van een pagina waarop er navigatielinks terug te vinden zijn. Deze links mogen zowel binnen het huidige document als naar andere documenten linken. Het **nav** element bevat veelal navigatiemenu's, inhoudsopgaven of indexen.

```
<header>
    
    <h1>Website Title/Company Name</h1>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
</header>
```

Voorbeeld van het gebruik van een **nav** element

- Het is niet noodzakelijk dat alle links op je pagina zich in een **nav** element bevinden. Het **nav** element is enkel bedoeld voor grote blokken navigatielinks.
- Een document kan gerust verschillende **nav** elementen bevatten. Bijvoorbeeld een **nav** element voor sitenavigatie en een voor paginanavigatie.

### 9.2.3 MAIN ELEMENT

Type: block element

Het **main** element vertegenwoordigt de hoofd inhoud van het **body** element van onze pagina. Inhoudelijk dient dit element dan ook te bestaan uit inhoud die direct verband houdt met het onderwerp van de pagina.

```
<body>
  <header>
    ...
    <nav>
      ...
    </nav>
  </header>
  <main>
    <!-- Hoofdinhoud van de pagina -->
  </main>
  <footer>
    ...
  </footer>
</body>
```

Voorbeeld van het gebruik van een **main** element

De inhoud van een **main** element moet uniek zijn en eigen aan de pagina. Inhoud die wordt herhaald op andere pagina's of in andere elementen zoals zijbalken, navigatielinks, auteursrechtinformatie, site-logo's en zoekformulieren, hoort niet in het **main** element thuis.

Opgelet, het **main** element levert geen bijdrage aan de outline van de pagina. Met andere woorden, in tegenstelling tot elementen zoals **body**, koppen zoals **h2** en dergelijke, heeft het **main** element geen invloed op het DOM-concept van de structuur van de pagina. Het is dus louter informatief.

### 9.2.4 ARTICLE ELEMENT

Type: block element

Het **article** element dient inhoud te bevatten die op zichzelf kan (be)staan. Wat wil zeggen dat het **article** element in principe zou moeten kunnen geknipt en geplakt worden op een andere pagina/site zonder dat

er verdere verduidelijking van de inhoud nodig is. Prima voorbeelden hiervan zijn: een forumbericht, een tijdschrift- of krantenartikel of een blogbericht.

```
<article class="film-review">
  <header>
    <h2>Jurassic Park</h2>
  </header>
  <section class="main-review">
    <p>Dinos were great!</p>
  </section>
  <section class="user-reviews">
    <article class="user-review">
      <p>Way too scary for me.</p>
      <footer>
        <p>Posted on <time datetime="2015-05-16 19:00">May 16</time> by Lisa.</p>
      </footer>
    </article>
    <article class="user-review">
      <p>I agree, dinos are my favorite.</p>
      <footer>
        <p>Posted on <time datetime="2015-05-17 19:00">May 17</time> by Tom.</p>
      </footer>
    </article>
  </section>
  <footer>
    <p>Posted on <time datetime="2015-05-15 19:00">May 15</time> by Staff.</p>
  </footer>
</article>
```

Voorbeeld gebruik van het **article** element (bron: © developer.mozilla.org)

- Elk **article** element moet worden geïdentificeerd, meestal door een kop (*h2 - h6 element*) op te nemen als een child element van het **article** element.
- Wanneer een **article** element is genest, vertegenwoordigt het binnenste element een artikel met betrekking tot het buitenste element. De opmerkingen van een blogpost kunnen bijvoorbeeld **article** elementen zijn die genest zijn in het **article** element dat de blogpost vertegenwoordigt.
- De publicatiedatum en -tijd van een **article** element kunnen worden beschreven met behulp van het **datetime** attribute van een **time** element (*zie verder voor het time element*).

## 9.2.5 SECTION ELEMENT

Type: block element

Het **section** element vertegenwoordigt een op zichzelf staande sectie/onderdeel, waar geen specifieker semantisch element van toepassing is. Doorgaans, maar niet altijd, hebben secties ook een **h2-h6** element.

```
<section>
    <h2>Introductie</h2>
    <p>Mensen vangen vis als voedsel sedert het begin van de geschiedenis...</p>
</section>
<section>
    <h2>Uitrusting</h2>
    <p>Het eerste wat je nodig hebt is een hengel...</p>
</section>
```

Voorbeeld van het gebruik van een section element

## 9.2.6 ASIDE ELEMENT

Type: block element

Het **aside** element wordt gebruikt om een deel van de pagina aan te duiden waarvan de inhoud alleen indirect gerelateerd is aan de hoofdinhoud van de pagina. Los daarvan wordt het **aside** element vaak gebruikt voor zijbalken of call-out boxen (*reclame doeleinden*).

```
<article>
    <p>
        <cite>Avengers: Endgame</cite> is een Amerikaanse superheldenfilm uit
        2019 van
        Marvel Studios, geregisseerd door Anthony en Joe Russo.
        De film is het zowel chronologisch als verhaaltechnisch het vervolg op
        <cite>Avengers: Infinity War</cite> en de climax van de eerste 22 films
        in de
        Marvel Cinematic Universe.
    </p>
    <aside>
        <p>
            De film verbrak al binnen 6 uur tijd het wereldrecord
            "meeste kaartverkopen in 24 uur".
        </p>
    </aside>
    <p>
        Meer informatie over de film...
    </p>
</article>
```

Voorbeeld van mogelijk gebruik van het aside element

## 9.2.7 FOOTER ELEMENT

Type: block element

Het **footer** element vertegenwoordigt een voettekst voor de dichtstbijzijnde sectie-inhoud of sectie-element. Een voettekst bevat meestal informatie over de auteur van de sectie, auteursrechtgegevens of links naar gerelateerde documenten.

```
<footer>
  <small>&copy; Our fansite copyright notice</small>
</footer>
```

Voorbeeld van het gebruik van een footer element

## 9.2.8 TIME ELEMENT

Type: inline element

Het **time** element vertegenwoordigt een specifieke tijdsperiode. Mogelijk kan het **time** element aangevuld worden met het **datetime** attribuut om zo datums te vertalen in een ‘leesbaar’ formaat voor onze browser. Door het gebruik hiervan krijg je betere resultaten van zoekmachines of mogelijks aangepaste/bijkomende functionaliteit zoals de optie om een herinnering in te stellen.

```
<p>Het festival start om <time datetime="2018-07-
07T20:00:00">20:00</time>.</p>
```

Voorbeeld van het gebruik van een **time** element

Denk eraan om het formaat van de datum in het **datetime** attribuut correct te noteren. Dit is namelijk *yyyy-MM-ddT24:59:59*

## 9.2.9 FIGURE EN CAPTION ELEMENT

Type: block element

Het **figure** element (*afbeelding met optioneel bijschrift*) kan gebruikt worden voor op zichzelfstaande inhoud. Dit element kan mogelijk vergezeld worden van een optioneel bijschrift. Indien gewenst, wordt het bijschrift toegevoegd door middel van het **figcaption** element.

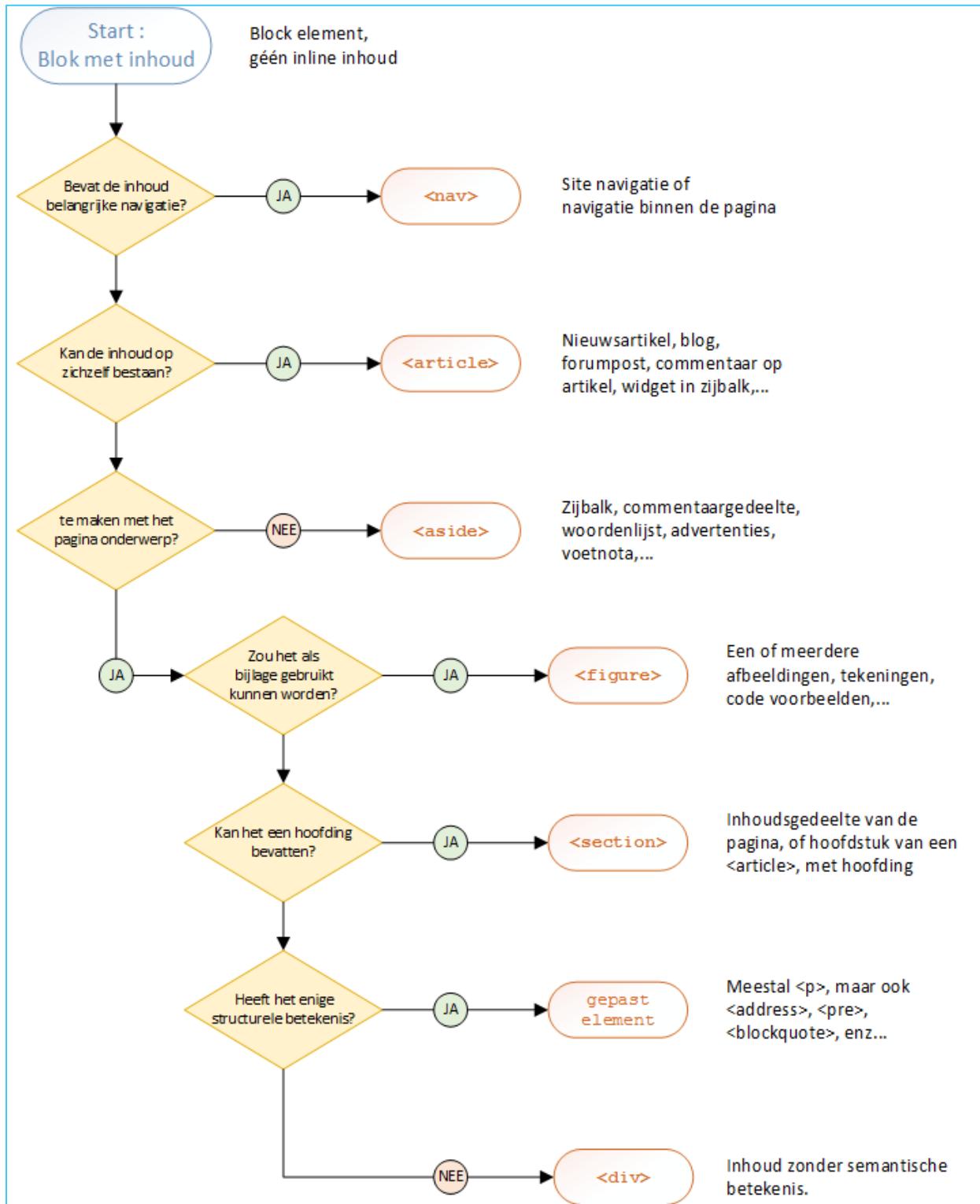
```
<figure>
  
  <figcaption>Charly sleeping in a box</figcaption>
</figure>
```

Voorbeeld van het gebruik van een **figure** element

- Gewoonlijk is een **figure** element een afbeelding, illustratie, diagram, codefragment, ... waarnaar verwzen kan worden in de algemene inhoud van het document zelf. Het zou echter wel mogelijk moeten zijn het **figure** element te verplaatsen naar een ander deel van het document of naar een bijlage zonder dat dit de algemene inhoud beïnvloedt.
- Het **figure** element heeft ook nog een unieke eigenschap als het op de Outline aankomt. Dit element wordt namelijk uitgesloten van de algemene outline van het document.

### 9.3 TO DIV OR NOT TO DIV?

Als je twijfelt wanneer je met een **div** element aan de slag zou gaan of wanneer je dit net niet zou doen dan kan onderstaande flowchart je misschien op weg helpen.









# Dieper in op CSS

Web Frontend Basics

## INHOUD

|  |          |
|--|----------|
| <b>10 FOCUS OP CSS</b>   | <b>3</b> |
| <b>10.1 Inheritance</b>  | <b>4</b> |
| <b>10.2 Voorrangsregels de basis</b>   | <b>5</b> |
| <b>10.3 CSS Selectoren – uitbreidung</b>                                       | <b>8</b> |
| 10.3.1 Selectie op attribuut   | 8        |
| 10.3.2 Combineren van selectoren - basis                                       | 8        |
| 10.3.3 Groeperen   | 8        |
| 10.3.4 Combineren  | 9        |
| 1.1.1.1 Descendant combinator ( <i>nakomeling van ...</i> )                    | 9        |
| 1.1.1.2 Child combinator ( <i>kind van ...</i> )                               | 9        |
| 1.1.1.3 General sibling combinator ( <i>jongere broer/zus van ...</i> )        | 9        |
| 1.1.1.4 Adjacent sibling combinator ( <i>eerstvolgende broer/zus van ...</i> ) | 9        |
| 10.3.5 Selectie op Attribute value   | 10       |
| 1.1.1.5 [attr=value]   | 10       |
| 1.1.1.6 [attr^=value]  | 10       |
| 1.1.1.7 [attr =value]  | 10       |
| 1.1.1.8 [attr\$=value]   | 10       |
| 1.1.1.9 [attr*=value]  | 10       |
| 1.1.1.10 Pseudo class  | 11       |
| 10.3.6 Pseudo  | 11       |
| 1.1.1.11 Pseudo class  | 11       |

## 10 FOCUS OP CSS

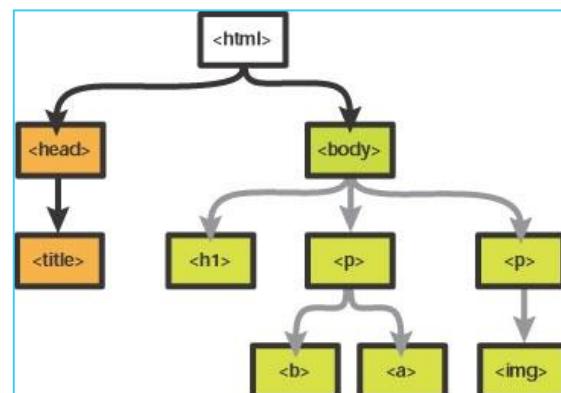
In dit hoofdstuk gaan we iets dieper in op CSS. We bekijken even hoe die overerving (cascading) nu in elkaar zit en we duiken vervolgens even wat dieper in op de selectors. Wat is er allemaal voor handen en hoe kan ons dit helpen met de opmaak van onze pagina's?

Zoals we reeds eerder zagen staat het acroniem **CSS** voor **Cascading Style Sheets**. En net dat eerste woord, **cascading**, is ongelooflijk belangrijk om te begrijpen. De manier waarop de **cascade** zich gedraagt is dan ook meteen de sleutel tot het begrijpen van CSS.

Gaande weg zal je tijdens het werken aan een pagina ongetwijfeld merken dat je **CSS**, waarvan je dacht dat die op een element zou toegepast worden, niet werkt! Geen paniek echter. Het probleem ligt meestal in het feit dat er meerdere regels **CSS** bestaan die op hetzelfde element van toepassing kunnen zijn. De **cascade** en het nauw verwante concept van **specificity** zijn mechanismen die bepalen welke regel van toepassing is wanneer er een dergelijk conflict is. De stijlregel die uiteindelijk toegepast wordt op je element is dan misschien niet degene die je verwacht. Belangrijk hier dus om te begrijpen waar en hoe je dit kan oplossen.

Ook belangrijk binnen **CSS** is het concept van **overerving** (*inheritance*). **Overerving** betekent dat sommige CSS properties standaard waarden overnemen die zijn ingesteld op het bovenliggende element (*parent*) van het huidige element, en sommige dan ook weer niet. En het is net dit concept dat voor de nodige 'stijl'-verwarring kan zorgen tijdens je uitwerking.

Zoals bij alles lijkt dit op het eerste zicht een grote brok met moeilijk te begrijpen regels, maar eenmaal je er zelf even mee aan de slag zal gaan, wordt alles snel duidelijker.



## 10.1 INHERITANCE

Laten we eerst even stil staan bij het **inheritance** verhaal. In onderstaand voorbeeld hebben we een **ul** element, met daarin twee niveaus van geneste unordered lists. Op het buitenste **ul** element hebben we de properties **border**, **padding** en **color** een waarde gegeven. We kijken even naar de code.

```
<ul class="main">
  <li>Eerste item</li>
  <li>Tweede item
    <ul>
      <li>2.1</li>
      <li>2.2</li>
    </ul>
  </li>
  <li>Derde item
    <ul class="special">
      <li>3.1
        <ul>
          <li>3.1.1</li>
          <li>3.1.2</li>
        </ul>
      </li>
      <li>3.2</li>
    </ul>
  </li>
</ul>
```

Voorbeeld HTML

```
.main {
  color: red;
  border: 2px solid greenyellow;
  padding: 1em;
}

.special {
  color: black;
  font-weight: bold;
}
```

Voorbeeld CSS

- Eerste item
- Tweede item
  - 2.1
  - 2.2
- Derde item
  - 3.1
    - 3.1.1
    - 3.1.2
  - 3.2

Output in de browser

Zoals je kan zien in dit voorbeeld is de kleur die we instellen voor het lettertype (*color*) een property welke '**overgeërfd**' wordt door de volgende, onderliggende, elementen. We kunnen dit aanpassen door een andere regel toe te voegen. Dit doen we op het derde item. We voorzien het van de **.special** class om een andere stijl op de tekst toe te passen. Vervolgens zie je dat deze aanpassing ook op zijn beurt **overgeerfd** wordt door de onderliggende elementen.

## 10.2 VOORRANGSREGELS DE BASIS

Omdat we voor elk element stijlregels kunnen definiëren, kunnen elementen soms verschillende conflicterende eigenschappen toegewezen krijgen. Onderstaand voorbeeld legt uit hoe stijlregels voorrang krijgen wanneer ze botsen met elkaar.

We voegen volgende regel HTML toe in de body van een html document, we voorzien hier ook een basis CSS selector voor in bijhorende CSS stylesheet.

```
<p>Spookeilanden zijn eilanden die oorspronkelijk op kaarten te vinden waren maar die waar later voor bewezen werd dat ze helemaal niet bestaan.</p>
```

Voorbeeld HTML

```
p {  
    font-family: Tahoma;  
    font-size: 12pt;  
    color: lightblue;  
}
```

Voorbeeld CSS

We zorgen er dus met deze selector voor dat elk **p** element een aangepaste stijl zal krijgen. Laten we even het **p** element dupliveren en voorzien van een **class** attribuut.

```
<p class="olivegreen">Spookeilanden zijn eilanden die oorspronkelijk op kaarten te vinden waren maar die waar later voor bewezen werd dat ze helemaal niet bestaan.</p>
```

Toevoeging in HTML

```
.olivegreen {  
    color: greenyellow;  
}
```

Toevoeging in CSS

Bij het bekijken van onze pagina, zien we dat wat we definiëerden in de **class oliffgroen** effectief voorrang krijgt op de eerder gemaakte selector op **element** niveau.

We dupliveren dit laatst toegevoegde **p** element met **class** attribuut nogmaals en deze keer voegen we er nog een **id** attribuut aan toe.

```
<p id="important" class="olivegreen">Spookeilanden zijn eilanden die oorspronkelijk op kaarten te vinden waren maar die waar later voor bewezen werd dat ze helemaal niet bestaan.</p>
```

Toevoeging in HTML

```
#important {  
    color: red;  
    font-weight: bold;  
}
```

Toevoeging in CSS

We zien nu dat de selector die het **id** bevat de bovenhand neemt op alle eerder gedefinieerde regels! Laten we nog 1 stapje verder gaan en ons **p** element nog een laatste keer gaan dupliveren. Deze keer voegen we er het **style** attribuut aan toe.

```
<p id="important" class="olivegreen" style="color:purple; font-size:20pt;">Spookeilanden zijn eilanden die oorspronkelijk op kaarten te vinden waren maar die waar later voor bewezen werd dat ze helemaal niet bestaan.</p>
```

Toevoeging in HTML



### Opgelet

We voegen in dit voorbeeld twee maal een HTML element toe met een identieke waarde in het **id** attribuut. Dit is louter om de voorrangsregels in CSS te illustreren. Je HTML pagina's dienen steeds unieke **id** waarden te bevatten.

Na het bekijken van bovenstaand voorbeeld zien we dat uiteindelijk de *inline* gedeclareerde **style** de bovenhand neemt van alle eerder gedefinieerde stijlen.

Hoe zit deze voorrang dan concreet in elkaar? We sommen het even op.

- 1 Standaard-stijlen van de browser worden toegepast op een HTML element.
- 2 Selectie op element niveau krijgt voorrang op regel 1.
- 3 Selectie op class niveau krijgt voorrang op regel 2.
- 4 Selectie op id niveau krijgt voorrang op regel 3.
- 5 Inline style declaratie krijgt voorrang op bovenstaande.

Dit zijn de basisregels om rekening mee te houden. Er speelt echter nog een extra regel een grote rol in deze voorrang en dat is de **specificity** van de selector zelf. Verdiep jezelf gerust even verder in deze materie, maar voor de scope van deze cursus houden we het voorlopig even op deze 5 algemene regels.

Op volgende pagina kan je alvast één van de vele **specificity** overzichten terugvinden die je online als leidraad kan terugvinden.

# Stuff & Nonsense



**a**

1 x element selector



**p a**

2 x element selectors



**.foo**

1 x class selector \*



**a.foo**

1 x element selector  
1 x class selector

Sith power: 0,0,1

Sith power: 0,0,2

Sith power: 0,1,0

Sith power: 0,1,1



**p a.foo**

2 x element selectors  
1 x class selector



**.foo .bar**

2 x class selectors



**p.foo a.bar**

2 x element selectors  
2 x class selectors



**#foo**

1 x id selector

Sith power: 0,1,2

Sith power: 0,2,0

Sith power: 0,2,2

Sith power: 1,0,0



**a#foo**

1 x element selector  
1 x id selector



**.foo a#bar**

1 x element selector  
1 x class selector  
1 x id selector



**.foo .foo #foo**

2 x class selectors  
1 x id selector



**style**

1 x style attribute

Sith power: 1,0,1

Sith power: 1,1,1

Sith power: 1,2,0

Sith power: 1,0,0,0



\* Same specificity

class selector =  
attribute attribute =  
pseudo-classes



!important

## 10.3 CSS SELECTOREN – UITBREIDING

Tot hertoe zagen we reeds geregeld volgende CSS selectoren de revue passeren.

Op element niveau

```
p {  
    property: value;  
}
```

Op class niveau

```
.class-name {  
    property: value;  
}
```

Op id niveau

```
#id-name {  
    property: value;  
}
```

In dit deel van de cursus bekijken we nog even een aantal andere mogelijkheden die we met CSS selectoren vorhanden hebben.

### 10.3.1 SELECTIE OP ATTRIBUUT

Volgende selector maakt het ons mogelijk om een selectie te maken op alle elementen die dit attribuut bevatten.

```
[lang]{  
    background-color: red;  
}
```

Alle elementen die het **lang** attribuut bevatten, zullen een rode achtergrondkleur krijgen.

### 10.3.2 COMBINEREN VAN SELECTOREN - BASIS

Het staat ons uiteraard vrij om onze selectoren specifieker te maken door wat we tot hertoe hebben te combineren. Het eenvoudig combineren doen we door bijvoorbeeld de selectoren aan elkaar te gaan schrijven. We geven hier alvast een aantal voorbeelden van.

```
div.eerste{  
    property: value;  
}
```

Alle **div** elementen die de **class eerste** hebben.

```
p.info{  
    property: value;  
}
```

Alle **p** elementen die de **class info** hebben.

```
a[href]{  
    property: value;  
}
```

Alle **a** elementen die een **href** attribuut hebben.

### 10.3.3 GROEPPEREN

We kunnen perfect een stijl overeen meerdere selectoren laten toepassen. Dit doen we door een **komma** te gebruiken in de opmaak van onze selector.

```
div, p{  
    property: value;  
}
```

Voorbeeld zal de stijl toepassen op alle **div** en alle **p** elementen.

Hou er rekening mee dat alle selectoren die verder aan bod komen, gegroepeerd kunnen worden met behulp van deze notatie. En dit ongeacht de complexiteit!

#### 10.3.4 COMBINEREN

##### 1.1.1.1 DESCENDANT COMBINATOR (NAKOMELING VAN ...)

Door middel van een **spatie** kunnen we een selectie maken op elementen die nakomelingen zijn van de eerste selector. Met andere woorden: het tweede deel na de **spatie** bevindt zich in het voorgaande element. Hoe diep deze genest is, is niet van belang.

```
div p{  
    property: value;  
}
```

Zal alle p elementen selecteren die zich in een div element bevinden.

##### 1.1.1.2 CHILD COMBINATOR (KIND VAN ...)

Met het **'groter dan'** teken kunnen we elementen selecteren die rechtstreeks een **child** elementen zijn van het voorgaande element. Dus enkel en alleen de elementen die het eerste element als parent hebben.

```
div > p{  
    property: value;  
}
```

Zal alle p elementen selecteren die als parent element een div element hebben.

##### 1.1.1.3 GENERAL SIBLING COMBINATOR (JONGERE BROER/ZUS VAN ...)

De tilde (~) is vervolgens onze optie om 'broers en zussen' die na het eerste element komen te selecteren.

```
p ~ span{  
    property: value;  
}
```

Zal alle span elementen selecteren die hetzelfde parent element van het p element delen en na dat p element voorkomen.

##### 1.1.1.4 ADJACENT SIBLING COMBINATOR (EERSTVOLGENDE BROER/ZUS VAN ...)

Rest er ons nog de + combinator, deze zal enkel de aangrenzende 'broer/zus' selecteren. Dit betekent dat het element direct volgend op het eerste met dezelfde **parent**.

```
h2 + p{  
    property: value;  
}
```

Zal alle p elementen selecteren die net na een h2 element volgen EN hetzelfde parent element delen.

Hou er rekening mee dat we in de voorbeelden steeds werken met een selectie op element niveau. Je kan deze combinators echter gebruiken met alle CSS syntax die we in deze cursus zullen zien.

### 10.3.5 SELECTIE OP ATTRIBUTE VALUE

We zagen reeds dat het mogelijk is om ook op **attribute** naam te selecteren, wel we kunnen zelfs nog een stapje verder gaan en dit uitbreiden met een selectie op een specifieke value die voorkomt in deze **attribute**.

#### 1.1.1.5 [ATTR=VALUE]

Selecteert elementen waarvan de **value** van het opgegeven **attribute** overeenkomt met de opgegeven waarde.

#### 1.1.1.6 [ATTR~=VALUE]

Selecteert elementen waarvan de **value** van het opgegeven **attribute** voorkomt als volledig woord.

#### 1.1.1.7 [ATTR|=VALUE]

Selecteert elementen waarvan de **value** van het opgegeven **attribute** start met de opgegeven waarde en gevuld wordt door een – (*streepje*).

#### 1.1.1.8 [ATTR^=VALUE]

Selecteert elementen waarvan de **value** van het opgegeven **attribute** start met de opgegeven waarde.

#### 1.1.1.9 [ATTR\$=VALUE]

Selecteert elementen waarvan de **value** van het opgegeven **attribute** eindigt met de opgegeven waarde.

#### 1.1.1.10 [ATTR\*=VALUE]

Selecteert elementen waarvan de **value** van het opgegeven **attribute** de opgegeven waarde bevat.

Onderstaand een aantal voorbeelden ter illustratie van bovenstaande selectoren.

|   |   |
|---|---|
| <pre>&lt;form&gt; &lt;input type="email" placeholder="E-mail" /&gt; &lt;br /&gt; &lt;input type="password" placeholder="Password" /&gt; &lt;br /&gt; &lt;/form&gt; &lt;hr /&gt; &lt;img src="images/voorbeeld.png"       alt="spookje in het wit" /&gt; &lt;img src="images/voorbeeld.png"       alt="spook kasteel" /&gt; &lt;img src="images/voorbeeld.png"       alt="zie je het spook?"/&gt; &lt;img src="images/voorbeeld.png"       alt="spook" /&gt; &lt;hr /&gt; &lt;a href="https://www.eensite.com"&gt;EEN LINK&lt;/a&gt; &lt;br /&gt; &lt;a href="http://www.eensite.be"&gt;EEN LINK&lt;/a&gt; &lt;br /&gt; &lt;a href="https://leho.howest.be"&gt;Leho Howest&lt;/a&gt;</pre> | <pre>input[placeholder="Password"]{     background-color: orange;     margin: 10px 0; } input[placeholder =E] {     background-color: red;     margin: 10px 0; } img[alt~="spook"] {     border: 10px solid red; } a[href^="https"]{     color: lawngreen; } a[href\$=".be"] {     color: yellow; } a[href*=eensite] {     color: pink; }</pre> |
|---|---|

## 10.3.6 PSEUDO

### 1.1.1.11 PSEUDO CLASS

Een **pseudo class** is een selector die elementen selecteert die zich in een **specifieke status** bevinden. Het element is bijvoorbeeld het eerste element van hun type of het element wordt aangeduid door de cursor. Ze helpen je dus met reeds vooraf gedefinieerde logica aan je CSS selectors toe te voegen.

Een pseudo class begint steeds met een dubbele punt (:). We geven een aantal voorbeelden van hoe je zo'n **pseudo class** kan gebruiken.

```
input[type=checkbox]:checked + label {  
    color: red;  
    font-weight: bold;  
}
```

Voorbeeld van het gebruik van een pseudo class samen met een combinator

Door gebruik te maken van de **:checked** pseudo class kunnen we nagaan als de checkbox in kwestie al dan niet aangevinkt is. Als we vervolgens ook nog gebruik maken van een combinator kunnen we het eerst volgende label element dan ook van een passende stijl gaan voorzien op het moment van de selectie.

Een **pseudo class** die je al iets vaker zal tegenkomen is de **:visited**. Het viel je misschien wel al op dat **a** elementen die je reeds bezocht een default opmaak meekrijgen van de browser zelf (*Useragent stylesheet*). Wel, deze opmaak kunnen we zelf bepalen door middel van de **:visited pseudo class**.

```
a:visited{  
    font-weight: bold;  
    color: orchid;  
}
```

Voorbeeld gebruik van de **:visited pseudo class**

Nog een ander voorbeeld dat vrij vaak gebruikt wordt is de **:nth-child pseudo class**. Hiermee kan je de opmaak instellen van een element dat op een bepaalde positie voorkomt binnen zijn bovenliggend element (= **parent**). Zo kan je bv. eenvoudig alternerende opmaak voorzien binnen lijsten en tabellen. In het onderstaande voorbeeld krijgt elke *derde* rij binnen de lijst een lichtgrijze achtergrondkleur. Er bestaan ook voorgedefinieerde specifieke versies zoals **:first-child** en **:last-child**.

```
li:nth-child(3n){  
    background-color: lightgray;  
}
```

Voorbeeld gebruik van de **:nth-child pseudo class**



#### Overzicht pseudo classes

Daar er een hele resem aan **pseudo classes** beschikbaar is, verwijzen we graag verder naar een overzicht hiervan dat terug te vinden is op MDN.

[Overzicht Pseudo classes op MDN](#)





# Webformulieren

**Web Frontend Basics**

## INHOUD

|   |           |
|---|-----------|
| <b>11 IN CONTACT MET JE GEBRUIKERS VIA WEBFORMULIEREN</b> | <b>3</b>  |
| <b>11.1 Interactie met de webserver</b>                   | <b>3</b>  |
| <b>11.2 Overzicht van de formuliervelden</b>              | <b>3</b>  |
| <b>11.3 Het form element</b>                              | <b>4</b>  |
| 11.3.1 Form attributes                                    | 4         |
| method  | 4         |
| action  | 5         |
| accept-charset  | 5         |
| autocomplete  | 5         |
| enctype   | 5         |
| name  | 6         |
| novalidate  | 6         |
| target  | 6         |
| <b>11.4 Het input element</b>                             | <b>7</b>  |
| 11.4.1 Input attributes                                   | 7         |
| 11.4.2 Tekst invoervakken                                 | 7         |
| Eenvoudig tekstvak met een enkele regel                   | 7         |
| Tekstvakken met een specifiek doel                        | 9         |
| Tekstvak met meerdere regels                              | 11        |
| 11.4.3 Datum gerelateerde input velden                    | 12        |
| datetime-local  | 12        |
| date  | 12        |
| time  | 13        |
| month   | 13        |
| week  | 13        |
| Algemeen  | 13        |
| 11.4.4 Numeriek input veld                                | 14        |
| number  | 14        |
| 11.4.5 Range input veld                                   | 15        |
| range   | 15        |
| 11.4.6 Kleur input veld                                   | 15        |
| color   | 15        |
| 11.4.7 Radioknoppen en checkboxen                         | 16        |
| 11.4.8 Neerklapmenu's en menulijsten                      | 17        |
| list attribute op input type 'x'                          | 18        |
| 11.4.9 Verborgen velden                                   | 18        |
| 11.4.10 Bestanden selecteren                              | 18        |
| 11.4.11 Knoppen   | 19        |
| <b>11.5 velden Organiseren</b>                            | <b>20</b> |
| 11.5.1 Tabvolgorde  | 20        |
| 11.5.2 Sneltoetsen  | 20        |
| <b>11.6 formulieren Versturen</b>                         | <b>20</b> |

## 11 IN CONTACT MET JE GEBRUIKERS VIA WEBFORMULIEREN

### 11.1 INTERACTIE MET DE WEBSERVER

Tot nu toe heb we enkel gezien hoe we informatie beschikbaar kunnen stellen via een webpagina. De gebruiker wil soms ook in staat zijn om bepaalde acties te ondernemen zoals zoeken, antwoorden op een artikel, zich inschrijven, enz. Hiervoor kunnen we gebruik maken van HTML-formulier elementen. Dit zijn elementen zoals tekstvakken, aankruisvakjes, neerklapmenu's, knoppen om op te klikken, enz.

Een webformulier doet precies wat je van een formulier verwacht: iemand vult het in en het wordt verstuurd naar een bepaalde plaats, waar de ingevoerde gegevens verwerkt moeten worden. We definiëren alle elementen die ons formulier zullen vormen binnen het **form** element. **Alle gerelateerde formulierelementen** moeten zich dus binnen dit onzichtbare **form** element bevinden!

Het is dus aan te raden om, wanneer je van plan bent gebruik te maken van form gerelateerde elementen, steeds een **form** tag te plaatsen zodra je aan een pagina begint.

De web resource (pagina, script, ...) die de informatie ontvangt, moet uiteraard in staat zijn om het uit te lezen. Vroeger werd er vaag gebruik gemaakt van CGI-BIN-scripts, maar tegenwoordig kan dit met server-side programma's zoals php, asp, asp.net, jsp, etc. Dat is echter leerstof buiten het bereik van deze cursus.

In de loop van dit hoofdstuk zullen we gebruik maken van reeds gemaakte pagina's en bestaande serverscripts om onze form te testen.

### 11.2 OVERZICHT VAN DE FORMULIERVELDEN

Met velden (of verder ook wel controls genoemd) bedoelen we de HTML-elementen die gebruikers in staat stellen om interactie te hebben met het formulier. De **value** attributen die we als programmeur eventueel meegeven, zijn steeds standaardwaarden en manipuleerbaar door interacties van de gebruiker.

Het zijn bijgevolg dus ook de **value** en **name** attributen die in combinatie doorgestuurd worden naar de server, zodat deze weet welke waarde bij welk control hoort.

## 11.3 HET FORM ELEMENT

Wanneer je elementen plaatst zoals tekstvakken, (radio)knoppen, vinkjes en dergelijke, dan moeten deze zich steeds binnen een **<form>** element bevinden. Dit is een element dat instructies bevat over hoe het formulier verzonden moet worden, en welke elementen tot het formulier behoren.

We kunnen meerdere **form** elementen op een pagina hebben, zolang ze elkaar maar niet overlappen.

```
<form method="post">
    <fieldset title="Persoonlijke informatie">
        <legend>Persoonlijke Informatie</legend>
        <div>
            <label for="txt-voornaam">Voornaam : </label>
            <input type="text" id="txt-voornaam" name="txt-voornaam" value="" />
        </div>
        <div>
            <label for="txt-naam">Naam : </label>
            <input type="text" id="txt-naam" name="txt-naam" value="" />
        </div>
    </fieldset>
</form>
```

Voorbeeld van het gebruik van een **form** element

Je merkt ook een zelfsluitend element op, **input** genaamd. Je zal zien dat dit element het meest voorkomt in dit hoofdstuk, omdat bijna alle velden op basis van het **input** element gemaakt worden.

### 11.3.1 FORM ATTRIBUTES

#### 1.1.1.1 METHOD

Het **method** attribuut specificeert welke http-methode (GET, POST) gebruikt zal worden om de formuliergegevens in te dienen.

In dit stuk van de cursus staan we voor onze **<form>** elementen even stil bij het gebruik van de **GET** en **POST** values voor dit attribute.

```
<form method="post">
    ...
</form>
```

Voorbeeld van het method attribuut op het form element

#### MOGELIJKE WAARDEN

##### GET ALGEMEEN

Als we kiezen om de **GET** value te gebruiken op ons method attribuut, dan zal de browser de formulierinhoud toevoegen aan het einde van de URL. Dit biedt een aantal voordelen voor eenvoudige formulieren. Hiermee kan de browser de resultaten van de formulierzending opslaan in het cachegeheugen, en kan de gebruiker ook een bladwijzer maken van de pagina nadat het formulier is verzonden. Een **GET** value voor een method attribute wordt dan ook over het algemeen gebruikt voor eenvoudige forms waarin beveiliging niet meteen belangrijk is.

## GET BEPERKINGEN

**GET** zal de volledige inhoud van wat er verzonden wordt zichtbaar maken in de URL. Als ons formulier dus gevoelige gegevens bevat, zijn we beter af met een **POST** value voor het **method** attribute. Aangezien **GET** de formuliergegevens aan de huidige URL toevoegt, kan deze alleen worden gebruikt als de inhoud van wat je zal verzenden (inclusief de volledige URL), resulteert in een reeks die minder dan 2048 tekens lang is. Dit is namelijk de maximaal aan te raden lengte van een URL. Je blijft hier echter best zo ver mogelijk onder om een functionele site overheen alle browsers te garanderen. Hou er ten slotte nog rekening mee dat een **GET** alleen kan worden gebruikt om ASCII-gegevens te verzenden (dus: geen speciale tekens).

## POST ALGEMEEN

Via een value **POST** in de **method** attribute zal de browser de gegevens naar een webserver versturen, zodat deze daar verwerkt kunnen worden. Dit is uiteraard nodig als we gegevens aan een database willen toevoegen of als er gevoelige informatie, zoals wachtwoorden, verzonden wordt.

## POST BEPERKINGEN

Wanneer er gegevens met **POST** worden verzonden, zal een formulier dat tweemaal ingediend wordt in een dubbele invoer resulteren (dit dient dan aan de serverzijde opgevangen te worden). Dit kan mogelijks een probleem zijn als het formulier gekoppeld is aan de aanvraag van een lidmaatschap, een online aankoop of eender welke andere eenmalige actie. Dit is dan ook meteen de reden waarom gebruikers de resultaten van een formulierzending niet kunnen opslaan als de gekozen **method** een value heeft van **POST**.

### 1.1.1.2 ACTION

De URI van een programma dat de formulierinformatie zal verwerken. Vanaf HTML5 is het niet langer verplicht om dit attribute op het form element te plaatsen. Dit dient een geldige, niet lege string te zijn.

```
<form method="post" action="https://jkorpela.fi/cgi-bin/echo.cgi">
```

### 1.1.1.3 ACCEPT-CHARSET

In dit attribute kan je verduidelijken welke charset(s) er door de server waarnaar je het formulier zal sturen, aanvaard worden. Voor een overzicht van mogelijke charsets kan je terecht op de [overzichtspagina van IANA](#) (*Internet Assigned Numbers Authority*).

```
<form method="post" action="https://jkorpela.fi/cgi-bin/echo.cgi" accept-charset="utf-8">
```

### 1.1.1.4 AUTOCOMPLETE

Autocomplete kan gebruikt worden om ervoor te zorgen dat eerder ingevulde gegevens niet verloren gaan als je de pagina verlaat. Het attribute **autocomplete** heeft twee mogelijke values, nl. **on** of **off**. We komen hier verder in dit hoofdstuk nog uitgebreid op terug met een aantal voorbeelden.

```
<form method="post" action="https://jkorpela.fi/cgi-bin/echo.cgi" autocomplete="on">
```

### 1.1.1.5 ENCTYPE

Het attribute enctype geeft aan hoe de formuliergegevens worden gecodeerd bij het indienen van het formulier bij de server. Het opgeven van een value in het enctype attribute werkt enkel indien het method attribute de value post bevat. De mogelijke waarden kunnen opgezocht worden in de [lijst van mogelijke](#)

MIME types, alvast wat web development betreft. Hou er rekening mee dat bij het werken met `<form>` elementen in de meeste gevallen onderstaande encatypes gebruikt worden:

- application/x-www-form-urlencoded (**default**)
- multipart/form-data (**gebruikt bij verzenden van bestanden**)
- text/plain

```
<form method="post" action="https://jkorpela.fi/cgi-bin/echo.cgi" enctype="text/plain">
```

#### 1.1.1.6 NAME

Via dit attribute kan je ervoor kiezen om je form te voorzien van een naam. Hou er rekening mee, indien je meerdere forms hebt op je pagina, dat de gekozen naam uniek dient te zijn per pagina. Tevens is een lege string niet toegelaten.

```
<form method="post" action="https://jkorpela.fi/cgi-bin/echo.cgi" name="my-form">
```

#### 1.1.1.7 NOVALIDATE

Het novalidate attribute geeft aan dat het formulier niet moet worden gevalideerd wanneer op de knop verzenden wordt geklikt. Je zal dus de standaard client side validatie van de browser niet laten triggeren door gebruik te maken van dit HTML5 boolean attribute.

```
<form method="post" action ="https://jkorpela.fi/cgi-bin/echo.cgi" novalidate>
```

#### 1.1.1.8 TARGET

Het target attribute geeft het doelvenster op om het antwoord weer te geven dat is ontvangen na het indienen van het formulier. Hier kan je gelijkaardig aan het gebruik van target bij het `<a>` element verduidelijking geven door middel van de volgende values te specifiëren:

- `_blank` Het antwoord wordt weergegeven in een nieuw venster of tabblad
- `_self` Het antwoord wordt weergegeven in hetzelfde frame (default)
- `_parent` Het antwoord wordt weergegeven in de parent frame
- `_top` Het antwoord wordt weergegeven in de body van het huidige venster
- `framename` Het antwoord wordt weergegeven in een benoemde iframe

```
<form method="post" action="https://jkorpela.fi/cgi-bin/echo.cgi" target="_blank">
```

Al deze attributes kunnen uiteraard gecombineerd worden in het opmaken van het `<form>` element. Hieronder vind je een voorbeeld van een dergelijke combinatie.

```
<form method="post" action="https://jkorpela.fi/cgi-bin/echo.cgi" accept-charset="utf-8"  
      autocomplete="on" enctype="text/plain" name="my-form" novalidate target="_blank">  
  
    <!-- Inhoud van het formulier -->  
  
</form>
```

Voorbeeld van mogelijke toepassing van attributen op het form element

## 11.4 HET INPUT ELEMENT

De HTML5 standaard bevat een hele reeks nieuwe elementen en standaardattributen voor de bouw van formulieren. De toevoegingen hebben voornamelijk betrekking op de gebruikerservaring en de nieuwe generatie browsers en mobiele apparaten. In dit stuk gaan we iets dieper in op de mogelijkheden van het **input** element.

### 11.4.1 INPUT ATTRIBUTES

Ons **input** element heeft als belangrijkste attribute het **type** attribute. Dit attribute zal ervoor zorgen dat we een betekenis en functionaliteit kunnen toekennen aan de verschillende soorten input die een gebruiker kan geven bij het gebruik van ons formulier. Een **input** element kan niet bestaan zonder waarde in het type attribute. Het is dus **verplicht** het te specifiëren. Om alles wat overzichtelijker te maken, proberen we in onderstaande secties de types, gegroepeerd per functionaliteit, verder in detail te bespreken.

### 11.4.2 TEKST INVOERVAKKEN

We onderscheiden drie soorten tekstvakken:

- Eenvoudig tekstvak met een enkele regel.
- Tekstvak met een specifiek doel (paswoord, telefoonnummer, e-mail, ...)
- Tekstvak dat bestaat uit meerdere regels.

#### 1.1.1.9 EENVOUDIG TEKSTVAK MET EEN ENKELE REGEL

Om een eenvoudig tekstvak of een paswoord veld te maken, gebruiken we het **input** element met *text* als value van het **type** attribute. We bespreken in de volgende secties uitvoerig het gebruik en de mogelijke toepassingen van attributes aan de hand het een **input** element met het **type** *text*.

```
<input type="text" />
```

Voorbeeld van een **input** element met type *tekst*

Deze HTML-code is reeds voldoende om onderstaande op het scherm te doen verschijnen.

Het spreekt natuurlijk voor zich dat we hier nog wel een aantal andere zaken nodig hebben om de gebruikers duidelijk te maken wat er aan input verwacht wordt. Voor onszelf zullen we de nodige attributen moeten voorzien om de ingegeven waarden te ontvangen en te koppelen aan de informatie die we nodig hebben. Om de input van een gebruiker op een formulier daadwerkelijk ook te kunnen ontvangen, moet elke element een naam krijgen. Dit doen we via het **name** attribute.

```
<input type="text" name="first-name"/>
```

Voorbeeld van een **input** element met type *tekst* met *name* attribuut

Het is aan te raden om bij een **<input>** element steeds een label te voorzien, zodat de gebruikers weten wat er van hen verwacht wordt. Hiervoor kunnen we gebruik maken van een **<label>** element. Dit wordt gekoppeld aan het bijhorende **<input>** element door een verwijzing naar het **id** attribute van het **<input>** element waartoe het label behoort.

```
<label for="txt-first-name">Voornaam: </label>
```

```
<input type="text" name="first-name" id="txt-first-name"/>
```

## GEBRUIKSVRIENDELijkheid

Met de HTML-code uit bovenstaande voorbeelden hebben we al meteen een minimalistische input voorzien voor onze gebruikers. Echter, we kunnen nog enkele zaken toevoegen die het gebruiksgemak verhogen en ook wat meer begeleiding voorzien voor onze gebruikers tijdens het invullen van formulieren.

### PLACEHOLDER

Met het placeholder attribute voorzien we een hint in het tekstvak zolang er geen input gegeven werd. Van zodra er input is in het tekstvak zal de placeholder verdwijnen.

### MINLENGTH

De minlength laat ons, zoals de naam reeds doet vermoeden, een minimale lengte van de ingave gaan opgeven.

### MAXLENGTH

De onmiskenbare tegenhanger van minlength. Wederom duidelijk dat deze de maximale lengte van de ingave zal begrenzen.

### REQUIRED

Het required (boolean) attribute laat ons toe om aan te geven dat de input value met dit attribute verplicht in te vullen is door de gebruiker om het formulier geldig te verzenden.

Laten we bovenstaande attributes even samenvoegen om het effect te bekijken op ons **input** element.

```
<label for="txt-first-name">Voornaam: </label>
<input type="text" name="first-name"
       id="txt-first-name"
       placeholder="Vul je voornaam in ..."
       minlength="4" maxlength="25"
       required />
<input type="submit" value="Verzenden" />
```

Voorbeeld van een uitwerking van het input element met extra attribute waarden

Voornaam:

Vul je voornaam in ...

Verzenden

Output in een browser



Afhankelijk van de browser, zal je bij een blanco input element een melding krijgen dat dit een verplicht veld is.

|   |   |  |
|---|---|--|
| <p><b>Voornaam:</b></p> <input type="text" value="Vul je voornaam in ..."/> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <b>Vul dit veld in.</b> </div> | <p><b>Voornaam:</b></p> <input type="text" value="Vul je voornaam in ..."/> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <b>Vul dit veld in.</b> </div> | <p><b>Voornaam:</b></p> <input type="text" value="Vul je voornaam in ..."/> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <b>Dit veld is vereist</b> </div> |
|---|---|--|

We kunnen de verschillen in weergave naar de gebruiker toe ook detecteren indien we onze **minlength** attribute even uittesten.



|   |  |
|---|--|
| <p><b>Voornaam:</b></p> <input type="text" value="Mar"/> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <b>Breed deze tekst uit tot 4 tekens of meer (je gebruikt momenteel 3 tekens).</b> </div> | <p><b>Voornaam:</b></p> <input type="text" value="Mar"/> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <b>Gebruik minstens 4 tekens (u gebruikt momenteel 3 tekens).</b> </div> |
|---|--|

Ondanks het feit dat minlength een HTML5 attribute is, houdt de Edge browser hier geen rekening mee. Dit is iets wat je als developer zeker en vast steeds dient te controleren. Uiteraard kunnen we de manier waarop deze meldingen getoond worden, aanpassen en stijlen, zodat deze binnen de stijl van onze website vallen. Hierover meer in een ander hoofdstuk.

#### 1.1.1.10 TEKSTVAKKEN MET EEN SPECIFIEK DOEL

##### PASSWORD

In dit type van input veld wordt de input van de gebruiker gemaskeerd door middel van een vaste waarde per ingegeven karakter.

```
<input type="password" />
```

Voorbeeld van het *input* element met *type password*

Voorbeeld output in een browser

##### EMAIL

Het input type *email* zal in de huidige browsers een controle uitvoeren aan de gebruikerszijde op een herkenbaar patroon van een e-mail adres. Daarnaast zal er op een mobiele client op het **input** element ook

meteen de toetsenbord layout voorzien worden van de meest gebruikte karakters in het typen van een e-mail adres.

Indien de browser dit type niet ondersteunt, zal de input beschouwd worden als van het type *text*.

```
<input type="email" />
```

Voorbeeld van het input element met type *email*

Voor de volledigheid kan je onderstaand ook nog het patroon terugvinden dat de browsers (indien het type ondersteund wordt) zullen gebruiken om het e-mail address te valideren.

```
/^[\a-zA-Z0-9.!#$%&'*+\/=?^`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9]))?$/
```

## TEL

Dit input type heeft als hoofddoel de ingave van een telefoonnummer toe te laten. Omdat er wereldwijd zoveel verschillende formaten in telefoonnummers zijn, wordt op dit input type geen automatische validatie door de browser uitgevoerd. We kaarten hier nogmaals het belang aan voor het gebruikersgemak. Op een mobiele client zal voor deze input box het toetsenbord naar numeriek verspringen.

Indien de browser dit type niet ondersteunt, zal de input beschouwd worden als van het type *text*.

```
<input type="tel" />
```

Ook hier is het uiteraard mogelijk om de gebruiker via een placeholder het verwachte formaat te tonen.

```
<input type="tel" placeholder="04xx/12.34.56" />
```

## URL

De voornaamste kracht van het *url* input type ligt wederom in het beschikbaar stellen van een gepaste toetsenbord layout bij de mobiele gebruikers. Hier krijgen de gebruikers (in de meeste gevallen) een kleinere spatiebalk en reeds een forward slash en een .com button gepresenteerd. Bijkomend is er natuurlijk ook een basis validatie vanuit de browser die checkt of de ingegeven waarde overeenstemt met een herkenbaar patroon voor url's.

Indien de browser dit type niet ondersteunt, zal de input beschouwd worden als van het type *text*.

```
<input type="url" />
```

## SEARCH

Input elementen van het type *search* willen het gebruik ervan verduidelijken voor de gebruiker. Sommige browsers zullen dit input type voorzien van een eigen stijl (browser afhankelijk) zodat de gebruiker dit element sneller doet denken aan een mogelijke zoekopdracht.

De werking echter is identiek aan het *text* type. Indien de browser dit type niet ondersteunt, zal de input beschouwd worden als van het type *text*.

```
<input type="search" />
```

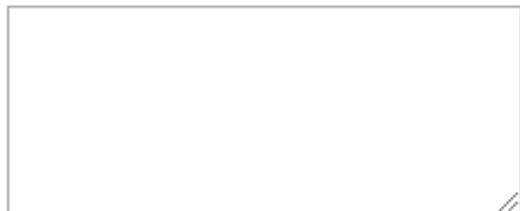
### 1.1.1.11 TEKSTVAK MET MEERDERE REGELS

Ons laatste soort tekstvak, is meteen ook een element op zich. Om een tekstvak te maken dat meerdere lijnen tekst kan bevatten, gebruik je het **textarea** element.

De grootte van het **textarea** element, kan beïnvloed worden door bijkomend de attributen **cols** en **rows** mee te geven.

Uiteraard zijn er nog een aantal andere mogelijke attributes maar deze bespreken we in het volgende stuk.

```
<textarea rows="5" cols="25"></textarea>
```



Voorbeeld output in browser

In tegenstelling tot de **input** elementen die we eerder bespraken, is het **textarea** element **niet zelfsluitend**. Hou hiermee zeker en vast rekening.

Via het **resize** element  in de rechterhoek onderaan het tekst vak zal de gebruiker kunnen schalen naar de gewenste grootte.

### 11.4.3 DATUM GERELATEERDE INPUT VELDEN

HTML5 biedt ons ook een aantal mogelijke types als we willen werken met datums. We overlopen dan ook even deze input types en hun gebruik.

#### 1.1.1.12 DATETIME-LOCAL

Dit input type laat ons op een makkelijke manier toe om met de moderne browsers de gebruiker een datum en tijd input te voorzien die zich aanpast naar de lokale instellingen gedetecteerd door de browser zelf.

```
<label for="dt-date-time">Local Date and Time: </label><br />
<input id="dt-date-time" type="datetime-local">
```

Laten we dit even verduidelijken met een voorbeeld, waar we ons input type gaan bekijken en onze lokale instellingen voor datumnotatie aanpassen op onze pc.

#### Datum tijd instelling Windows

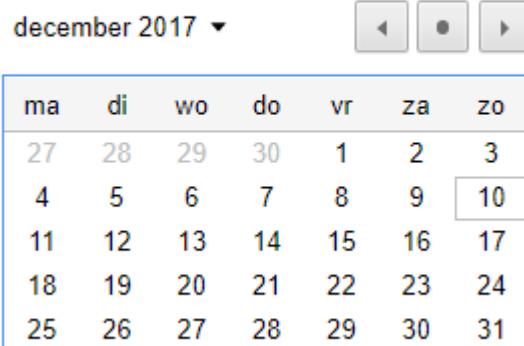
Korte datumnotatie

d/MM/jjjj

#### Resultaat in browser

Local Date and Time:

dd/mm/jjjj --:--



#### Datum tijd instelling Windows

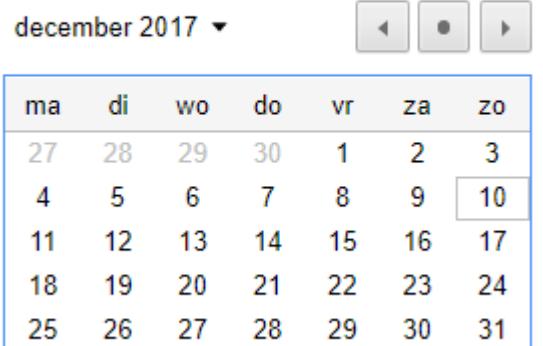
Korte datumnotatie

jjjj-MM-dd

#### Resultaat in browser

Local Date and Time:

jjjj-mm-dd --:--



#### 1.1.1.13 DATE

Dit input type laat de gebruiker net als in bovenstaande input type een datum ingeven.

```
<label for="dt-birthdate">Geboortedatum: </label><br />
<input id="dt-birthdate" type="date">
```

Geboortedatum:

dd/mm/jjjj

Voorbeeld output in browser

#### 1.1.1.14 TIME

Dit input type is voorzien om de gebruiker een ingave van tijd te laten maken.

```
<label for="dt-appointment">Uur: </label><br />
<input id="dt-appointment" type="time">
```

Uur:

Voorbeeld output in browser

#### 1.1.1.15 MONTH

Het month input type geeft de gebruiker wederom een date-picker ter beschikking, echter zal de gekozen datum zich vertalen in de maand en het jaar waarin deze datum valt.

```
<label for="dt-chosen-month">Maand: </label><br />
<input id="dt-chosen-month" type="month">
```

Maand:

Voorbeeld output in browser

#### 1.1.1.16 WEEK

Week als input type zal wederom via de date-picker werken, maar neemt van de gekozen datum de week over in het input element.

```
<label for="week">Week: </label><br />
<input id="week" type="week">
```

Week:

november 2017

| Week | ma | di | wo | do | vr | za | zo |
|------|----|----|----|----|----|----|----|
| 44   | 30 | 31 | 1  | 2  | 3  | 4  | 5  |
| 45   | 6  | 7  | 8  | 9  | 10 | 11 | 12 |
| 46   | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 47   | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 48   | 27 | 28 | 29 | 30 | 1  | 2  | 3  |

Voorbeeld output in browser

#### 1.1.1.17 ALGEMEEN

Indien de browser een van bovenstaande types niet ondersteunt, zal de input beschouwd worden als van het type *text*. Wil je de gebruiker toch nog ondersteunen met een soortgelijke functionaliteit, dan zal je dit met javascript moeten opvangen.

**Let op!** Het gebruik van het type **datetime** wordt niet langer als geldige input type ondersteund. Dit werd vervangen door **datetime-local**.

#### 11.4.4 NUMERIEK INPUT VELD

##### 1.1.1.18 NUMBER

We hebben standaard ook de optie om het type van ons **input** element voor cijfers te gebruiken. Hiervoor dient het **number** type.

```
<label for="score">Je Score: </label><br />
<input id="score" type="number">
```

De browser zal bij mobiele clients ook hier het keyboard aanpassen naar cijfers of in sommige gevallen met een lijst of draaiwiel (spinner) gaan werken. Op een desktop zal de browser aan de zijkant van het **<input>** element twee pijltjes voorzien om de waarde omhoog of omlaag te laten wijzigen.

We kunnen hier nog een aantal extra attributes meegeven om de input van onze gebruikers te begeleiden. Door gebruik te maken van het step attribute kunnen we bepalen met welke grootteorde een stap omhoog of omlaag gemaakt wordt. De ingegeven waarde zal dan steeds een veelvoud van onze step attribute moeten zijn.

```
<input id="score" type="number" step="2">
```

In bovenstaand voorbeeld dient er dus steeds een veelvoud van 2 ingevuld te worden.

Verder kunnen we de minimale en maximale waarden ook gaan begrenzen door gebruik te maken van de **min** en **max** attributes.

```
<input id="score" type="number" step="2" min="0" max="10">
```

Zorg ervoor dat je ook steeds duidelijk maakt aan de gebruikers wat er van hen verwacht wordt. Denk hiervoor aan het eerder besproken **placeholder** attribute.

```
<label for="score">Je Score: </label><br />
<input id="score" type="number"
placeholder="Geef een veelvoud van 2 in. (Max 10)"
step="2" min="0" max="10">
```

Je Score:

Geef een veelvoud van 2 in. (Max 10)

**Let op!** De enige beperking waarmee je voor het type number rekening dient te houden, is dat de standaard grootte van het step attribute een waarde van 1 heeft. Wil je de gebruikers toelaten om decimale waarden in te geven, dan zal je dit dus specifiek moeten integreren via het step attribute. Onderstaand een voorbeeld voor ingave met twee cijfers na de komma.

```
<input id="score" type="number" step="0.01">
```

Indien de browser dit type niet ondersteunt, zal de input beschouwd worden als van het type **text**.

## 11.4.5 RANGE INPUT VELD

Het range input type heeft ook zijn intrede gemaakt met HTML5. Range kan gebruikt worden indien je ergens een keuze wil laten maken aan de hand van een schuifbalk.

### 1.1.1.19 RANGE

```
<label for="satisfaction">Hoe tevreden ben je?: </label>
<br />
<input id="satisfaction" type="range" min="0" max="10">
```

Hoe tevreden ben je?: 

Bovenstaande code zal je de schuifbalk rechts als resultaat geven.

Omdat we meestal ook een schaal aanduiding voor onze gebruikers willen voorzien, is bovenstaande weergave niet meteen iets wat we op een formulier zouden plaatsen.

We kunnen voor dit type control een verbinding leggen naar een datalist. Die bevat de keuzemogelijkheden voor onze gebruikers.

```
<label for="satisfaction">Hoe tevreden ben je?:</label>
<br />
<input id="satisfaction" type="range" list="my-scale">
<datalist id="my-scale">
  <option value="0">
  <option value="1">
  <option value="2">
  <option value="3">
  <option value="4">
  <option value="5">
  <option value="6">
  <option value="7">
  <option value="8">
  <option value="9">
  <option value="10">
</datalist>
```

Hoe tevreden ben je?: 

**Let op!** Deze functionaliteit wordt enkel door de nieuwste browsers ondersteund.

## 11.4.6 KLEUR INPUT VELD

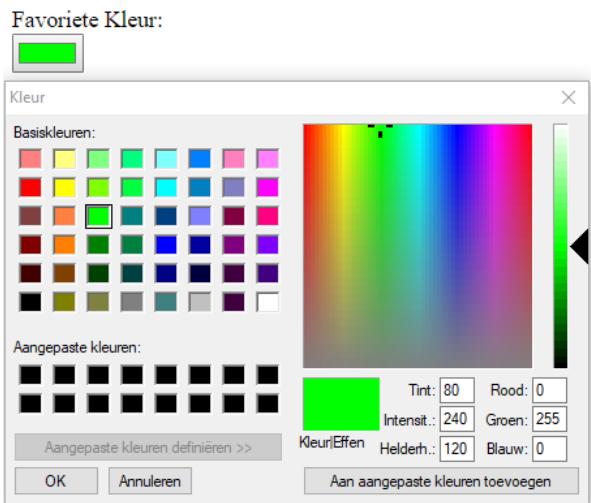
HTML5 introduceerde voor de input elementen ook het type **color**. Door een **input** element van het type color in je **form** element te voorzien, laat je de gebruiker toe om via een color picker een kleur aan te duiden. De gekozen kleur wordt bij het versturen van het formulier als hexadecimale waarde verzonden (#000000 - #ffffff).

### 1.1.1.20 COLOR

Indien we geen start waarde (*value* attribute) opgeven voor ons input type color zal deze standaard zwart kiezen (#000000). Onderstaand tonen we een input element van het type color waar we de startwaarde op groen plaatsten.

```
<label for="favorite-color">Favoriete Kleur: </label><br />
<input id="favorite-color" type="color" value="#00ff00">
```

Dit resulteert in onderstaande weergave in de browser.



#### 11.4.7 RADIOKNOPPEN EN CHECKBOXEN

We maken beide controls met het `input` element, afhankelijk van het type attribuut zoals hierboven beschreven. In dit geval kunnen we het `checked` attribuut gebruiken als het nodig is om ze standaard aangevinkt te laten verschijnen. `input` elementen van het type `checkbox` werken los van elkaar.

```
<input type="checkbox" name="checkbox1" value="A" checked="checked" />
Vinkje 1 met waarde A (standaard actief) <br />
<input type="checkbox" name="checkbox2" value="B" />
Vinkje 2 met waarde B <br />
<input type="checkbox" name="checkbox3" value="C" disabled="disabled" />
Vinkje 3 met waarde C (disabled)<br />
```

Bovenstaande code zal resulteren in onderstaand resultaat.

- Vinkje 1 met waarde A (standaard actief)
- Vinkje 2 met waarde B
- Vinkje 3 met waarde C(disabled)

Bij radioknoppen en checkboxen kunnen we verschillende waarden specifiëren. Als we `<input>` elementen van het type radio groeperen onder eenzelfde name attribute kunnen deze slechts 1 waarde hebben. Met andere woorden, bij de selectie van een andere radioknop in dezelfde groep zal de eerder geselecteerde waarde vervangen worden door de recent geselecteerde waarde.

```
<input type="radio" name="group-a" value="A" checked="checked"/>
radioknop met waarde A (standaard actief) <br />
<input type="radio" name="group-a" value="B" />
radioknop met waarde B <br />
<input type="radio" name="group-a" value="C" />
radioknop met waarde C <br />
```

| Standaard weergave   | Na selectie van de derde optie  |
|--|---|
| <input checked="" type="radio"/> radioknop met waarde A (standaard actief)<br><input type="radio"/> radioknop met waarde B<br><input type="radio"/> radioknop met waarde C | <input checked="" type="radio"/> radioknop met waarde A (standaard actief)<br><input type="radio"/> radioknop met waarde B<br><input checked="" type="radio"/> radioknop met waarde C |

#### 11.4.8 NEERKLAPMENU'S EN MENULIJSTEN

Nog een ander element dat we kunnen gebruiken, is het **select** element. Hiermee kunnen gebruikers een keuze maken uit een lange reeks opties. Het select element bevat child-elementen van het type **option** die elk een value hebben zodat een script de geselecteerde waarde kan bepalen.

Het **select** element wordt op twee mogelijke manieren weergegeven:

- Dropdownlist (neerklapmenu) – laat toe om slechts 1 element te selecteren en toont pas alle elementen nadat de gebruiker de control uitklapt.
- Listbox (menulijst) – toont een aantal van de mogelijk opties in een lijst, waar de gebruiker één of meerdere opties kan selecteren.

| Attribuut       | Beschrijving  |
|-----------------|---|
| <b>size</b>     | Indien hoger dan "1", dan krijgen we een listbox  |
| <b>multiple</b> | Wanneer we met een listbox werken, stelt dit attribuut de gebruiker in staat om meerdere opties te selecteren met behulp van de CTRL toets. |

```
<select name="country">
    <option value="BE">België</option>
    <option value="DE" selected="selected">Duitsland</option>
    <option value="UK">Engeland</option>
    <option value="FR">Frankrijk</option>
    <option value="LU">Groot Hertogdom Luxemburg</option>
    <option value="NL">Nederland</option>
</select>
```

Standaard is het **select** element een dropdownlist die enkel de huidig geselecteerde optie weergeeft. We kunnen er makkelijk een listbox van maken door het "size" attribuut op 2 of groter in te stellen; dit is het aantal zichtbare opties.

```
<select name="country" multiple="multiple" size="5">
    <option value="BE" selected="selected">België</option>
    <option value="DE" selected="selected">Duitsland</option>
    <option value="UK">Engeland</option>
    <option value="FR">Frankrijk</option>
    <option value="LU">Groot Hertogdom Luxemburg</option>
    <option value="NL">Nederland</option>
</select>
```

Er kunnen ook categorieën opgegeven worden voor de opties. Dit kan met het **optgroup** element en het attribuut **label**.

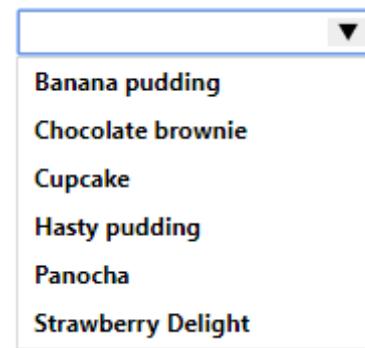
```
<select name="wood-selection">
  <optgroup label="Hardhout">
    <option value="EIK">Eik</option>
    <option value="BEU">Beuk</option>
    <option value="MER">Meranti</option>
  </optgroup>
  <optgroup label="Zachthout">
    <option value="CED">Ceder</option>
    <option value="DEN">Den</option>
  </optgroup>
</select>
```

#### 1.1.1.21 LIST ATTRIBUTE OP INPUT TYPE 'X'

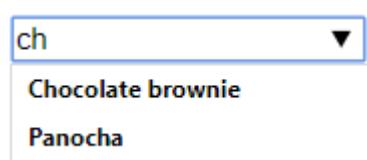
In HTML5 is het mogelijk om mogelijke opties aan de gebruiker voor te schotelen via **input** elementen.

Door gebruik te maken van een **list** attribute gekoppeld aan een **datalist** element dat zich in hetzelfde document bevindt, kan je de gebruikers een aantal opties geven. Het linken gebeurt door het **id** van het **datalist** element als value van het **list** attribute te gebruiken. We bekijken hiervan een voorbeeld:

```
<input type="text" list="desserts" />
<datalist id="desserts">
<option value="Banana pudding">
<option value="Chocolate brownie">
<!-- Verdere opties --&gt;
&lt;/datalist&gt;</pre>
```



Een mogelijk voordeel bij het gebruik van een **datalist** gekoppeld aan een input element van het type **text** is dat de gebruiker nog steeds ook zijn eigen ingave kan doen los van de keuzelijst. Bijkomend zal er door middel van wat de gebruiker reeds ingeeft in het input veld een filtering gemaakt moeten worden van de beschikbare opties.



**Let op!** Het list attribute wordt genegeerd op volgende types: hidden, checkbox, radio, file en button.

#### 11.4.9 VERBORGEN VELDEN

Deze verborgen velden worden meestal gegenereerd door de server die de pagina toegestuurd krijgt om bepaalde zaken te onthouden op het moment dat het formulier teruggezonden wordt door de browser. Ze worden nooit weergegeven en zijn niet manipuleerbaar door de gebruiker zelf, maar wel d.m.v. javascript. Verborgen velden hebben enkel een name en/of id en een value attribuut.

```
<input type="hidden" name="hidden-input" value="Hello Forms?" />
```

#### 11.4.10 BESTANDEN SELECTEREN

Het is vaak nuttig om bestanden te kunnen uploaden. Hiervoor heeft HTML een speciale control die het lokale besturingssysteem gebruikt om de gebruiker een bestand te laten selecteren. Zodra de form

verstuurd wordt, zal dit bestand meegestuurd worden als binaire data. Het spreekt voor zich dat het zeer lang kan duren wanneer iemand een zeer groot bestand uploadt. Om misbruik te voorkomen hebben de meeste servers een maximum bestandsgrootte ingesteld, die enkel aan de server kant ingesteld kan worden.

```
<input type="file" name="uploaded-file"/>
```

Er moet nog opgemerkt worden dat deze control **zéér** variabel is in zowel uiterlijk als functionaliteit per browser en per besturingssysteem. Safari zal bijvoorbeeld een *compleet* andere control tonen dan Firefox en IE. Bovendien is de controle van kleuren en posities met CSS zeer beperkt. Testen op verschillende browsers is dus andermaal de boodschap!

#### 11.4.11 KNOPPEN

Knoppen zijn wellicht de controls die het minst inspanning vragen van de gebruiker om iets gedaan te krijgen.

We onderscheiden vier soorten:

- **Submit** knoppen: versturen het formulier in een nieuwe HTTP Request naar de server, met de ingevulde data  
(wordt gestuurd naar de “action” URL met de HTTP methode “get” of post”)
- **Reset** knoppen: herstellen het formulier naar de staat waarin het oorspronkelijk toekwam in de browser (gebruikersinvoer is ongedaan gemaakt)
- **Neutrale** knoppen: doen in feite niets uit zichzelf maar zijn bedoeld om een bepaalde functie uit te voeren, bijvoorbeeld javascripts.
- **HTML** knoppen: hetzelfde als elke voorgaande knop, maar we kunnen er HTML code in stoppen om ze mooier te maken.

Submit, reset en neutrale knoppen maken we aan via verschillende instellingen van de “type” attribuut van de **input** tag. HTML knoppen hebben een eigen tag genaamd **button**. Allen worden ondersteund door de meest populaire browsers.

Een submit knop is de enige essentiële om het formulier te kunnen versturen. Hoewel een formulier ook via javascript verstuurd kan worden, dankzij “neutrale” knoppen, stellen we meestal een submit knop in om zonder extra code het formulier zijn taak te kunnen laten vervullen.

```
<input type="submit" value="Verstuur Formulier" />
```

We kunnen ook een figuur gebruiken als submit button, wanneer we het **type** attribuut instellen op “image”. Vergeet in dit geval niet om een **src** op te geven voor de figuur. Het resultaat van het onderstaand element ziet eruit als een figuur in een hyperlink, maar is in werkelijkheid een knop:

```
<input type="image" src="blue_question_mark.png" alt="Questionmark" title="Meer Info" />
```

Weinig te zien maar nog steeds geldig is de “reset” knop. Omwille van gebruikers die per ongeluk hun invoer wissen is deze knop in gebruik geraakt:

```
<input type="reset" value="Ongedaan maken" />
```

Een “neutrale” knop wordt ook gemaakt met een **input** element.

```
<input type="button" value="Klik en er gebeurt noppes" />
```

## 11.5 VELDEN ORGANISEREN

---

### 11.5.1 TABVOLGORDE

Gebruikers die vaak een formulier moeten invullen doen dit soms liever niet via de muis, maar met de tab toets. Op deze manier kan een gebruiker snel doorheen de verschillende velden nageren. Standaard is de tabvolgorde de volgorde waarin de elementen voorkomen in de pagina. Het kan zeer vervelend zijn wanneer deze volgorde niet strookt met de manier waarop de velden visueel verschijnen of gegroepeerd zijn.

Met behulp van het **tabindex** attribuut, dat vrijwel voor elk form-element voorkomt, kunnen we dit manipuleren: de kleinste getalwaarde voor tabindex wordt steeds eerst doorlopen.

### 11.5.2 SNELTOETSEN

Minder gebruikt maar toch van toepassing is tot slot de mogelijkheid om sneltoetsen in te stellen voor velden. Door het attribuut **accesskey** in te stellen op een knop (bvb “e”), kan de gebruiker door bvb. ALT+E in te drukken dit veld automatisch activeren. Het spreekt voor zich dat deze accesskeys uniek moeten zijn.

## 11.6 FORMULIEREN VERSTUREN

---

Om je oefeningen te testen, kan je gebruik maken van de pagina <https://ikorpela.fi/cgi-bin/echo.cgi>. Een echo script zal daar de verzonden data weergeven. Je kan dit script gebruiken in je **action** attribuut met zowel een **GET** als **POST** methode.

Bij wijze van test kunnen we echter ook een e-mail adres opgeven, waarna het formulier zichzelf zal proberen te mailen naar het opgegeven e-mail adres. Het is echter **zeer onverstandig** om dit te gebruiken op een echte website, omdat hiervan makkelijk misbruik gemaakt kan worden. Niettemin toch een voorbeeld:

```
<form method="post" action="mailto:mijnemail@mijnmailsite.com">
    <label for="email">email:</label><br />
    <input id="email" name="email" type="email" /><br />
    <label for="pass">paswoord:</label><br />
    <input id="pass" name="pass" type="password" /><br />
    <label for="txt-comment">Commentaar:</label> <br />
    <textarea id="txt-comment" name="txt-
comment" cols="50" rows="5"></textarea><br />
    <input type="submit" value="Versturen" />
</form>
```

Dit werkt enkel indien je method “post” is. De ontvanger krijgt in dat geval een nogal moeilijk leesbaar bericht met de geposte data uit het formulier. De gebruiker die het formulier stuurt, kan bovendien een waarschuwing krijgen van bepaalde antivirusprogramma’s dat de browser automatisch een mailbericht wil versturen.

Het gebruik van server-side scripts voor de verwerking van formulieren (zoals PHP, ASP.NET en CGI) biedt voor deze en heel wat andere zaken een oplossing





# Positionering van elementen

**Web Frontend Basics**

## INHOUD

|   |          |
|---|----------|
| <b>12 POSITIONERING VAN ELEMENTEN</b>         | <b>3</b> |
| <b>12.1 Width en height</b>                   | <b>3</b> |
| <b>12.2 Alternatieve positionering</b>        | <b>3</b> |
| <b>12.3 Verduidelijking van positionering</b> | <b>5</b> |
| 12.3.1 Position Relative                      | 7        |
| 12.3.2 Position Absolute                      | 8        |
| <b>12.4 Overflows</b>                         | <b>9</b> |

## 12 POSITIONERING VAN ELEMENTEN

Tot hiertoe volgden we hoofdzakelijk de standaard positionering die de browser voor onze elementen bepaalde. Dit aan de hand van het soort element, het type (*block of inline*) en eventuele aanpassingen op de margin property van het box model van dit element via CSS. We zijn echter ook in staat om onze elementen nog iets specifieker te gaan positioneren indien we hier nood aan hebben.

Er zijn een viertal manieren om een element te positioneren:

- We laten het element in zijn *flow*, dit is de default (*static*).
- We verplaatsen het element ten opzichte van zijn standaard positie (*relative*).
- Het element positioneren ten opzichte van het eerste, bovenliggende, parent element wat een positie anders dan static heeft (*absolute*).
- Het element positioneren via x-y coördinaten ten opzichte van het **browservenster** (*fixed*).

Zodra we weten waar het element geplaatst moet worden, kunnen we behalve de opmaak (*kleuren, fonts, alignering...*) ook de layout verder beïnvloeden door de marges (*margin*), opvulling (*padding*), de rand (*border*), breedte (*width*), hoogte (*height*), ... te bepalen. Hou er echter rekening mee dat bepaalde eenheden (*zoals %*) beïnvloed worden door het voorliggende element (*parent*).

### 12.1 WIDTH EN HEIGHT

Voor we aan de slag gaan met het positioneren van onze elementen staan we eerst nog even stil bij twee belangrijke CSS properties die de uiteindelijke positionering van een element ook beïnvloeden.

Op de meeste elementen kunnen we de **width** en **height** instellen (*figuren, form-elementen, tekstblokken, ...*) via de gelijknamige properties. De meest gebruikte eenheden voor deze eigenschappen zijn **%** en **px**. We kunnen ook **auto** als waarde opgeven. In dat geval berekent de browser de afmetingen zelf (*afhankelijk van de flow, andere elementen*); **auto** is dus meteen ook de standaardwaarde van beide eigenschappen.

Enkele bijzonderheden aan **width** en **height**:

- Percentages als eenheid zijn steeds in verhouding met de hoogte/breedte van het bovenliggende (parent) element.
- Width en height worden niet overgeërfd
- Width en height worden ingesteld voor de **inhoud** van de box, dus border, padding en margin zijn hier niet bijgerekend

### 12.2 ALTERNATIEVE POSITIONERING

Zoals hierboven vermeld, bestaat er ook een *property* genaamd **position** waarmee we nog een stap verder kunnen gaan dan layout met margins, floats en dergelijke. Met position kunnen we een element uit zijn

normale positie halen en deze ergens elders neerplanten. Deze eigenschap gebruik je in principe enkel wanneer je een iets complexere layout wil maken.

Zodra je **position** op een andere waarde dan *static* instelt, zal je ook de positie moeten opgeven met de stijleigenschappen **top**, **right**, **bottom** en **left** die vervolgens een eenheid bevatten (*px*, *%*, ...). Deze waarden worden ook wel de **offset** genoemd:

| CSS Property  | Beschrijving  |
|---------------|---|
| <b>top</b>    | Afstand tussen de <i>top</i> van het element en de <i>top</i> van zijn referentiepunt                   |
| <b>right</b>  | Afstand tussen de <i>rechterzijde</i> van het element en de <i>rechterzijde</i> van zijn referentiepunt |
| <b>bottom</b> | Afstand tussen de <i>onderkant</i> van het element en de <i>onderkant</i> van zijn referentiepunt       |
| <b>left</b>   | Afstand tussen de <i>linkerzijde</i> van het element en de <i>linkerzijde</i> van zijn referentiepunt   |

Nu we weten hoe de positie in te stellen moet de **position** eigenschap het referentiepunt voor de nieuwe positie bepalen:

| CSS Property    | Waarde          | Beschrijving waarde   |
|-----------------|-----------------|---|
| <b>position</b> | <b>static</b>   | Standaardwaarde: het element behoudt zijn normale positie in de flow en <b>negeert</b> "top, right, bottom en left" instellingen.           |
|                 | <b>relative</b> | Positioneert het element in relatie tot zijn normale positie in de flow.  |
|                 | <b>absolute</b> | Speciaal geval: Positioneert het element ten opzichte van het eerste, bovenliggende parent element dat een positie anders dan static heeft. |
|                 | <b>fixed</b>    | Bevestigt het element aan het <b>browservenster</b> : het element is niet langer onderhevig aan scrollen.                                   |

## 12.3 VERDUIDELIJKING VAN POSITIONERING

---

Aan de hand van een aantal voorbeelden proberen we in dit deel van de cursus bovenstaande informatie iets duidelijker te maken.

We starten met volgende HTML code in ons **body** element van onze pagina.

```
<header>
  <p>Onze header</p>
</header>
<section id="section-01">
  <p>Onze Section</p>
  <article id="article-01">
    <p>Ons eerste artikel</p>
    <p>
      Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque e
      uismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac.
    </p>
  </article>
  <article id="article-02">
    <p>Ons tweede artikel</p>
    <p>
      Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque e
      uismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac.
    </p>
  </article>
  <article id="article-03">
    <p>Ons derde artikel</p>
    <p>
      Tincidunt integer eu augue augue nunc elit dolor, luctus placerat scelerisque e
      uismod, iaculis eu lacus nunc mi elit, vehicula ut laoreet ac.
    </p>
  </article>
</section>
<footer>
  <p>Onze footer</p>
</footer>
```

Startsituatie HTML

We koppelen vervolgens een stylesheet aan onze pagina en plaatsen onderstaande CSS syntax in deze stylesheet.

```
body{  
    background-color: darkgrey;  
    margin:200px 20px 0 100px;  
}  
  
#section-01 {  
    background-color: black;  
    padding: 15px;  
    color: white;  
}  
  
#article-01 {  
    background-color: orangered;  
}  
  
#article-02 {  
    background-color: forestgreen;  
}  
  
#article-03 {  
    background-color: blueviolet;  
}  
  
header {  
    background-color: cornflowerblue;  
}  
  
p {  
    margin: 0;  
}  
  
footer {  
    background-color: cornflowerblue;  
}
```

Basis CSS

Op deze manier hebben we alvast een visualisatie van onze elementen in de browser. Hierdoor zullen we tijdens onze uitwerking prima kunnen zien hoe deze elementen zich gedragen bij het positioneren. Zie onderstaand voorbeeld.



### 12.3.1 POSITION RELATIVE

Laten we eerst even kijken wat de value **relative** met een element zal doen. Hiervoor passen we nog even onze CSS selector **#section01** aan. En voegen er volgende syntax aan toe.

```
#section-01 {  
    background-color: black;  
    padding: 15px;  
    color: white;  
    position: relative;  
    top: 20px;  
    left: -40px;  
}
```

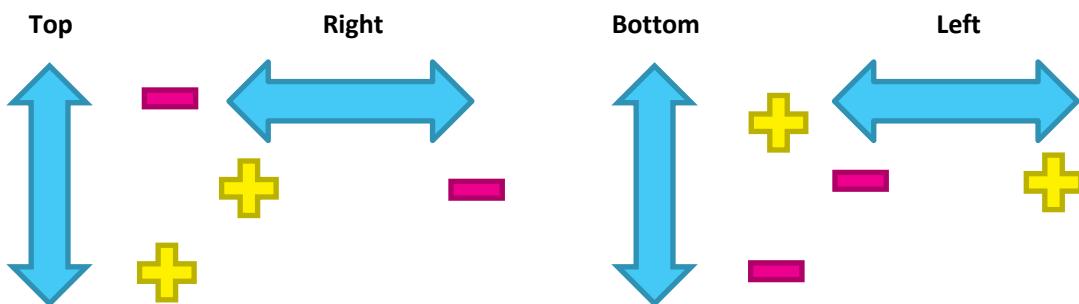
Aanpassing op de CSS selector #section01

We bekijken het resultaat in onze browser.



We zien dat het element met id **section01** verschoven is **ten opzichte van zijn originele positie**. En dit namelijk met *20 pixels* van de bovenzijde en *40 pixels* naar links. Door middel van het gebruik van de properties **top**, **right**, **bottom** en **left** kunnen we dus onze element met een zekere precisie een plaats gaan geven op onze pagina.

Hieronder vind je een overzicht van hoe een element zich verplaatst bij positieve of negatieve waarden op de **top**, **right**, **bottom** en **left** properties.



### 12.3.2 POSITION ABSOLUTE

We gaan verder met de code die we tot hertoe hebben om vervolgens de **absolute** value van de **position** property te verduidelijken.

Hier voor plaatsen we eerst even de drie regels CSS syntax die we toevoegden op onze **#section01** selector terug in commentaar.

```
#section-01 {  
    background-color: black;  
    padding: 15px;  
    color: white;  
    /* position: relative;  
    top: -20px;  
    right: -40px; */  
}
```

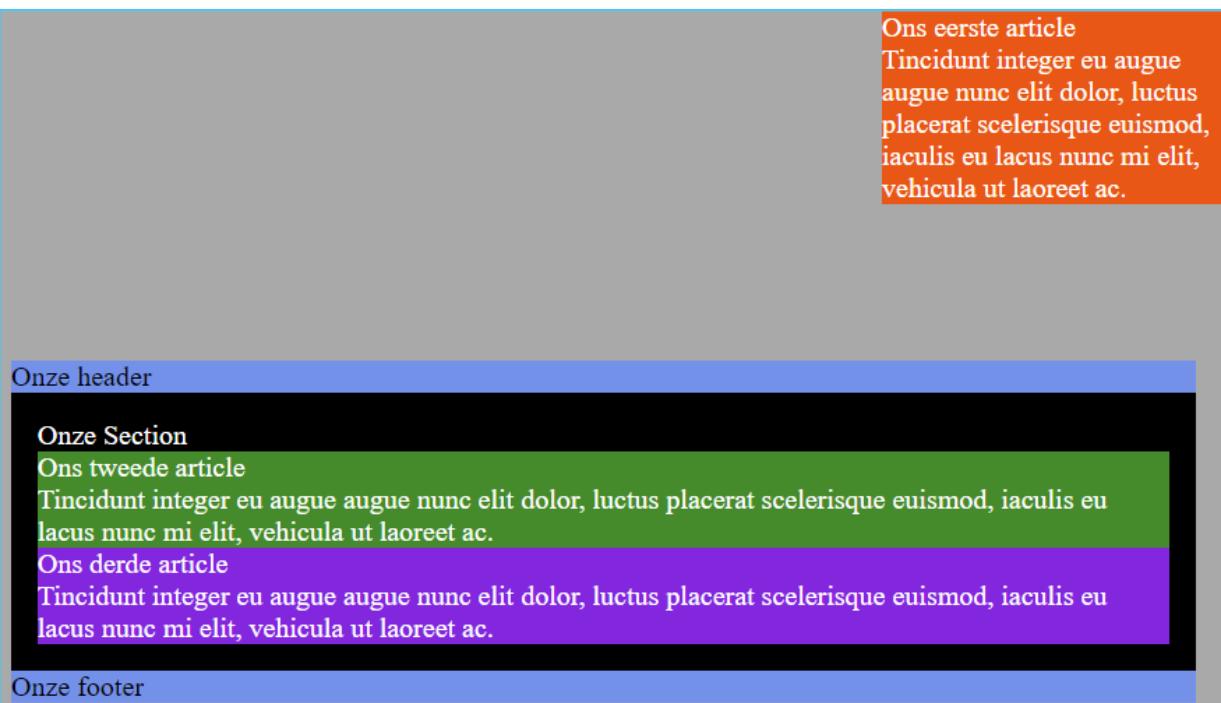
Aanpassing op de CSS selector

Vervolgens voegen we volgende regels CSS toe aan onze **#article01** selector.

```
#article-01 {  
    background-color: orangered;  
    position: absolute;  
    top: 0;  
    right: 0;  
    width: 200px;  
}
```

Toevoeging op de **#article01** selector

Als we de pagina na de aanpassing in een browser bekijken, zien we onderstaand resultaat.



We zien dus dat met de toepassing van **position : absolute** ons element met id **article01** helemaal in de rechter bovenhoek van ons browservenster geplaatst wordt. Hoe komt dit nu?

Wel, als we gebruik maken van de **absolute** waarde op een element, dan gaat de browser op zoek naar het allereerste parent element van dat element dat een position waarde anders dan **static** heeft.

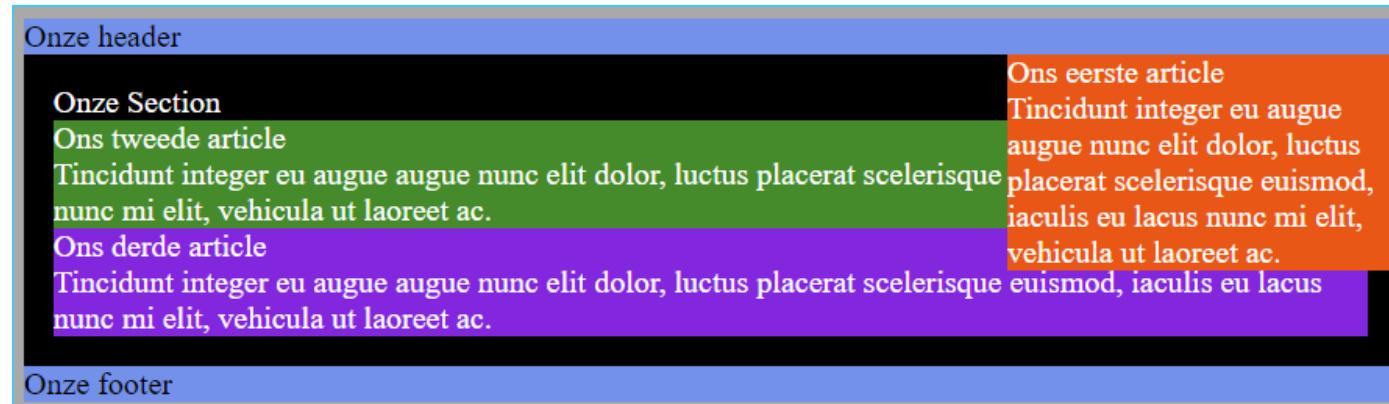
In ons huidige voorbeeld is er geen enkel parent element van het element met id **article01** dat aan deze voorwaarde voldoet en de browser eindigt vervolgens zijn zoektocht bij het body element. Daarom zien we ons **article01** element helemaal in de hoek bovenaan ons scherm kleven.

Laten we even met een kleine aanpassing het parent element met id **section01** terug voorzien van de property **position** met de waarde **relative**.

```
position: relative;
```

Toevoegen aan de `#section01` selector.

Als we nu even teruggaan naar onze browser zien we dat ons **article01** element niet langer in de bovenhoek van de browser plaats zal nemen maar wel in de bovenhoek van het **section01** element. De browser vindt deze keer dus wel een parent element met een position relative en zal vervolgens de zoektocht staken en hier ons element een plaats gaan geven.



## 12.4 OVERFLOWS

Elementen of tekst bevinden zich niet altijd binnen hun box. Dit klinkt vreemd, maar er zijn tal van redenen mogelijk: misschien is de grootte om zijn inhoud te kunnen bevatten te klein, misschien heb je de inhoud buiten de box gepositioneerd met negatieve offsets of marges. Het maakt eigenlijk niet uit, want we kunnen met de eigenschap **overflow** bepalen **hoe een box zich moet gedragen** wanneer zich dit voordoet.

| CSS Property    | Waarde         | Beschrijving waarde   |
|-----------------|----------------|---|
| <b>overflow</b> | <b>visible</b> | Dit is de standaardwaarde: de box zal zich automatisch proberen aan te passen aan de inhoud ervan. (denk maar aan een div die zijn hoogte aanpast aan het aantal regels tekst die erin moet komen) Er zullen nooit scrollbars getoond worden. |
|                 | <b>auto</b>    | Zodra de inhoud niet past in de ingestelde hoogte of breedte worden er scrollbars aangemaakt in de box. Die stellen de gebruiker in staat om verborgen inhoud te lezen. Indien de inhoud past, dan zijn deze scrollbars verborgen.            |

|               |  |
|---------------|--|
| <b>scroll</b> | Er worden scrollbars voorzien zodat de gebruiker verborgen inhoud kan lezen, onafhankelijk van een passende boxinhoud.             |
| <b>hidden</b> | Zodra de inhoud niet past in de ingestelde hoogte of breedte van een box zal deze verborgen blijven, onleesbaar voor de gebruiker. |





# Meta elementen

Web Frontend Basics

## INHOUD

|  |          |
|--|----------|
| <b>13 META ELEMENTEN ONDER DE LOEP</b>   | <b>3</b> |
| <b>13.1 Wat zijn meta tags</b>           | <b>3</b> |
| <b>13.2 Attributen in meta tags</b>      | <b>3</b> |
| 13.2.1 Charset                           | 3        |
| 13.2.2 Name                              | 3        |
| 13.2.3 Content                           | 4        |
| <b>13.3 Values in het name attribuut</b> | <b>4</b> |
| 13.3.1 Application-name                  | 4        |
| 13.3.2 Description                       | 4        |
| 13.3.3 Keywords                          | 5        |
| 13.3.4 Viewport                          | 5        |
| 13.3.5 Robots                            | 6        |
| 13.3.6 Geo                               | 6        |
| <b>13.4 Meta tags en sociale media</b>   | <b>7</b> |
| 13.4.1 Open Graph protocol               | 8        |
| <b>13.5 Basic OG meta data</b>           | <b>8</b> |
| 1.1.1.1 Title                            | 8        |
| 1.1.1.2 Type                             | 8        |
| 1.1.1.3 Url                              | 8        |
| 1.1.1.4 Image                            | 9        |
| <b>13.6 Optional OG meta data</b>        | <b>9</b> |
| <b>13.7 Debuggen van OG meta data</b>    | <b>9</b> |

## 13 META ELEMENTEN ONDER DE LOEP

### 13.1 WAT ZIJN META TAGS

Een metatag is een element dat specifieke informatie bevat en zich altijd bevindt tussen het head element. Een document kan en zal in de meeste gevallen meerdere meta elementen bevatten.

```
<head>
    <title>Titel van de pagina</title>
    <meta charset="utf-8" />
</head>
```

We zien hierboven al een eerste voorbeeld van hoe een meta element eruit ziet. Zoals je kan zien is dit meta element zelfsluitend. Sinds HTML5 is het echter niet meer noodzakelijk dat een metatag zelfsluitend is. De slash '/' mag dus weggelaten worden.

Meta tags zijn héél flexibel en kunnen een breed gamma aan informatie over het document bevatten met behulp van verschillende attributen.

Alhoewel meta tags optioneel zijn, kunnen ze je site optimaliseren voor zowel browsers, zoekmachines, sociale media platformen en verdere informatie-uitwisseling tussen systemen.

Er zijn dus héél wat verschillende meta tags met elk hun eigen doel of functie.

### 13.2 ATTRIBUTEN IN META TAGS

Attributen zijn ook in meta tags essentieel om te gebruiken. Anders is de functie van de metatag niet duidelijk. Hieronder lijsten we enkele van de meest gebruikte attributen op.

#### 13.2.1 CHARSET

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

Via het charset attribuut maken we duidelijk aan de browser dat deze website gebruik maakt van de UTF-8 karakterset. Hierdoor zal iemand die surft naar onze pagina met bvb. een Aziatische browserinstelling de karakters op het scherm zien zoals wij ze kunnen lezen.

Deze metatag met dit attribuut plaats je best ook bovenaan je webpagina (in je head!) om zo snel mogelijk door te geven aan de browser in welke karakterset de website getoond moet worden.

#### 13.2.2 NAME

```
<head>
    <meta name="author" content="John Doe" />
</head>
```

Via het name attribute kunnen we metadata specificeren door middel van de waarde (value) van het meta element. In dit voorbeeld geven we de auteur op van de webpagina, blogpost, ...

### 13.2.3 CONTENT

```
<head>
  <meta name="author" content="John Doe" />
</head>
```

Het content attribuut is steeds verplicht wanneer we een name-attribuut gebruiken. Via het content attribuut geven we namelijk de invulling mee van de waarde van het name attribuut.

In bovenstaand voorbeeld wordt in content de naam van de auteur als value geschreven. In het name attribuut geven we mee dat de content value de naam van de auteur zal bevatten.

## 13.3 VALUES IN HET NAME ATTRIBUUT

---

Het name attribuut kan verschillende waarden (values) bevatten. Alle values bespreken die een name attribuut kan bevatten is onbegonnen werk. Toch zullen we enkele van de meest voorkomende bespreken.

### 13.3.1 APPLICATION-NAME

```
<head>
  <meta name="application-name" content="Howest cursus app" />
</head>
```

Specificeert de naam van de web applicatie die de pagina voorstelt.

### 13.3.2 DESCRIPTION

```
<head>
  <meta name="description" content="Well organized and easy to understand Web
building tutorials with lots of examples of how to use HTML, CSS, JavaScript,
SQL, PHP, Python, Bootstrap, Java and XML." />
</head>
```

De description value is er éénje die je best niet vergeet als je gevonden wil worden door de zoekmachines. Hierin kan je een korte beschrijving geven over het onderwerp van je pagina. De value van deze content pas je best aan per pagina op je website. Elke pagina in je website zal immers over een ander onderwerp gaan.

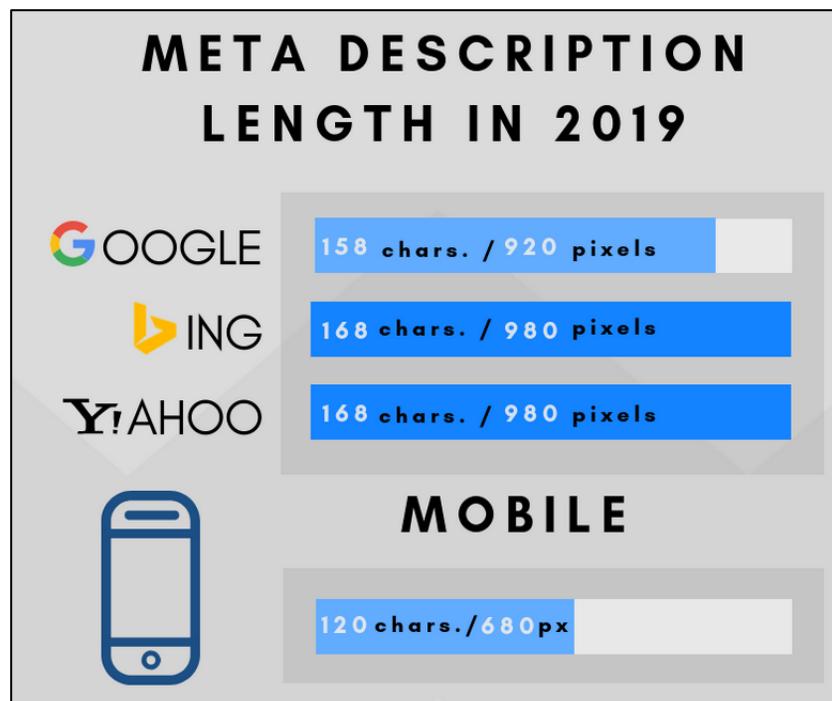
De value van deze content wordt getoond in de zoekresultaten van de zoekmachines.

### Howest, de Hogeschool West-Vlaanderen: we develop people!

<https://www.howest.be> › ... ▾

Howest is een innovatieve en ondernemende hogeschool met 23 future proof bachelors en 6 graduaten, waaronder een aantal unieke voor Vlaanderen.

Het aantal karakters is hier in principe onbeperkt, al wordt aangeraden in 2019 om deze tussen de 120 en 158 karakters te houden.



#### 13.3.3 KEYWORDS

```
<head>
  <meta name="keywords" content="HTML, meta tag, content, name" />
</head>
```

In de content van de keywords value kan je verschillende woorden opgeven, gescheiden door een komma. Hierdoor kunnen zoekmachines zien over welke onderwerpen je webpagina gaat. Al is deze value minder relevant geworden voor zoekmachines, de description value krijgt een hoger gewicht toegekend in het bepalen van de zoekresultaten dan de keywords value.

#### 13.3.4 VIEWPORT

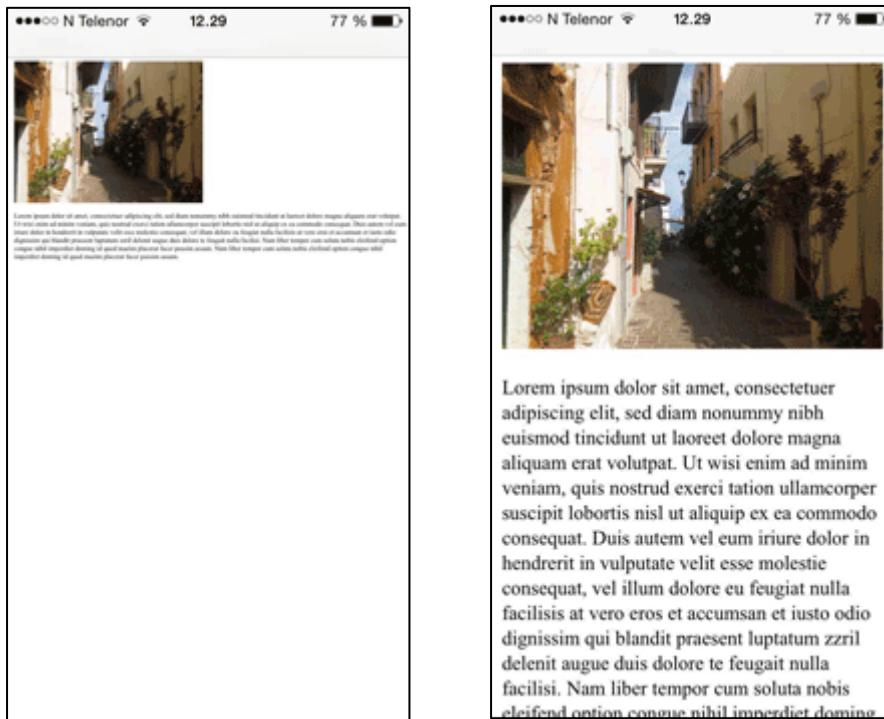
```
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
</head>
```

Controleert de viewport (de zichtbare ruimte van een gebruiker van een webpagina). Aangezien we te maken hebben met verschillende schermformaten is het belangrijk om deze altijd te voorzien. Deze metatag geeft instructies aan de browser hoe die de dimensies en schaling moet controleren.

De **width=device-width** value zorgt ervoor dat de breedte van de webpagina de schermbreedte volgt van het toestel (laptop, smartphone, tablet, ...) waarop de website bekeken wordt.

De **initial-scale=1.0** value stelt het initiële zoomlevel in wanneer de pagina voor het eerst geladen wordt.

Een voorbeeld zonder en met de viewport meta tag op een mobiel toestel:



### 13.3.5 ROBOTS

Wanneer je deze niet voorziet, zullen alle zoekmachines je website opnemen om te indexeren. Wanneer dit gebeurd is, kan je gevonden worden door de verschillende zoekmachines.

```
<head>
  <meta name="robots" content="noindex" />
</head>
```

Mocht je toch één of meerdere webpagina's hebben die je niet wenst op te laten nemen in de zoekresultaten, dan kan je gebruik maken van deze meta tag. Doormiddel van de noindex geef je aan dat zoekmachines deze pagina moeten negeren en dus niet mogen opnemen in de zoekresultaten.

### 13.3.6 GEO

```
<head>
  <meta name="geo.position" content="latitude; longitude" />
  <meta name="geo.placename" content="Plaats naam" />
  <meta name="geo.region" content="Landcode" />

</head>
```

Met deze meta tags kun je aangeven waar het bedrijf, persoon, ... zich situeert. Anno 2019 wordt er door Google niet gekeken naar deze meta tags. De Bing zoekmachine dan weer wel.

## 13.4 META TAGS EN SOCIALE MEDIA

Meta tags beginnen een belangrijke rol te spelen wanneer je websites wenst te delen via sociale media, WhatsApp, iMessage, Telegram, ...

Wanneer je een link deelt via sociale media zal er altijd een afbeelding getoond worden met de titel van de pagina waar de link naartoe leidt. Bekijk onderstaand voorbeeld. Men wenst de website van Howest te delen op Facebook. Wanneer de link geplakt wordt in het berichtenvenster zal er onmiddellijk een preview getoond worden van de webpagina waarnaar gelinkt wordt.



Alle info die we zien in deze preview wordt uit de meta tags uitgelezen:

- **Titel** (Howest, de Hogeschool West-Vlaanderen: we develop people!)
- **Description** (Howest is een innovatieve en ondernemende hogeschool met 23 future...)
- **Afbeelding** (foto met Howest vlaggen)

Via de meta tags kunnen we dus ook kiezen welke foto we wensen te tonen bij welke pagina. Hiervoor maken we gebruik van "**The Open Graph protocol**".

### 13.4.1 OPEN GRAPH PROTOCOL

Het Open Graph protocol wil er voor zorgen dat alle meta data op een open standaard manier gelezen kan worden. Dit protocol zorgt er ook voor dat er extra data toegevoegd kan worden bovenop de standaard meta tags. Denk hierbij aan afbeeldingen, audio, video, ...

Sociale media platformen en chat services zoals Facebook, LinkedIn, WhatsApp, iMessage, ... maken gebruik van de Open Graph meta tags om previews te genereren van webpagina's gedeeld op hun platform.

Meer info hierover is te vinden op de website van Open Graph: <https://ogp.me/>

We zullen enkele basis meta data elementen bespreken. Deze meta tags gebruiken echter niet het name attribuut, maar wel het **property** attribuut, al is de functionaliteit dezelfde. De value van de property wordt altijd voorafgegaan door de prefix "og:".

## 13.5 BASIC OG META DATA

---

ag

### 1.1.1.1 TITLE

De naam zegt het zelf, hier geven we de titel van de pagina mee via Open Graph meta tags.

### 1.1.1.2 TYPE

Hiermee geven we het type aan waarover de pagina gaat. In dit voorbeeld gaat het over de film "The Rock". We geven dus mee via **og:type** dat het over een film gaat via de content value **video.movie**. Andere mogelijke values zijn video.actor, music.song, music.artist, ... kortom er zijn er héél wat.

Hier vindt je een lijst met alle Open Graph types: <https://ogp.me/#types>

### 1.1.1.3 URL

Hiermee geven we de **canonical URL** mee van de webpagina. In de meeste gevallen is de eigenlijke url van de pagina ook de canonical url. Wanneer je echter in een situatie komt waarbij je webpagina via twee verschillende urls te bereiken is, moet je gebruik maken van de canonical url meta tag. Hiermee geef je aan dat de canonieke URL de hoofd-url is van deze webpagina. Ook al is die webpagina via twee of meerdere urls te benaderen.

Voorbeeld van vier verschillende urls die verwijzen naar dezelfde pagina (een productpagina van een HDMI kabel):

- [www.webshop.be/hdmi-kabel](http://www.webshop.be/hdmi-kabel)
- [www.webshop.be/hdmi-kabels/hdmi-kabel](http://www.webshop.be/hdmi-kabels/hdmi-kabel)
- [www.webshop.be/computeraccessoires/hdmi-kabel](http://www.webshop.be/computeraccessoires/hdmi-kabel)
- [www.webshop.be/televisieaccessoires/hdmi-kabel](http://www.webshop.be/televisieaccessoires/hdmi-kabel)

In de meta tag geven we dan als canonical url de hoofd-URL mee; bijvoorbeeld [www.webshop.be/hdmi-kabel](http://www.webshop.be/hdmi-kabel). Hierdoor weet Google (of een andere zoekmachine) wat de hoofdpagina is.

Wordt de canonical url niet meegegeven, dan zullen de zoekmachines bovenstaande urls als vier verschillende webpagina's zien. Ook al verwijzen ze allemaal naar dezelfde pagina. Je positie in de zoekresultaten zal ook afgestraft worden, omdat zoekmachines er van uitgaan dat je vier verschillende pagina's hebt met telkens dezelfde inhoud. Zoekmachines interpreteren dit als duplicate content.

#### 1.1.1.4 IMAGE

Belangrijk om niet te vergeten. Via deze meta tag geven we mee welke afbeelding getoond mag worden. Dit kan een afbeelding zijn die niet getoond wordt op de webpagina zelf, maar die je wel wenst weer te geven wanneer iemand deze pagina deelt via sociale media, chat clients, ... Wanneer we kijken naar het voorbeeld van [www.howest.be](http://www.howest.be) en deze delen op bijvoorbeeld facebook zien we de foto van de vlaggen wanneer we dit wensen te delen. Als we op de webpagina zelf gaan kijken, zien we deze foto niet op de website staan. Deze foto wordt dus uitgelezen uit de **og:image** url.

Vul hier echter altijd het absolute pad in! Het zijn externe clients (Facebook, LinkedIn, WhatsApp, ...) die deze afbeeldingurl gaan opvragen en de afbeelding zullen vertonen. Dus niet vanuit onze server zelf.

### 13.6 OPTIONAL OG META DATA

---

Er zijn nog gigantisch veel andere Open Graph meta data. Deze kan je hier terugvinden: <https://ogp.me/#optional>

### 13.7 DEBUGGEN VAN OG META DATA

---

Soms wil je eens testen of je aangepaste OG meta tags wel werken, of dat de aanpassingen wel werken. Hiervoor kan je gebruik maken van de Facebook Debugger. (<https://developers.facebook.com/tools/debug/sharing/>)

In het voorbeeld van [www.howest.be](http://www.howest.be) zou dit volgende url genereren: <https://developers.facebook.com/tools/debug/sharing/?q=www.howest.be>

We zien dan ook welke OG meta data er uitgelezen werd:

| Based on the raw tags, we constructed the following Open Graph properties |   |
|---|---|
| og:url  | <a href="https://www.howest.be/nl">https://www.howest.be/nl</a>   |
| og:type   | website   |
| og:title  | Howest, de Hogeschool West-Vlaanderen: we develop people!   |
| og:image  | <a href="https://www.howest.be/sites/default/files/vlaggenRSS.jpg">https://www.howest.be/sites/default/files/vlaggenRSS.jpg</a>                 |
| og:description  | Howest is een innovatieve en ondernemende hogeschool met 23 future proof bachelors en 6 graduaten, waaronder een aantal unieke voor Vlaanderen. |
| og:updated_time   | 1570213141  |

Mochten we wijzigingen aangebracht hebben aan een eigen pagina en we zien de wijzigingen hier nog niet in terecht komen, kunnen we klikken op de knop “Scrape again”. Dan wordt de cache van Facebook ververst door de recentste versie.

**When and how we last scraped the URL**

Time Scraped      September 15 at 11:21 PM      [Scrape Again](#)

Response Code      206

Fetched URL      <http://www.howest.be/>





# Responsive design

Web Frontend Basics

## INHOUD

|  |          |
|--|----------|
| <b>14 RESPONSIVE DESIGN</b>            | <b>3</b> |
| <b>14.1 Inleiding</b>                  | <b>3</b> |
| 14.1.1 Media                           | 3        |
| 14.1.2 Responsifixuidaptive?           | 3        |
| 14.1.3 Mobile-first en andere patterns | 4        |
| <b>14.2 ViewPort</b>                   | <b>5</b> |
| 14.2.1 Viewports en mobiele schermen   | 5        |
| 14.2.2 Viewport meta-element           | 5        |
| <b>14.3 Media queries</b>              | <b>6</b> |
| <b>14.4 Wat kies ik best ?</b>         | <b>7</b> |

## 14 RESPONSIVE DESIGN

### 14.1 INLEIDING

Wanneer men in het begin van de jaren '10 het succes van mobile surfers zag, moest het web plots rekening houden met de veel kleinere schermpjes waarmee de gebruikers op hun GSM of tablet webpagina's wurdmen. Net toen men besefte dat resoluties als 640\*480 tot het verleden behoorden kwam men met nog kleinere aandraven. Veel gemaakte websites waren daar niet klaar voor en er werden 2<sup>de</sup> websites gemaakt (vaak voorafgegaan door een 'm' van mobile).

Deze 'tijdelijke' oplossing bleek na verloop van tijd ononderhoudbaar. Het is beter 1 website te maken die een volledige reeks van schermpjes ondersteunt. De tijden van **fixed** en **fluid** layouts kwamen op hun eind, entered **adaptive** en **responsive** layouts.

#### 14.1.1 MEDIA

Het populairste **medium** dat gebruikt wordt om een webpagina te presenteren, is uiteraard een scherm. In onze CSS code kunnen we regels vastleggen die gekoppeld zijn aan een medium, de zogenaamde **media queries**. Momenteel bestaan er 3 verschillende media :

| Medium | Betekenis  |
|--------|--|
| screen | Een scherm, dat zowel een scherm van een desktop, laptop, tablet, televisie, projector of smartphone kan zijn  |
| print  | Als een pagina wordt afgedrukt, dan wordt de webpagina ingelezen met het "print" medium. Hiermee maak je ondermeer (moderne) donkere achtergronden wit om inkt te besparen |
| speech | Gebruik voor screenreaders om een tekst voor te lezen  |

#### 14.1.2 RESPONSIFIXUIDAPTIVE?

Als je op zoek gaat naar onderwerpen over webdesign, dan wordt heel wat terminologie gebruikt om aan te geven hoe een pagina structureel is opgebouwd. Heel vaak worden deze termen door elkaar gehaald, een beetje zoals bovenstaande titel. We sommen de verschillende layout-smaken even op :

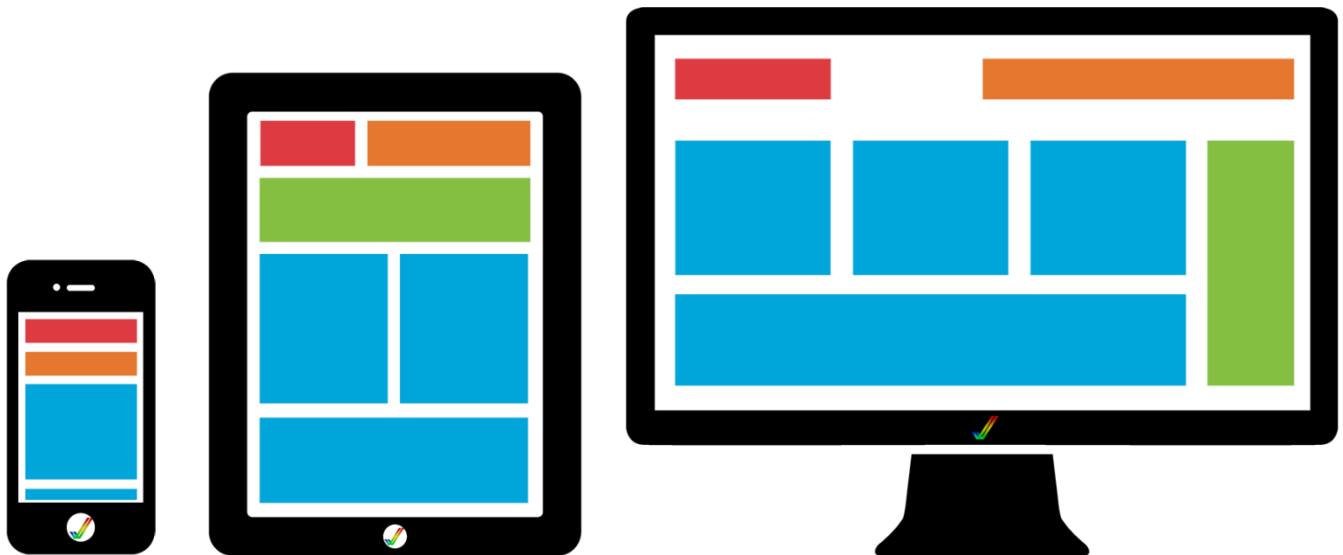
| Term       | Betekenis   |
|------------|---|
| Fixed      | De webpagina of een onderdeel ervan heeft een vaste grootte (vaak in pixels uitgedrukt). Bij het herschalen blijft het uitzicht van deze onveranderd, onafgezien het scherm een lage of hoge resolutie heeft.   |
| Fluid      | De grootte van een element wordt voornamelijk bepaald door percentages. Hierdoor schaalt de inhoud van de pagina mee in het browservenster die op verschillende resoluties geschaald wordt.   |
| Adaptive   | Deze pagina's of onderdelen ervan maken gebruik van media queries om te onderzoeken welke resolutie het browservenster exact heeft. Wanneer een bepaalde breedte wordt bereikt, dan past de inhoud zich daarop aan. Er kan gebruik gemaakt worden van een raster zodat de individuele blokken inhoud steeds gelijkaardige afmetingen krijgen. |
| Responsive | Deze pagina's gebruiken ook media queries en passen hun inhoud aan op een vloeiende manier.   |

### 14.1.3 MOBILE-FIRST EN ANDERE PATTERNS

Bij zowel adaptive als responsive design wordt er vaak gewerkt met de mobile-first strategie. Stel dat we een nieuw webdesign maken, dan gaat dit als volgt :

- De website wordt ontworpen voor de kleinere (mobile) resoluties, met inhouden, afbeeldingen, enzovoort.
- Er worden media queries toegevoegd om schermresoluties te detecteren.
- In deze media queries worden er stijlregels toegevoegd om de inhoud aan te passen aan grotere resoluties.

Het is vaak zo dat sommige functionaliteiten van een website verborgen blijven voor mobile, maar zichtbaar worden bij grotere schermen. Mobile first vertrekt dus van een minimum en groeit naarmate je grotere resoluties ondersteunt.



#### Meer weten

Volgende links bevatten voorbeelden, artikels en frameworks omtrent responsive / adaptive design :

- <http://bradfrost.github.io/this-is-responsive>  
Deze website bevat heel wat voorbeelden rond responsive/adaptive design en bespreekt per voorbeeld de voor- en nadelen van de gekozen aanpak. Leerrijk als je van plan bent een nieuw design aan te maken of een ontwerp aan te passen.
- <http://getbootstrap.com>  
Wellicht het meest populaire mobile-first framework dat een heel arsenaal aan vooraf gedefinieerde CSS en javascript bevat om snel aan de slag te kunnen gaan met een nieuw ontwerp. Nadeel is dat heel wat designers de standaard stijlen niet (kunnen) wijzigen waardoor heel veel websites op elkaar beginnen lijken. Een 2<sup>de</sup> mogelijk nadeel is dat Bootstrap werkt met jQuery, een javascriptbibliotheek die heden ten dage nogal veel problemen geeft bij het werken met Backend technologien (react.js ...)

## 14.2 VIEWPORT

### 14.2.1 VIEWPORTS EN MOBIELE SCHERMEN

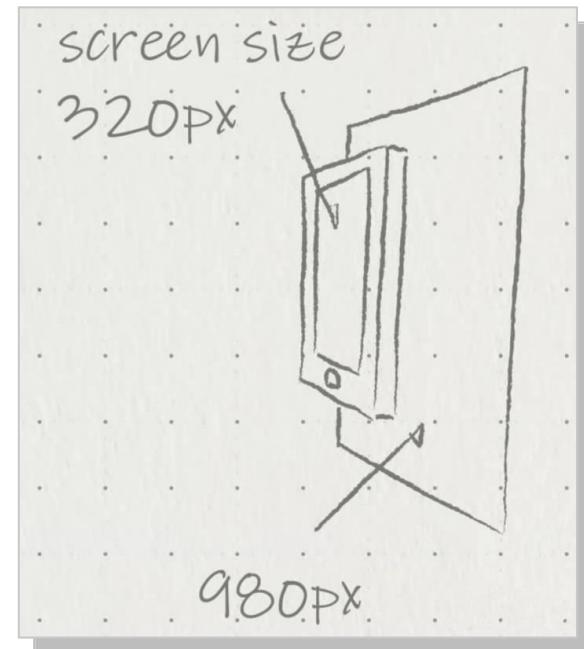
Een viewport is het gedeelte van het scherm dat gebruikt wordt voor de weergave van de inhoud van je pagina, dus alles wat je tussen de `<body>` tags plaatst.

Op een desktop apparaat kan je de viewport eenvoudig wijzigen door het browservenster kleiner of groter te maken. Bij een mobiel apparaat kan dat niet; een browservenster is steeds full screen.

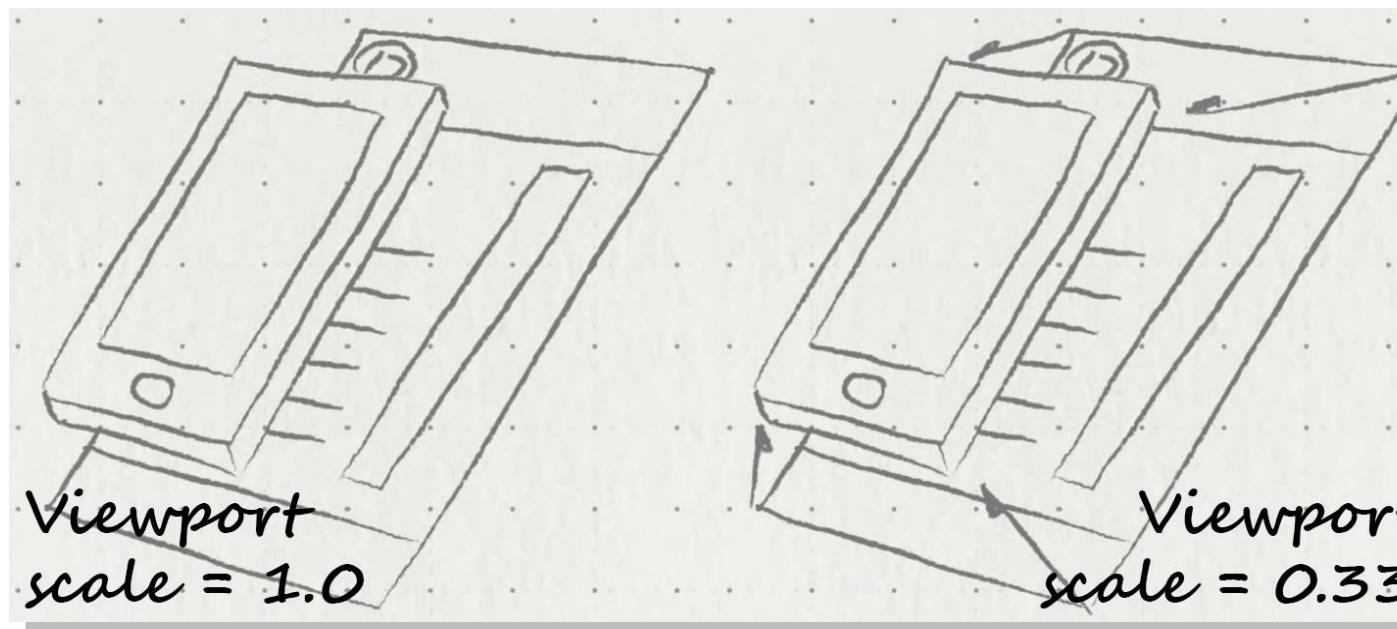
Stel dat we een webdesign maken met een **fixed width** van 1024px en we geven deze weer op een mobiel apparaat met de volgende kenmerken:

- Schermbreedte: 320px (=device-width)
- Viewport: 980px (=width)

Een mobile viewport is meestal groter dan de eigenlijk resolutie van de hardware die je gebruikt zodat je websites kan ondersteunen die voor een grotere resolutie zijn ontworpen. De meeste mobiele browsers hebben een viewport van 960px.



In dit geval zal een mobiele browser zijn viewport **automatisch herschalen** naar 320px zodat de volledige pagina op het scherm past. Da's handig, want anders zou de gebruiker naar links of rechts moeten pannen om de linker en rechterkant van de pagina te zien.



### 14.2.2 VIEWPORT META-ELEMENT

Het automatisch herschalen op mobiele apparaten is prima voor websites die geen rekening houden met kleinere resoluties; op die manier blijven de meeste sites min of meer bruikbaar op een mobiel apparaat.

Er blijft echter een nadeel: omdat de website 3x zo klein is geworden, moet de gebruiker zoomen om iets te kunnen lezen. Omdat hij inzoomt, verliest hij het overzicht en moet hij opnieuw pannen naar links en rechts.

Daarom worden websites adaptive en responsive gemaakt: de layout herschikt zich automatisch naar de beschikbare ruimte zodat de gebruiker niet meer moet zoomen en pannen (enkel scrollen naar beneden).

Wanneer we een responsive site maken, is de automatische herschaling van de viewport dus **ongewenst** omdat de website zichzelf kan aanpassen aan de device-width. Voor een responsive of adaptive site moet de viewport dus de volgende eigenschappen hebben:

- De resolutie van het apparaat handhaven (viewport width = device-width)
- Niet automatisch herschalen (viewport scale = 1.0)

We kunnen deze eigenschappen instellen met de volgende **meta** tag in de <head> van je pagina:

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

Er is echter een vervelende eigenschap bij iPhone apparaten; wanneer deze gekanteld worden en vervolgens weer recht gezet worden, verliezen ze deze schaling. Daarom wordt de schaal soms ook beperkt tot een maximum:

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



#### Niet voor alle sites

- Als er geen media queries aanwezig zijn die de layout zelf aanpassen, maak dan gebruik van het standaard gedrag bij automatische schaling

### 14.3 MEDIA QUERIES

Media queries kunnen gebruikt worden om de CSS van je website aan te passen aan het apparaat waarop ze bekeken wordt. We zullen hier enkel de schermgrootte bespreken, maar in principe kun je met media queries veel meer doen.

De media queries voor tablets en smartphones toevoegen, doe je door het volgende toe te voegen [in de css-file](#).

```
@media only screen and (max-width: 60em){  
    /* Plaats hier de CSS regels die specifiek zijn voor tablets */  
}  
  
@media only screen and (max-width: 30em){  
    /* Plaats hier de CSS regels die specifiek zijn voor smartphones */  
}
```

Of als je liever met pixels werkt:

```
@media only screen and (min-width : 150px) and (max-width :600px){  
    /* Hier komt de css-aanpassingen voor dit scherm */  
}
```

Bij een schermbreedte tussen 150px en 600px wordt de bestaande css-content door de content overschreven binnen de media-query. De eerder gemaakte niet-gewijzigde css wordt behouden.

#### **14.4 WAT KIES IK BEST ?**

---

Een veelgestelde vraag na het doornemen van deze cursus is: 'Wat is de beste oplossing ?'. Helaas is hier geen eenduidig antwoord op te geven. Als je alle technologieën beheert, kun je als front-end developer zelf een zeer overtuigende keuze gaan maken.

Dit hangt (uiteindelijk) af van heel wat parameters :

- Heb ik voldoende tijd om een werkende website af te leveren ?
- Welke functionaliteiten wil ik op mijn website ?
- Gaat de focus meer naar backend of frontend ?
- ...





# Flexbox en grid

Web Frontend Basics

## INHOUD

|  |           |
|--|-----------|
| <b>15 WERKEN MET FLEXBOX EN GRID</b>                   | <b>3</b>  |
| <b>15.1 Flexbox</b>                                    | <b>3</b>  |
| 15.1.1 Flex-containers en flex-items                   | 3         |
| 15.1.2 Flex Direction en flex-wrap                     | 4         |
| 15.1.3 De main en cross-axis                           | 8         |
| 15.1.4 align-items                                     | 8         |
| 15.1.5 justify-content                                 | 9         |
| 15.1.6 Verdere properties                              | 9         |
| <b>15.2 Een grid-layout gebruiken</b>                  | <b>9</b>  |
| 15.2.1 De grid-container                               | 9         |
| 15.2.2 Rijen en kolommen                               | 10        |
| 15.2.3 Witruimte tussen rijen en kolommen creëeren     | 14        |
| 15.2.4 Elementen rechtstreeks positioneren in het grid | 14        |
| 15.2.5 Grid template areas                             | 18        |
| 15.2.6 Alignering                                      | 20        |
| <b>15.3 Responsive design met Flexbox en Grid</b>      | <b>21</b> |
| 15.3.1 Media queries                                   | 23        |
| 15.3.2 De display property op none plaatsen            | 26        |

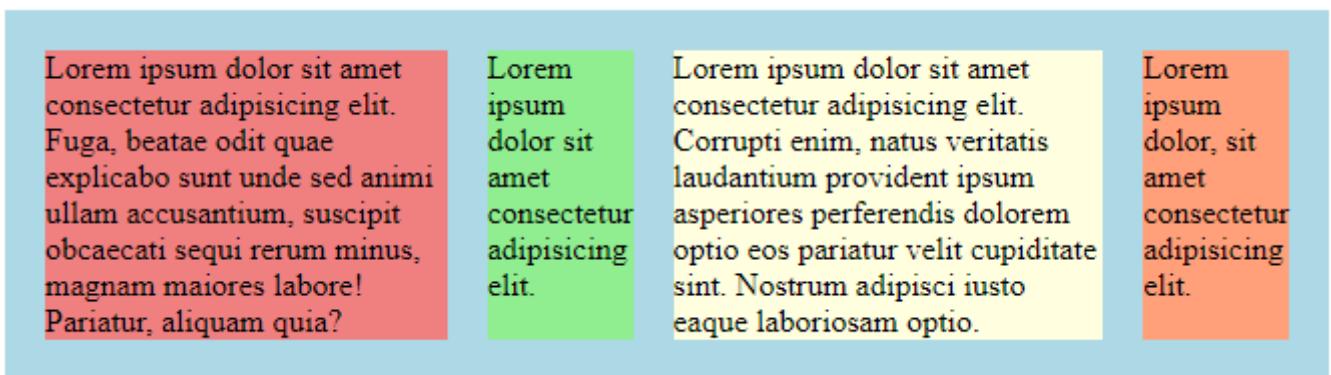
## 15 WERKEN MET FLEXBOX EN GRID

### 15.1 FLEXBOX

Voorlopig hebben we binnen CSS vooral gewerkt met de waarden *block* en *inline* voor de *display* property. Flexbox is een nieuwe layout-modus die toegevoegd is in CSS3 waarmee we de elementen op onze pagina flexibeler kunnen plaatsen. Met de komst van adaptive en responsive design is het gebruik van flexbox belangrijk geworden.

#### 15.1.1 FLEX-CONTAINERS EN FLEX-ITEMS

Een flexbox layout maken, zul je doen door de *display*-property van een html-element op *flex* in te stellen. Dat element noemen we de flex-container. De directe kinderen van de flexcontainer noemen we de flex-items. Onderstaande afbeelding bevat bijvoorbeeld een flex-container (in het blauw) met 4 flex-items (in het rood, groen, geel en oranje).



Deze afbeelding is gemaakt met de volgende html- en css-code.

```
<div id="flex-container">
  <div id="first">Lorem ipsum...</div>
  <div id="second"> Lorem ipsum...</div>
  <div id="third"> Lorem ipsum...</div>
  <div id="fourth"> Lorem ipsum...</div>
</div>
```

```
#flex-container {
  padding: 10px;
  background-color: lightblue;
  display: flex;
}
#first, #second, #third, #fourth {
  margin: 10px;
}
#first {
  background-color: lightcoral;
}
#second {
  background-color: lightgreen;
}
#third {
  background-color: lightyellow;
}
#fourth {
  background-color: lightsalmon;
}
```

Door `display: flex;` toe te voegen aan onze flex-container hebben we ervoor gezorgd dat de verschillende flex-items niet meer onder elkaar, maar naast elkaar komen te staan en dat ze allemaal dezelfde hoogte gekregen hebben.

### 15.1.2 FLEX DIRECTION EN FLEX-WRAP

Flex-direction is een css-property die ingesteld kan worden op een flex-container. Deze property stelt in hoe de verschillende flex-items geplaatst zullen worden. Er zijn 4 mogelijke waarden

- row: de items worden **horizontaal van links naar rechts** geplaatst, dit is de standaardwaarde.
- row-reverse: de items worden **horizontaal van rechts naar links** geplaatst
- column: de items worden **verticaal van boven naar onder** geplaatst
- column-reverse: de items worden **verticaal van onder naar boven** geplaatst

De webpagina zal er dan als volgt uitzien voor de verschillende waarden:

```
#flex-container {  
    flex-direction: row;  
}
```

The image shows four rectangular boxes arranged horizontally. The first box is red and contains the text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga, beatae odit quae explicabo sunt unde sed animi ullam accusantium, suscipit obcaecati sequi rerum minus, magnam maiores labore! Pariatur, aliquam quia?". The second box is green and contains: "Lorem ipsum dolor sit amet consectetur adipisicing elit.". The third box is yellow and contains: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Corrupti enim, natus veritatis laudantium provident ipsum asperiores perferendis dolorem optio eos pariatur velit cupiditate sint. Nostrum adipisci iusto eaque laboriosam optio.". The fourth box is orange and contains: "Lorem ipsum dolor, sit amet consectetur adipisicing elit."

```
#flex-container {  
    flex-direction: row-reverse;  
}
```

The image shows four rectangular boxes arranged horizontally. The first box is orange and contains: "Lorem ipsum dolor, sit amet consectetur adipisicing elit.". The second box is green and contains: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Corrupti enim, natus veritatis laudantium provident ipsum asperiores perferendis dolorem optio eos pariatur velit cupiditate sint. Nostrum adipisci iusto eaque laboriosam optio.". The third box is yellow and contains: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Corrupti enim, natus veritatis laudantium provident ipsum asperiores perferendis dolorem optio eos pariatur velit cupiditate sint. Nostrum adipisci iusto eaque laboriosam optio.". The fourth box is red and contains: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga, beatae odit quae explicabo sunt unde sed animi ullam accusantium, suscipit obcaecati sequi rerum minus, magnam maiores labore! Pariatur, aliquam quia?".

Zoals je kan zien, is de volgorde van de flex-items omgekeerd wanneer we `row-reverse` gebruiken.

```
#flex-container {  
    flex-direction: column;  
}
```

**Quodlibet** ipsum dolor sit amet consectetur adipisicing elit. Fuga, beatae odit quae explicabo sunt unde sed animi ullam accusantium, suscipit obcaecati sequi rerum minus, magnam maiores labore! Pariatur, aliquam quia?

*Lorem ipsum dolor sit amet consectetur adipisicing elit. Corrupti enim, natus veritatis laudantium  
  provident ipsum asperiores preferendis dolorem optio eos pariatur velit cupiditate sint. Nostrum  
  adipisci iusto eaque laboriosam optio.*

Nu staan de verschillende flex-items onder elkaar gerangschikt en hebben ze allemaal dezelfde breedte gekregen in plaats van dezelfde hoogte.

```
#flex-container {  
    padding: 10px;  
    background-color: lightblue;  
    display: flex;  
    flex-direction: column-reverse;  
}
```

*Lorem ipsum dolor sit amet consectetur adipisicing elit. Corrupti enim, natus veritatis laudantium  
  provident ipsum asperiores preferendis dolorem optio eos pariatur velit cupiditate sint. Nostrum  
  adipisci iusto eaque laboriosam optio.*

**Quae explicabo sunt unde sed animi ullam accusantium, suscipit obcaecati sequi rerum minus, magnam maiores labore! Pariatur, aliquam quia?**

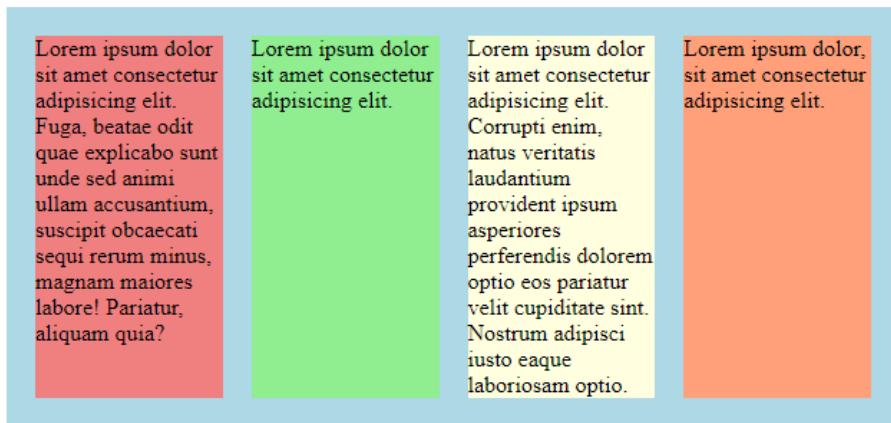
Hier zijn de flex-items terug omgekeerd.

Naast *flex-direction* is er ook nog een property *flex-wrap*. Om het effect van deze property te tonen, geven we eerst via css een breedte aan de flex-items.

```
#first, #second, #third, #fourth {  
    width: 200px;  
}
```

De drie mogelijke waardes zijn *nowrap* (de standaardwaarde), *wrap*, en *wrap-reverse*. Bij *nowrap* zal er nooit een nieuwe rij (als *flex-direction* gelijk is aan *row* of *row-reverse*) of kolom (als *flex-direction* gelijk is aan *column* of *column-reverse*) gestart worden. De *width* property op onze flex-items wordt dus genegeerd.

en als ons browser-venster minder breed wordt, worden onze flex-items dat ook. We tonen even het effect van deze property als de *flex-direction* gelijk is aan *row*.



Als we kiezen voor *wrap*, zal er wel een nieuwe lijn begonnen worden als de flex-items niet meer passen op 1 lijn.

Dan krijg je dus het volgende.



De gele en oranje flex-items passen niet meer op dezelfde lijn als de rode en de groene en zijn dus op een nieuwe lijn gaan staan. Als we kiezen voor *wrap-reverse*, dan komen onze items ook op een nieuwe lijn, maar de nieuwe lijn begint niet onder, maar boven de vorige lijn.



De properties *flex-direction* en *flex-wrap* kunnen gecombineerd worden in 1 property *flex-flow*. Als je *flex-direction* op *column* wil plaatsen en *flex-wrap* op *wrap-reverse*, kun je dat dus ook als volgt doen.

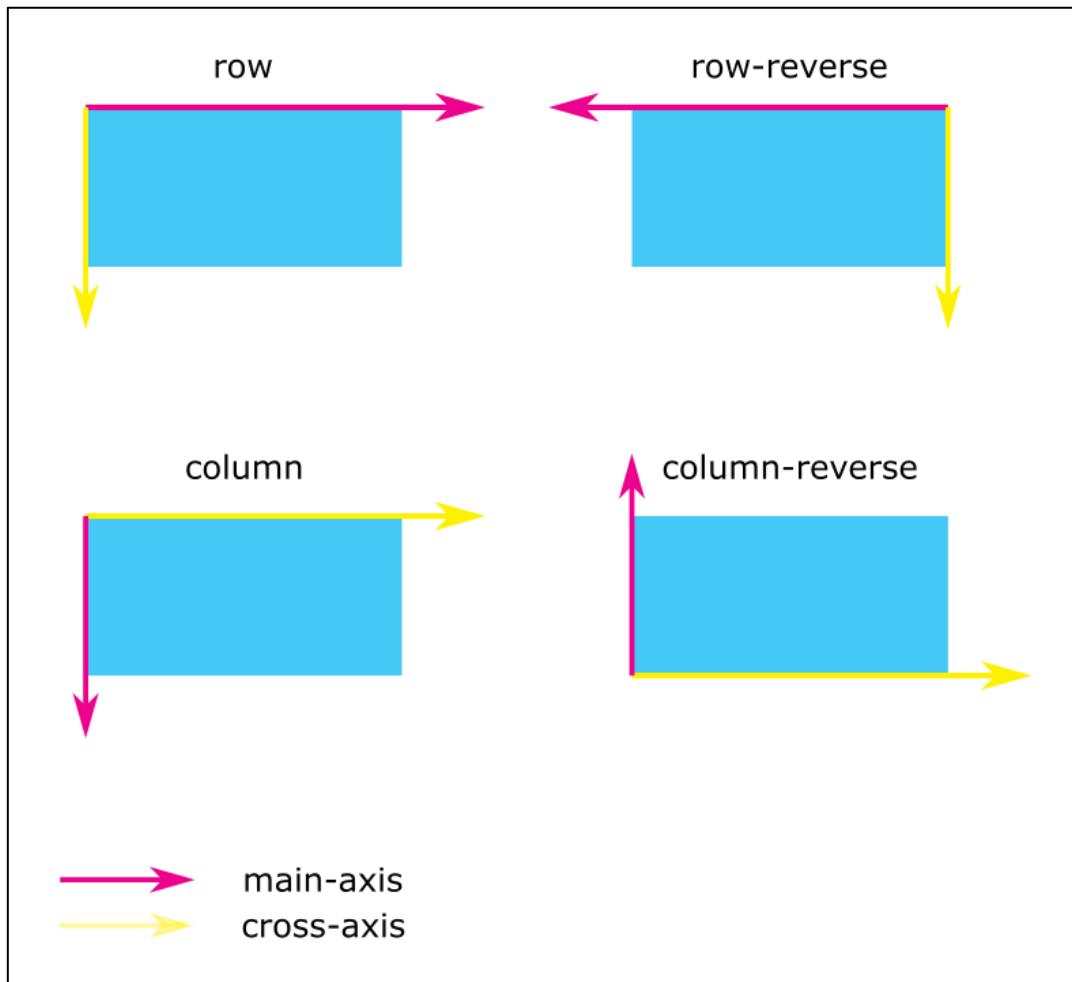
```
#flex-container {  
    display: flex;  
    flex-flow: column wrap-reverse;  
}
```



Probeer gerust eens de verschillende waarden van flex-flow uit om te zien hoe alles gerangschikt wordt en hoe het eruit ziet als je de pagina herschaalt.

### 15.1.3 DE MAIN EN CROSS-AXIS

De main en cross-axis zijn belangrijke concepten binnen Flexbox. Deze kun je het best voorstellen als pijlen op je scherm en ze hangen volledig af van de waarde die je aan flex-direction geeft. Er zijn dus 4 mogelijkheden die hieronder afgebeeld staan.



De main-axis volgt dus altijd de *flex-direction* en de cross-axis staat loodrecht op de main-axis.

### 15.1.4 ALIGN-ITEMS

Met de *align-items* property kun je de positie van de flex-items ten opzichte van de cross-axis veranderen. De standaardwaarde voor deze property is *stretch*. *stretch* zorgt ervoor dat alle flex-items die op dezelfde rij staan dezelfde hoogte krijgen (wanneer *flex-direction* gelijk is aan *row* of *row-reverse*) of dat alle flex-

items die op dezelfde kolom staan dezelfde breedte krijgen (wanneer *flex-direction* gelijk is aan *column* of *column-reverse*).

Enkele mogelijke andere waarden zijn

- *flex-start*: aligneer de flex-items aan het begin langs de cross-axis
- *flex-end*: aligneer de flex-items op het einde langs de cross-axis
- *center*: aligneer de flex-items langs het midden van de cross-axis



Voor meer mogelijke waarden kun je eens kijken op <https://developer.mozilla.org/en-US/docs/Web/CSS/align-items>

Wanneer je deze property niet voor alle flex-items wil toepassen, kun je gebruik maken van de property *align-self* op de flex-items zelf.

#### 15.1.5 JUSTIFY-CONTENT

Met de *justify-content* property kun je de positie van de flex-items ten opzichte van de main-axis veranderen. De standaardwaarde voor deze property is *flex-start*. Deze waarde zorgt ervoor dat onze flex-items mooi aansluiten, startend vanaf het begin van de main-axis.

Enkele mogelijk andere waarden zijn

- *flex-end*: onze flex-items sluiten mooi aan, maar eindigen aan het einde van de main-axis
- *center*: onze flex-items sluiten elkaar mooi aan, maar zijn gecentreerd op de main-axis
- *space-between*: onze flex-items worden mooi verdeeld op de main-axis



Voor meer mogelijke waarden kun je eens kijken op <https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content>

#### 15.1.6 VERDERE PROPERTIES

Als je ingewikkelder zaken wil doen, kun je altijd eens kijken naar de properties *align-content* en *justify-items* die van toepassing zijn op de flex-container en de properties *order*, *flex-grow*, *flex-shrink* en *flex-basis* die je op de flex-items kunt toepassen. Deze behoren echter niet tot de inhoud van de cursus aangezien ze ons te ver zouden leiden.

### 15.2 EEN GRID-LAYOUT GEBRUIKEN

We hebben geleerd hoe je er met flexbox voor kan zorgen dat html-elementen mooi gerangschikt kunnen worden over rijen **of** kolommen, maar wat als je een combinatie van de twee wil? Dan kun je sinds CSS3 een Grid-layout gebruiken. Met deze techniek kun je relatief snel complexere web-layouts creëren die responsief zijn (ze zien er goed uit op verschillende schermgroottes) en consistent over de browsers heen.

#### 15.2.1 DE GRID-CONTAINER

Net zoals bij flexbox is er opnieuw een html element dat we gaan gebruiken als container en dat dus al onze andere elementen zal bevatten. Er zijn dan twee keuzes voor zijn *display* property.

- *grid*: de container gedraagt zich als een block element
- *inline-grid*: de container gedraagt zich als een inline element

```
<div id="grid-container">  
    <h1>Mijn eerste Grid container</h1>  
</div>
```

Voorbeeld HTML

```
#grid-container{  
    display: grid;  
}
```

De grid-container gedraagt zich als een block element

```
#grid-container{  
    display: inline-grid;  
}
```

De grid-container gedraagt zich als een inline element

Wanneer je `display: grid;` gebruikt, krijg je het volgende in de html-inspector.

The screenshot shows the Chrome DevTools Elements tab. On the left, there's a preview of a blue header bar with the text "Mijn eerste Grid container". Below it, a tooltip shows "div#grid-container 572 x 79.88". On the right, the DOM tree shows the following structure:

```
<!doctype html>
<html lang="en">
  <head>...</head>
  <body>
    <div id="grid-container">...</div> == $0
  </body>
</html>
```

The "grid-container" node is selected. The "Properties" tab is active, showing the following CSS properties:

```
element.style {  
}  
#grid-container { display: grid; }  
div { user agent stylesheet  
      display: block; }
```

In the bottom right corner, there's a visual representation of the grid container as a blue rectangle with dimensions 572 x 79.875. It has a dashed orange border around its padding area, and a solid orange border around its margin area.

En bij `display: inline-grid;` is dit het resultaat.

The screenshot shows the Chrome DevTools Elements tab. On the left, there's a preview of a blue header bar with the text "Mijn eerste Grid container". Below it, a tooltip shows "div#grid-container 367 x 79.88". On the right, the DOM tree shows the same structure as the previous screenshot:

```
<!doctype html>
<html lang="en">
  <head>...</head>
  <body>
    <div id="grid-container">...</div> == $0
  </body>
</html>
```

The "grid-container" node is selected. The "Properties" tab is active, showing the following CSS properties:

```
element.style {  
}  
#grid-container { display: inline-grid; }  
div { user agent stylesheet  
      display: block; }
```

In the bottom right corner, there's a visual representation of the grid container as a blue rectangle with dimensions 367 x 79.875. It has a dashed orange border around its padding area, and a solid orange border around its margin area.

## 15.2.2 RIJEN EN KOLOMMEN

Het is dus zeer gemakkelijk om een grid te definiëren. Nu moeten we enkele kolommen en/of rijen toevoegen. Hiervoor gebruiken we de css-properties `grid-template-columns` en `grid-template-rows`. De

waarden die bij deze property horen, bepalen hoe hoog de rijen en hoe breed de kolommen zullen zijn, we hebben hierbij een aantal opties zoals pixels (px), procenten (%) of fractions (fr).

Fractions vraagt misschien wat extra uitleg. Neem bijvoorbeeld deze code.

```
<div id="grid-container">
  <div class="red">A</div>
  <div class="green">B</div>
  <div class="blue">C</div>
</div>
```

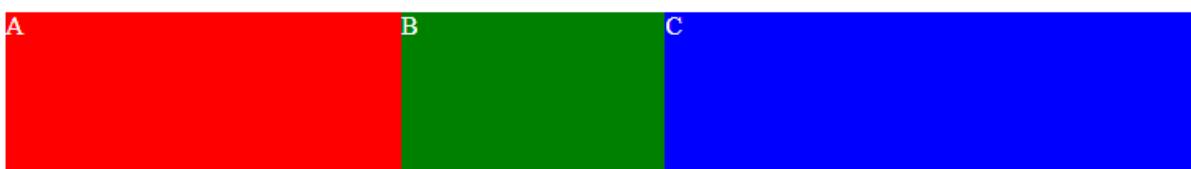
Voorbeeld HTML

```
#grid-container{
  display: grid;
  grid-template-columns: 3fr 2fr 4fr;
}

.red, .green, .blue {
  height: 100px;
  color: white;
}
```

Voorbeeld CSS

Als we er dan voor zorgen dat de juiste achtergrondkleur wordt gegeven aan onze elementen met de klassen *red*, *green* en *blue* krijgen we het volgende te zien in onze browser.



Zoals je kan zien, zijn onze elementen gerangschikt in 3 kolommen waarbij kolom B minder breed is dan kolom A, en kolom B minder breed is dan kolom C. Door het gebruik van fractions heeft de browser de volledige breedte opgesplitst in een aantal even grote stukken en van die stukken heeft hij er 3 aan A gegeven, 2 aan B, en 4 aan C.

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| A | A | A | B | B | C | C | C | C |
|---|---|---|---|---|---|---|---|---|

Laat ons even onze html en css veranderen naar

```
<div id="grid-container">
  <div>A</div>
  <div>B</div>
  <div>C</div>
  <div>D</div>
  <div>E</div>
  <div>F</div>
  <div>G</div>
</div>
```

Voorbeeld HTML

```
#grid-container{
  display: grid;
  grid-template-columns: 3fr 2fr 4fr;
}

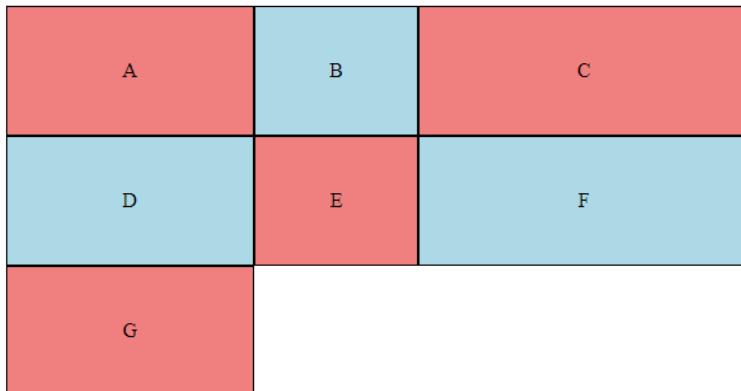
#grid-container>div {
  height: 100px;
}

#grid-container>div:nth-child(even) {
  background-color: lightgray;
}

#grid-container>div:nth-child(odd) {
  background-color: darkgray;
}
```

Voorbeeld CSS

We hebben dus nog steeds 3 kolommen, maar nu hebben we 7 div elementen die we moeten plaatsen. Het resultaat in de browser is dan het volgende (we hebben er in onze CSS voor gezorgd dat de achtergrondkleuren afwisselen tussen rood en blauw).



Wanneer alle kolommen opgebruikt zijn, begint onze grid-layout met een nieuwe rij. De hoogte van die rijen kan ingesteld worden met de *grid-template-rows* property. Net zoals bij *grid-template-columns*, kun je hier ook gebruik maken van pixels, procenten en fractions.

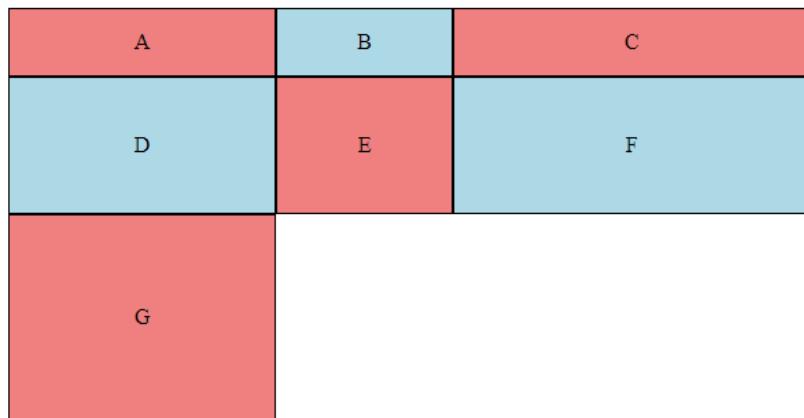
```
#grid-container{
    display: grid;
    grid-template-columns: 3fr 2fr 4fr;
    grid-template-rows: 50px 100px 150px;
}

#grid-container>div:nth-child(even) {
    background-color: lightblue;
}

#grid-container>div:nth-child(odd) {
    background-color: lightcoral;
}

#grid-container>div {
    display: flex;
    align-items: center;
    justify-content: center;
    border: 1px solid black;
}
```

Als we dus de hoogte niet langer instellen op de individuele divs zelf, maar via de grid-container, zien we dat de hoogte van de rijenangepast is.



Naast pixels, procenten en fractions heb je ook de mogelijkheid om het keyword *auto* te gebruiken bij *grid-template-rows* en *grid-template-columns*. In de meeste gevallen zal dit ertoe leiden dat de kolom of rij met waarde auto zo groot mogelijk gemaakt zal worden zodat zijn parent element volledig gevuld is.

Aangezien je vaak veel kolommen hebt die dezelfde breedte moeten hebben, is er een verkorte schrijfwijze. Je kunt namelijk gebruik maken van de *repeat* functie. Als je bijvoorbeeld

```
grid-template-columns: repeat(4, 100px);
```

gebruikt, zullen er 4 kolommen van 100px gecreëerd worden.

Je kan de repeat gerust combineren met “gewone” waarden. Zo zal

```
grid-template-columns: repeat(3, 100px) 150px;
```

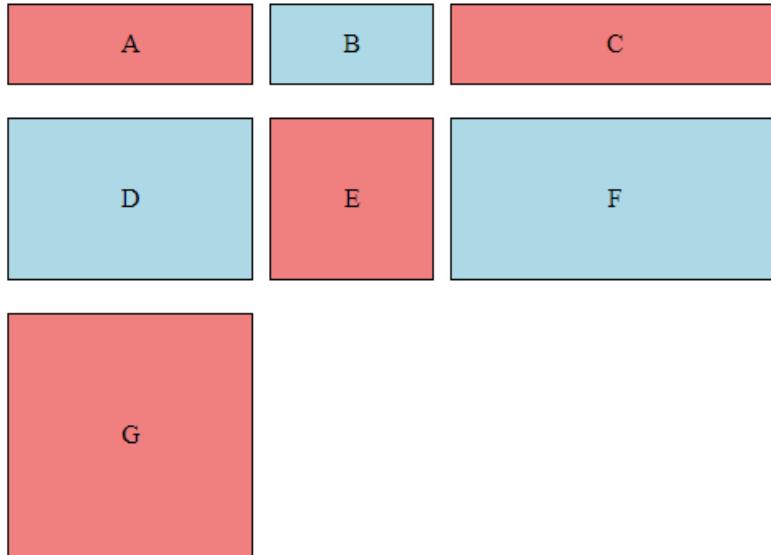
4 kolommen aanmaken. Eerst 3 van 100px, en dan eentje van 150px.

### 15.2.3 WITRUIIMTE TUSSEN RIJEN EN KOLOMMEN CREEËREN

Het is ook mogelijk om witruimte tussen de kolommen en de rijen van je grid toe te voegen. Dit kun je doen met de CSS properties *column-gap* en *row-gap*. Als je dus de regels

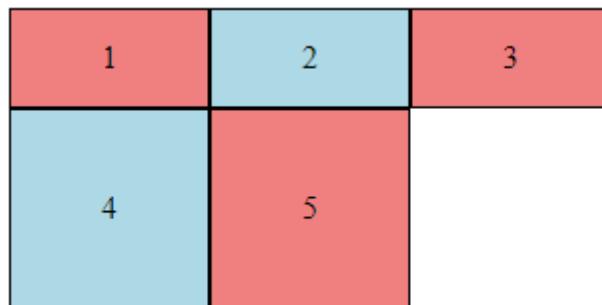
```
column-gap: 10px;  
row-gap: 20px;
```

toevoegt aan de css van je grid-container, krijg je het volgende in de browser.



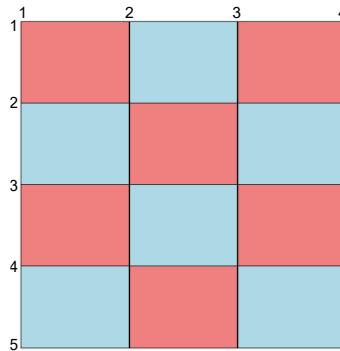
### 15.2.4 ELEMENTEN RECHTSTREEKS POSITIONEREN IN HET GRID

Standaard worden de HTML-elementen altijd in het eerste nog niet opgevulde vakje van je grid geplaatst. Zo wordt je grid rij per rij opgevuld van links naar rechts zoals in onderstaande afbeelding die een grid met 3 kolommen voorstelt.



Maar dit gedrag kunnen we wijzigen door gebruik te maken van de CSS properties *grid-column-start*, *grid-column-end*, *grid-row-start* en *grid-row-end*. Om de invloed van deze properties te kunnen visualiseren,

moeten we eerst vertrouwd geraken met grid-lines. Een grid met 3 kolommen en 4 rijen zal bestaan uit 4 verticale en 5 horizontale grid-lines. Deze zijn dan als volgt genummerd.



De verschillende grid-lines in een grid met 3 kolommen en 4 rijen

Je ziet dus dat er altijd 1 grid-line meer is dan het aantal rijen/kolommen.

Met de *grid-column-start* kunnen we nu instellen op welke grid-line we ons element willen laten starten. En met *grid-column-end* kunnen we aangeven waar we willen dat ons element stopt. Met *grid-row-start* en *grid-row-end* kunnen we dan hetzelfde doen voor de rijen. Kijk bijvoorbeeld naar de volgende html en css code

```
<div id="grid-container">
  <div id="A">A</div>
  <div id="B">B</div>
  <div id="C">C</div>
  <div id="D">D</div>
  <div id="E">E</div>
  <div id="F">F</div>
</div>
```

HTML code

```
#grid-container{
  display: inline-grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(4, 1fr);
  height: 500px;
  width: 250px;
}

#grid-container>div:nth-child(even) {
  background-color: lightblue;
}

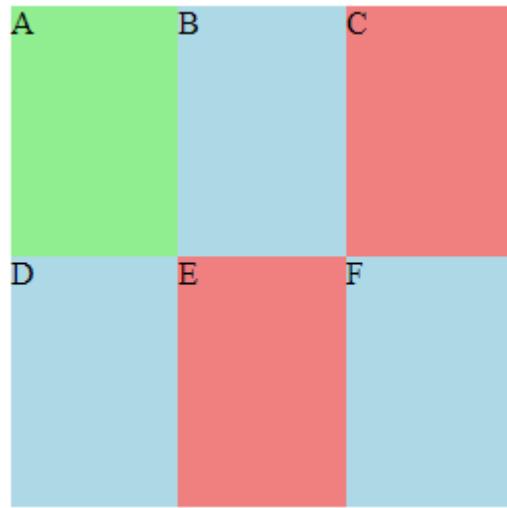
#grid-container>div:nth-child(odd) {
  background-color: lightcoral;
}

#A {
  background-color: lightgreen !important;
}
```

CSS code

Het *!important* keyword zorgt er hier voor dat de stijlregel niet overschreven zal worden in andere CSS-selectoren.

In onze browser ziet onze grid-container er zo uit.

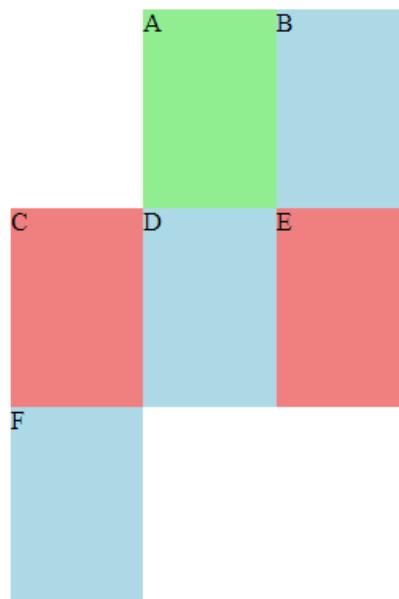


Zoals verwacht staat ons groene A element in het eerste vakje van onze grid. Het is namelijk het eerste kind van onze grid-container.

Maar als we nu de *grid-column-start* van het A element instellen

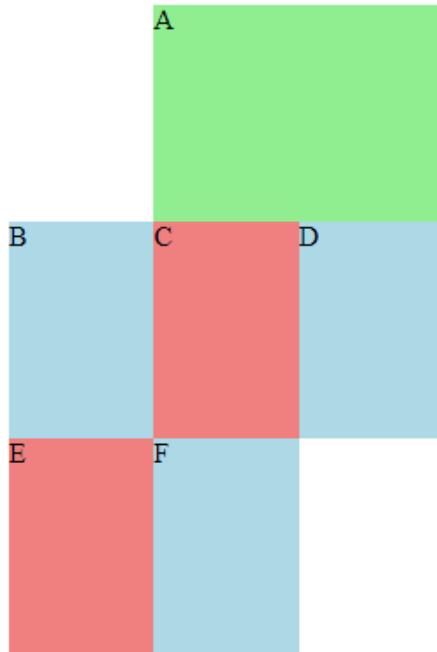
```
#A {  
    background-color: lightgreen !important;  
    grid-column-start: 2;  
}
```

Dan zal het A-element pas starten na de tweede verticale grid-line.



Als we ook `grid-column-end` instellen, zal ons A-element ook op een andere positie eindigen.

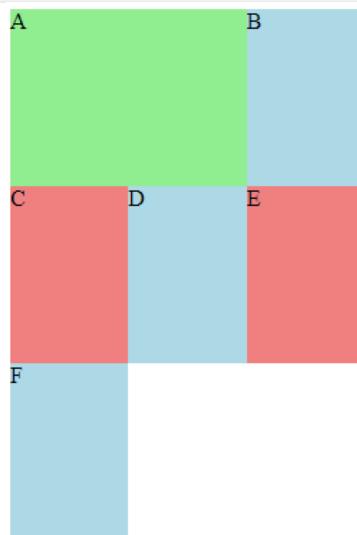
```
#A {  
    background-color: lightgreen !important;  
    grid-column-start: 2;  
    grid-column-end: 4;  
}
```



Je ziet dus dat het nu 2 vakjes inneemt.

Als je zou willen dat het A-element de eerste 2 vakjes inneemt, zou je dit ook als volgt kunnen doen.

```
#A {  
    background-color: lightgreen !important;  
    grid-column-start: span 2;  
}
```



### 15.2.5 GRID TEMPLATE AREAS

Het is ook mogelijk om met zogenaamde grid-areas te werken. Dan kun je in grid-container iedere cel een label geven waarnaar toe je dan verwijst in de elementen die in je grid komen. Stel bijvoorbeeld dat je de volgende website wil maken.

The screenshot shows a website layout with a dark sidebar on the left and a main content area on the right. The sidebar has four items: 'Home', 'Arms', 'Legs', and 'Brains'. The main content area has a header 'Welkom op de website van Zombie inc.' and a section titled 'Zombie stuff' with three paragraphs of placeholder text (Lorem ipsum). The footer contains the copyright notice '© 2019'.

**Welkom op de website van Zombie inc.**

**Zombie stuff**

**Home**

**Arms**

**Legs**

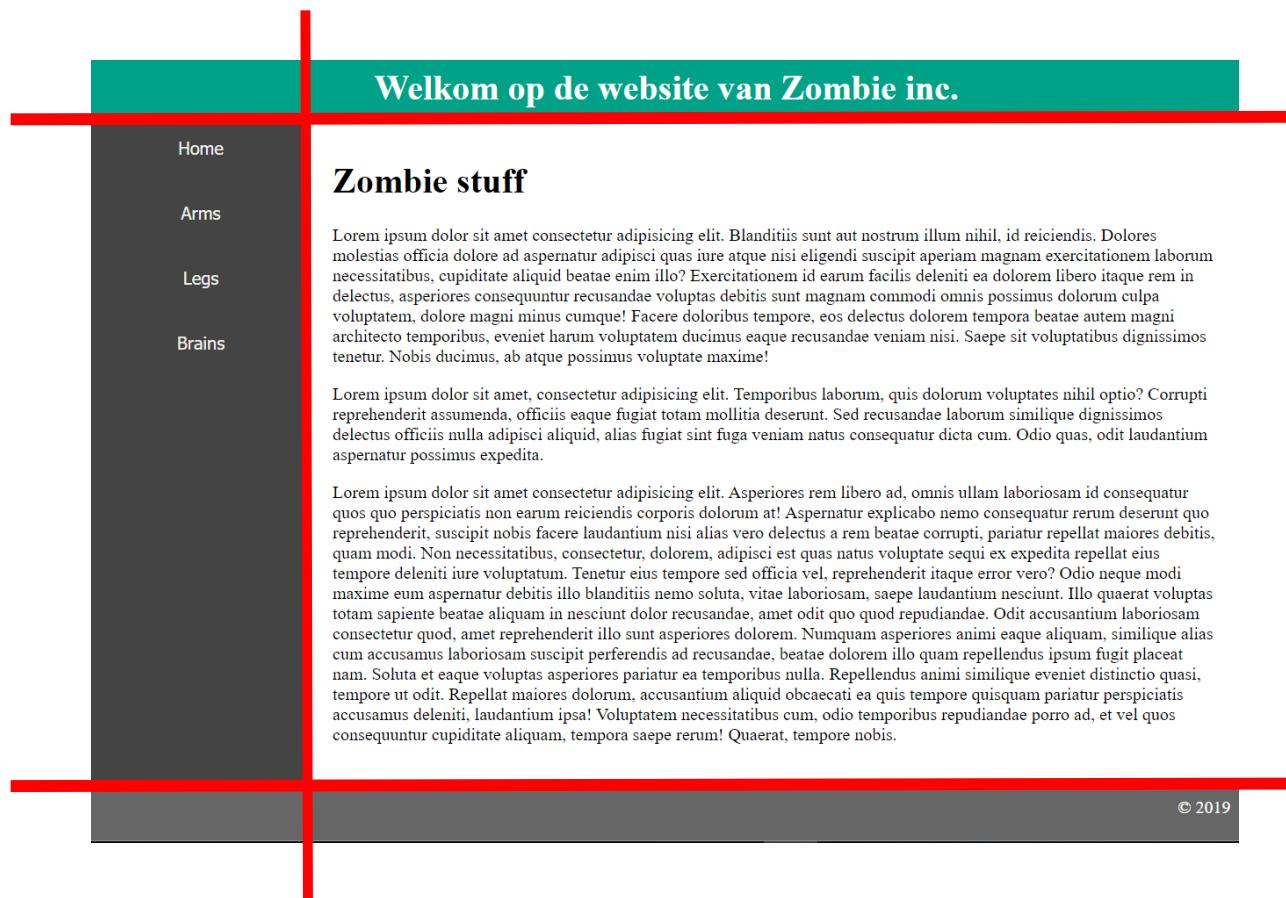
**Brains**

**© 2019**

We gebruiken daarvoor de volgende html code.

```
<header>
  <h1>Welkom op de website van Zombie inc.</h1>
</header>
<nav>
  <ul>
    <li>Home</li>
    <li>Arms</li>
    <li>Legs</li>
    <li>Brains</li>
  </ul>
</nav>
<main>
  <h1>Zombie stuff</h1>
  <p>...</p>
  <p>...</p>
  <p>...</p>
</main>
<footer>
  © 2019
</footer>
```

Als je kijkt naar de structuur van deze webpagina, bestaat deze uit 3 rijen en 2 kolommen.



We zullen dus de volgende CSS moeten toevoegen op ons body element.

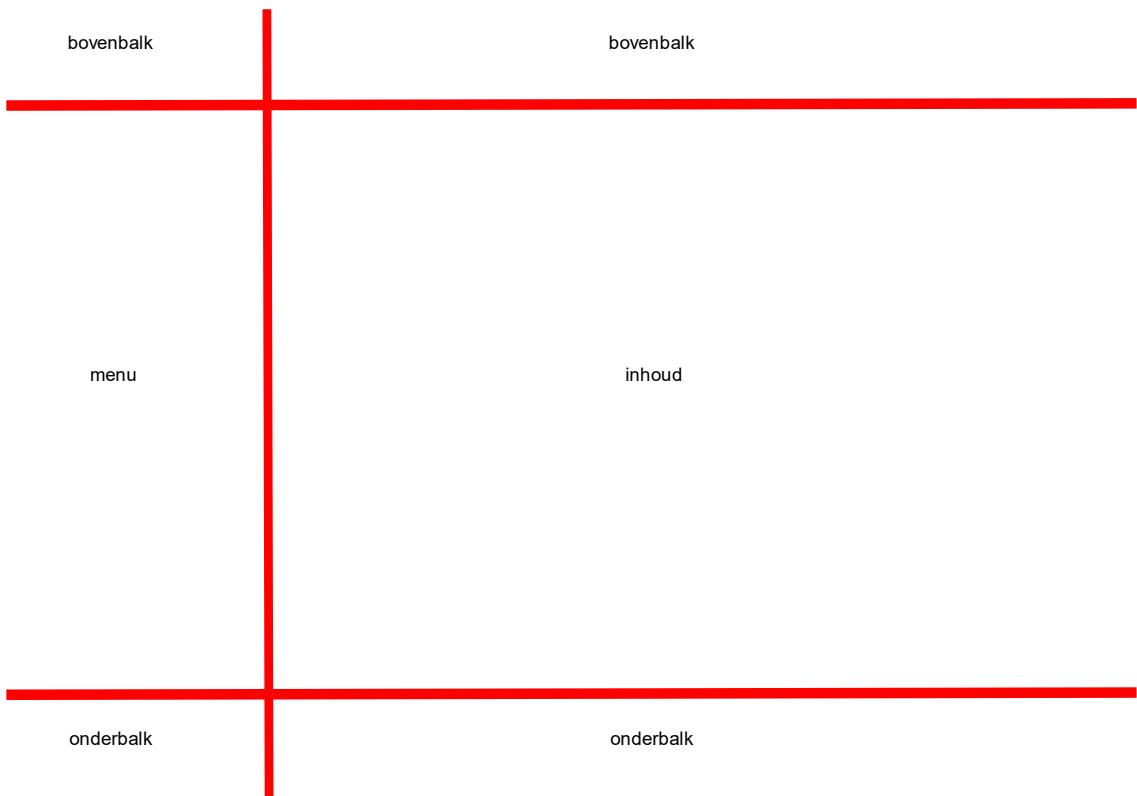
```
body {  
    display: grid;  
    grid-template-columns: 200px auto;  
    grid-template-rows: 50px auto 50px;  
}  
  
body, html {  
    height: 100%; /* zodat de footer beneden staat */  
    margin: 0;  
    padding: 0;  
}
```

We stellen dus de breedte van het zij-menu in op 200px en de hoogte van de header en de footer op 50px. Aangezien de tweede kolom en rij de overige plaats mogen innemen, geven we ze de waarde *auto*.

Om de zaken wat makkelijker te maken voegen we dan de *grid-template-areas* property toe aan onze grid-container.

```
body {  
    display: grid;  
    grid-template-columns: 200px auto;  
    grid-template-rows: 50px auto 50px;  
    grid-template-areas: "bovenbalk bovenbalk" "menu inhoud" "onderbalk onderbalk";  
}
```

Hiermee geven we voor iedere rij aan wat er in de vakken van het grid terecht moet komen. In ons voorbeeld is dat dus.



Nu hebben we onze grid-container dus opgedeeld in 4 grid-areas (bovenbalk, menu, inhoud en onderbalk), zodat in ieder gebied één van onze elementen kan komen te staan. We kunnen nu dus de *grid-area* property toepassen op onze grid-elementen, zodat deze weten waar ze terecht moeten komen.

```
header {  
    grid-area: bovenbalk;  
}  
  
nav {  
    grid-area: menu;  
}  
  
footer {  
    grid-area: onderbalk;  
}  
  
main {  
    grid-area: inhoud;  
}
```

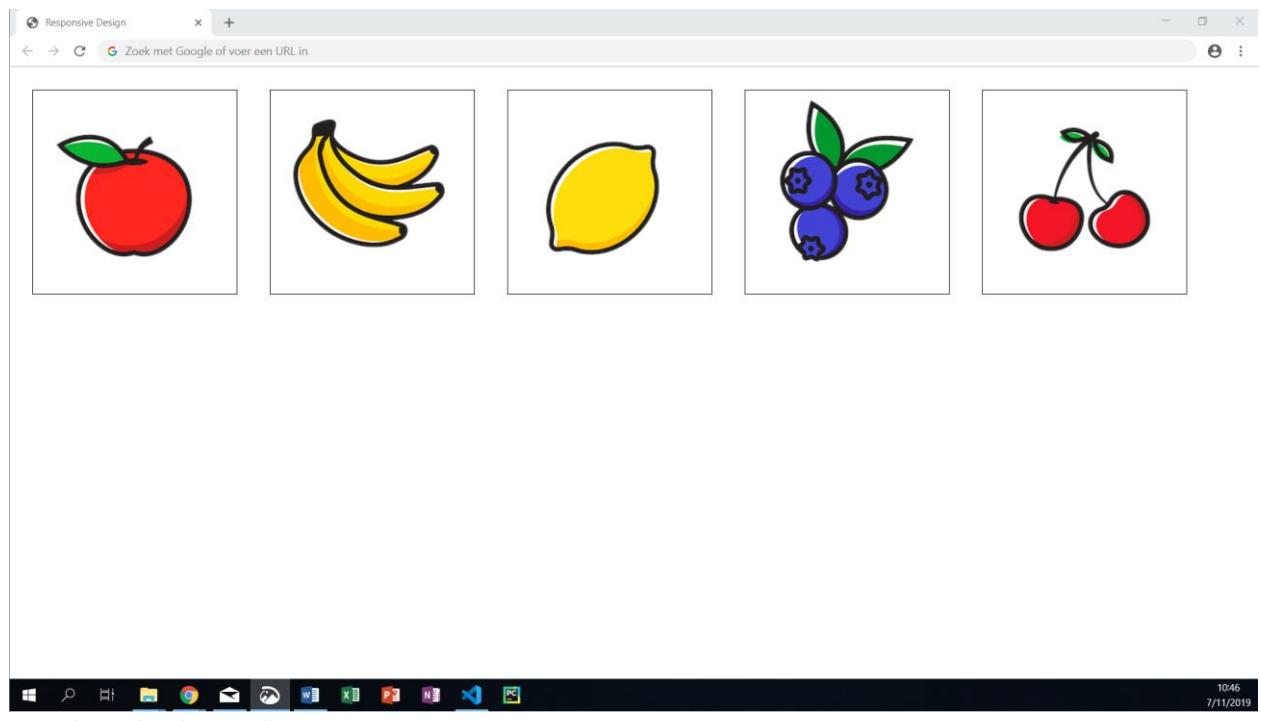
### 15.2.6 ALIGNERING

Net zoals bij flexbox kun je terug gebruik maken van de properties *align-items*, *align-content*, *justify-items* en *justify-content* om de elementen in je grid layout te aligneren.

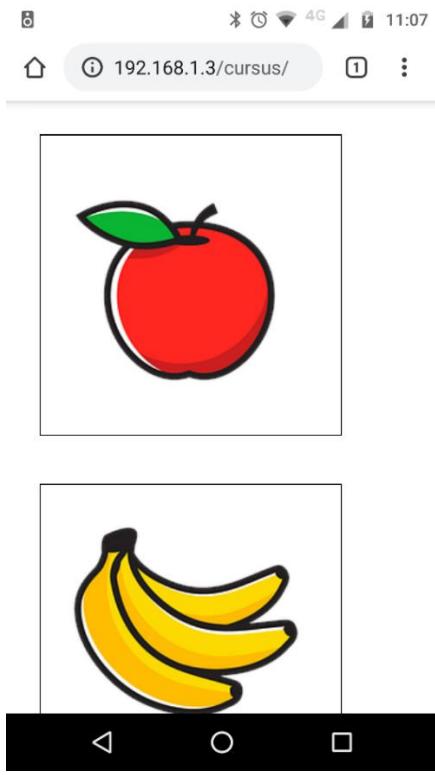
### 15.3 RESPONSIVE DESIGN MET FLEXBOX EN GRID

Er voor zorgen dat je website er goed uitziet op verschillende schermgroottes wordt responsive design genoemd. Omdat websites de dag van vandaag op zowel smartphones, tablets en laptops bekeken worden, is responsive design zeer belangrijk geworden bij de ontwikkeling van websites. De komst van flexbox en grid in CSS3 heeft dit een stuk eenvoudiger gemaakt.

Als je gebruik maakt van flex-wrap in je layout, zullen je flex-items automatisch anders geordend worden als de schermgrootte wijzigt.

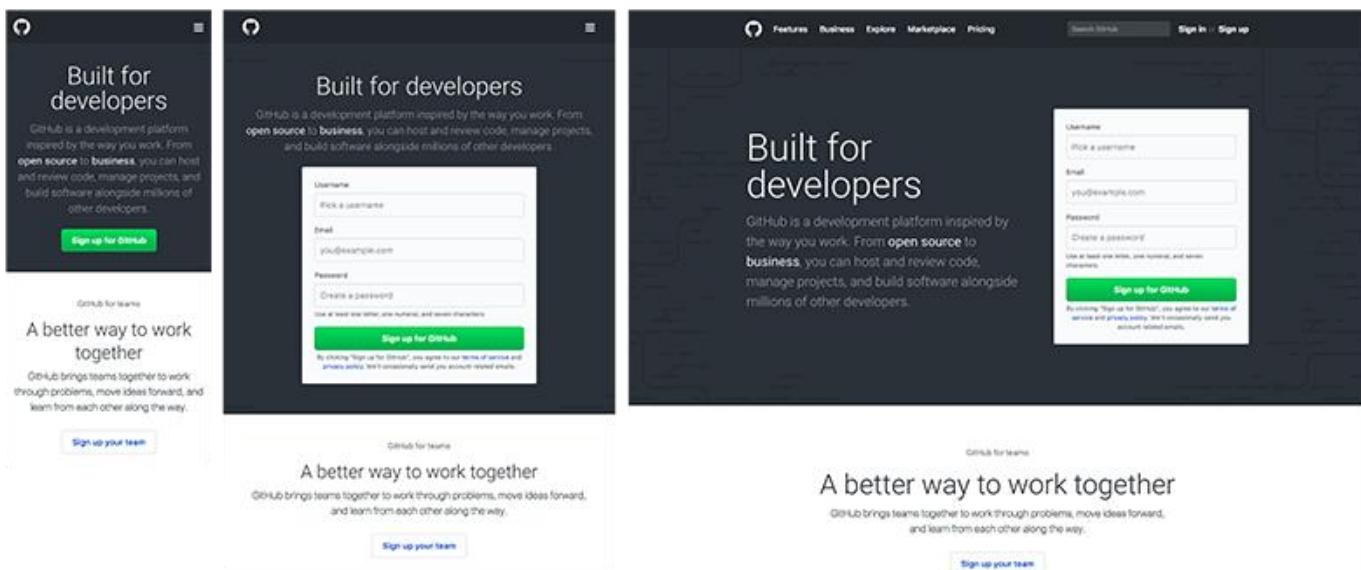


De websiteinhoud op een laptop



De websiteinhoud op een smartphone

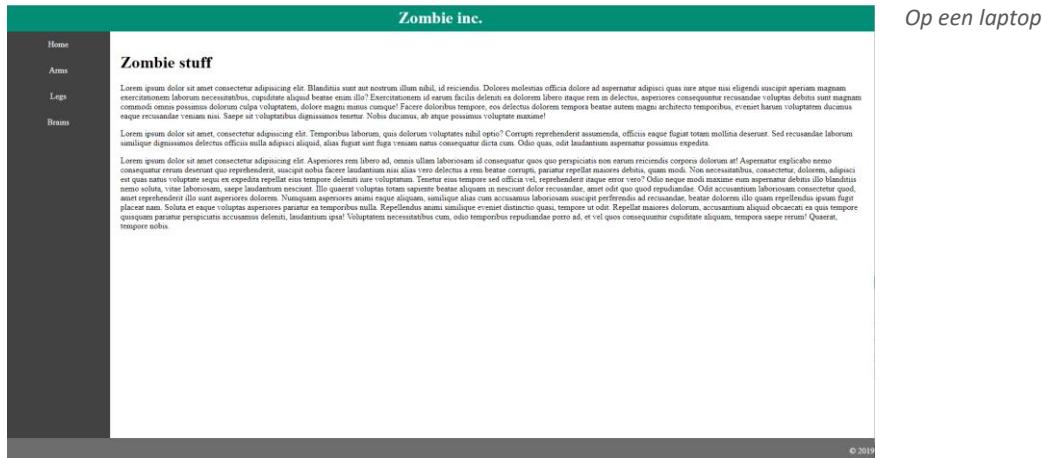
En als je gebruik maakt van fractions of procenten, zal je grid-layout zich ook automatisch aanpassen aan je schermgrootte. Maar soms wil je meer. Kijk bijvoorbeeld naar hoe de website [github.com](https://github.com) eruit ziet op verschillende devices.



Zoals je kan zien, verandert er veel meer dan enkel de schikking van de componenten. Zo bevat de menu-balk bijvoorbeeld geen items op tablets en smartphones, en is het inlogformulier vervangen door 1 enkele knop op smartphones.

### 15.3.1 MEDIA QUERIES

Onze Zombie website kunnen we er als volgt laten uitzien op verschillende devices.



Op een laptop



Op een tablet

# Zombie inc.

Home

Arms

Legs

Brains

*Op een smartphone*

## Zombie stuff

  Lorem ipsum dolor sit amet consectetur  
  adipisicing elit. Blanditiis sunt aut nostrum illum  
  nihil, id reiciendis. Dolores molestias officia  
  dolore ad aspernatur adipisci quas iure atque nisi  
  eligendi suscipit aperiam magnam  
  exercitationem laborum necessitatibus,  
  cupiditate aliquid beatae enim illo?  
  Exercitationem id earum facilis deleniti ea  
  dolorem libero itaque rem in delectus, asperiores  
  consequuntur recusandae voluptas debitis sunt  
  magnam commodi omnis possimus dolorum

We gebruiken de volgende HTML-code voor deze website

```
<body>
<header>
  <h1>Zombie inc.</h1>
</header>
<nav>
  <ul>
    <li>Home</li>
    <li>Arms</li>
    <li>Legs</li>
    <li>Brains</li>
  </ul>
</nav>
<main>
  <h1>Zombie stuff</h1>
  <p>...</p>
  <p>...</p>
  <p>...</p>
</main>
<footer>
  © 2019
</footer>
</body>
```

En de volgende CSS code voor de laptop versie:

```
html, body {
  height: 100%;
  margin: 0px;
  padding: 0px;
}

body {
  display: grid;
  grid-template-columns: 200px auto;
  grid-template-rows: 50px auto 50px;
  grid-template-
areas: "bovenbalk bovenbalk" "navigatie inhoud"
"onderbalk onderbalk";
}

header {
  background-color: #008e76;
  grid-area: bovenbalk;
}

nav {
  background-color: #424242;
  grid-area: navigatie;
}

main {
  grid-area: inhoud;
}

footer {
  background-color: #6d6d6d;
  grid-area: onderbalk;
}

header h1 {
  text-align: center;
  color: white;
  margin: 5px;
}

main {
  margin: 20px;
}

nav {
  text-align: center;
}

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

nav ul li {
  display: inline-block;
  text-align: center;
  width: 100%;
  height: 50px;
  line-height: 50px;
```

```

        color: white;
    }

footer {
    color: white;
    text-align: right;
    line-height: 50px;
}

```

Vervolgens voegen we de volgende media-query toe om de tablet versie deftig te doen werken

```

@media screen and (max-width: 60em){
    body {
        grid-template-columns: auto;
        grid-template-rows: 50px 50px auto 50px;
        grid-template-areas: "bovenbalk" "navigatie" "inhoud" "onderbalk";
    }

    nav ul li {
        width: 100px;
    }
}

```

We hebben dus de structuur van onze grid-layout veranderd naar 4 rijen en onze menu-items een vaste breedte van 100px gegeven.

Voor de smartphone versie kunnen we dan de volgende media-query toevoegen

```

@media screen and (max-width: 480px){
    body {
        grid-template-columns: auto;
        grid-template-rows: 50px 200px auto 50px;
        grid-template-areas: "bovenbalk" "navigatie" "inhoud" "onderbalk";
    }

    nav ul li {
        width: 100%;
    }
}

```

### 15.3.2 DE DISPLAY PROPERTY OP NONE PLAATSEN

Ondertussen zijn we al een aantal keer in aanraking gekomen met de CSS display property. Naast *block*, *inline*, *inline-block*, *flex* en *grid* is het ook mogelijk om *none* te gebruiken. Als je de *display*-property van een element op *none* zet, zal het niet getoond worden op de pagina. Hiervan kun je dus gebruik maken als je bijvoorbeeld een element niet wil tonen op de smartphone versie van je website. Dan zet je in de bijhorende mediaquery gewoon de *display* property van dat element op *none*.





# Transformaties en animaties

**Web Frontend Basics**

## INHOUD

|   |          |
|---|----------|
| <b>16 TRANSFORMEREN EN ANIMEREN</b>             | <b>3</b> |
| <b>16.1 Transformeren</b>                       | <b>3</b> |
| 16.1.1 De transform property                    | 3        |
| 1.1.1.1 Overzicht                               | 4        |
| 16.1.2 De transform-origin property             | 5        |
| <b>16.2 Transitions</b>                         | <b>7</b> |
| 16.2.1 Vertragingen (delay) en timing functions | 7        |
| 1.1.1.2 timing functions                        | 8        |
| 16.2.2 Specifieke transition properties         | 8        |
| <b>16.3 Animations</b>                          | <b>9</b> |
| 16.3.1 Hulpmiddelen                             | 11       |

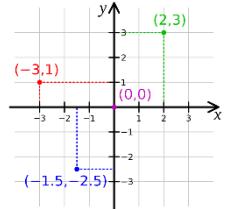
# 1 TRANSFORMEREN EN ANIMEREN

## 16.1 TRANSFORMEREN

### 16.1.1 DE TRANSFORM PROPERTY

Met de CSS transform-property kun je elementen roteren, schalen, scheeftrekken of verplaatsen. De elementen die getransformeerd worden, breken de flow van het document niet.

Om de elementen te transformeren wordt het (Cartesisch) coördinatenstel gebruikt. Indien relevant, kan je dus aan de hand van een x- of y-waarde de elementen transformeren. Het eerste getal is daarbij steeds de x-coördinaat, het tweede de y-coördinaat. Werken we in 3D, dan komt daar ook nog een z-coördinaat bij.



Bij rotaties is dit coördinatenstelsel uiteraard minder relevant. Hier geef je gewoon een enkele waarde mee: 45° roteren bijvoorbeeld. Deze waarden kunnen echter niet worden uitgedrukt in een lengte of grootte. We hebben nood aan eenheden om hoeken uit te drukken. Hieronder een aantal mogelijke opties:

- deg (degrees): uitdrukking in graden, 360deg is een volle cirkel
- grad (gradians): uitdrukking in decimale graden, 400grad is een volle cirkel
- rad (radians): uitdrukking in radialen, 6.2832rad is een volle cirkel
- turn: uitdrukking in toeren, 1turn is een volle cirkel

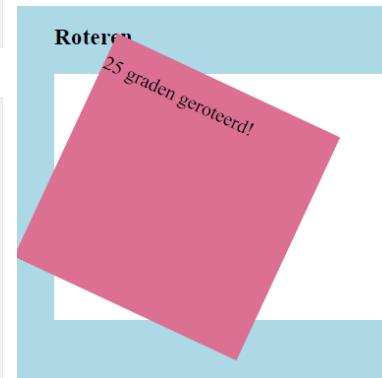
Hieronder een voorbeeld van hoe je een element kan roteren:

```
<h3>Roteren</h3>
<div class="container">
  <div class="square rotate"><p>25 graden geroteerd!</p></div>
</div>
```

```
.container {
  background-color: white;
}

.square {
  width: 200px;
  height: 200px;
  background-color: palevioletred;
  display: inline-block;
}

.rotate {
  transform: rotate(25deg);
}
```



In het bovenstaande voorbeeld beginnen we met een div die via css als een roze vierkantje wordt weergegeven. Daarnaast voorzien we een extra klasse, "container", om duidelijker te kunnen zien hoe het vierkant getransformeerd wordt.

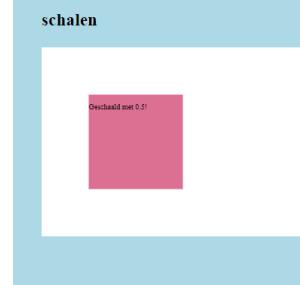
Met behulp van een tweede klasse wordt de CSS transform property ingesteld en wordt er met een *transform function* gekozen hoe het element getransformeerd moet worden. Hier kiezen we er dus voor om het element 25° te roteren.

Merk op dat het vierkant een stuk van de titel bedekt.

We maken nu een aantal nieuwe klassen aan die de andere *transform functions* illustreren. In het voorbeeld hieronder schalen we het element:

```
<h3>schalen</h3>
<div class="container">
  <div class="square scale"><p>Geschaald met 0.5!</p></div>
</div>

...
.scale {
  transform: scale(0.5);
  /* Hetzelfde als scaleX(0.5) + scaleY(0.5) */
}
```



In het voorbeeld hieronder vervormen we het element. Deze keer geven we de x- en de y-waarde apart mee. De kortere variant staat in commentaar:

```
<h3>skew (scheeftrekken, vervormen)</h3>
<div class="container">
  <div class="square skew"><p>Scheefgetrokken!</p></div>
</div>
```

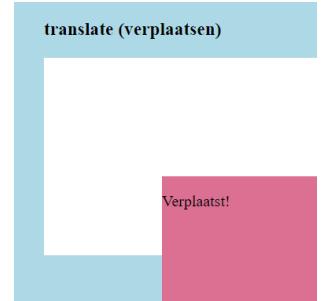
```
...
.skew {
  /* transform: skew(25deg, 2deg); */
  transform: skewX(25deg) skewY(2deg);
}
```



In het voorbeeld hieronder verschuiven we het element:

```
<h3>translate (verplaatsen)</h3>
<div class="container">
  <div class="square translate"><p>Verplaatst!</p></div>
</div>
```

```
...
.translate {
  transform: translate(120px, 120px);
}
```



Tot slot is het mogelijk om meerdere *transform functions* op één lijn te definiëren. Een voorbeeld:

```
<h3>alles gecombineerd</h3>
<div class="container">
  <div class="square combined"><p>Gecombineerd!</p></div>
</div>
```

```
...
.combined {
  transform: skewX(25deg) skewY(2deg) scaleX(1.3)
    scaleY(0.5) rotate(-10deg)
    translateX(120px);
}
```



## 1.1.1.1 OVERZICHT

| Transform function  | Beschrijving   |
|---------------------|--|
| <b>rotate()</b>     | Roteert het elementen o.b.v. een meegegeven hoek. Indien het getal positief is, wordt het element klokwijs geroteerd. Is het getal negatief, dan wordt het element tegen de klok in geroteerd.<br><i>Voorbeeld: rotate(-25deg);</i>  |
| <b>scale()</b>      | Bepaalt hoe het element geschaald moet worden over de x- en y-as. Indien er maar één waarde wordt meegegeven, zijn de x- en y-waarde gelijk.<br>Een waarde gelijk aan 1 zal dus geen effect hebben. Is de waarde groter, dan wordt het element vergroot (bv. 2 betekent twee keer zo groot). Is de waarde kleiner, dan wordt het element verkleind (bv. 0.5 betekent half zo groot).<br><i>Voorbeeld: scale(0.5) of scale(0.5, 1.5);</i> |
| <b>scaleX()</b>     | Schaalt enkel langs de x-as  |
| <b>scaleY()</b>     | Schaalt enkel langs de y-as  |
| <b>skew()</b>       | Trekt het element (elk punt) scheef op basis van een bepaalde hoek in de horizontale en verticale as.<br><i>Voorbeeld: skew(25deg, 2deg);</i>  |
| <b>skewX()</b>      | Trekt enkel scheef langs de x-as   |
| <b>skewY()</b>      | Trekt enkel scheef langs de y-as   |
| <b>translate()</b>  | Verplaatst het element op basis van een opgegeven afstand in de horizontale en verticale richting.<br><i>Voorbeeld: translate(120px, 200px);</i>   |
| <b>translateX()</b> | Verplaatst enkel langs de x-as   |
| <b>translateY()</b> | Verplaatst enkel langs de y-as   |

Transform eigenschappen in css

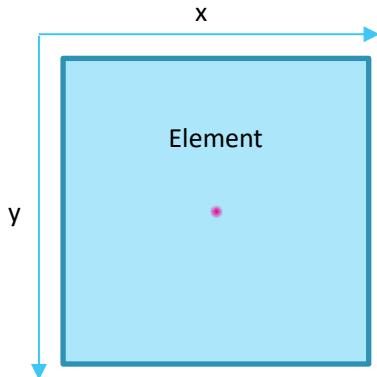
### 16.1.2 DE TRANSFORM-ORIGIN PROPERTY

Wanneer we een element roteren, schalen of scheeftrekken, dan kunnen we bepalen ten opzichte van welk punt het element getransformeerd moet worden. Standaard wordt het middelpunt van het element gekozen, maar dit kunnen we dus aanpassen.

Op de tekening hieronder zie je waar de standaardwaarde van de *transform-origin* ligt. Dit is dus 50 % 'ver' op de x-as, 50 % op de y-as (en 0 op de z-as). Dit wordt ook wel aangeduid met 'center'.

Door andere waarden mee te geven dan de standaardwaarden kunnen we dus het punt waarlangs de transformaties gebeuren, aanpassen. Je geeft dan een eenheid mee langs de x- of y-as (bv. 10px, 20%, enz.). Je kan echter ook kiezen voor de volgende keywords:

- top, bottom
- left, right
- center



```
transform-origin: 50% 50%; /* standaard waarde volgens x, y */  
transform-origin: center; /* standaard waarde met keyword */
```

Bekijk de volgende voorbeelden. De punten waarlangs geroteerd werd, zijn in de voorbeelden aangeduid.

```
<h2>Transform-origin</h2>  
<h3>Een ander punt waarlangs  
we transformeren ...</h3>  
<div class="container">  
  <div class="square transformOriginTo  
p">  
    <p>transform top left!</p>  
  </div>  
  <div class="square transformOriginBo  
ttom">  
    <p>transform x: 80%, y: 10%!</p>  
  </div>  
</div>
```

```
.transformOriginTop {  
  transform-origin: top left;  
  transform: rotate(-5deg);  
  border: 1px solid white;  
}  
  
.transformOriginBottom {  
  transform-origin: 80% 10%;  
  transform: rotate(5deg);  
  border: 1px solid white;  
}
```

Een ander punt waarlangs we transformeren ...



## 16.2 TRANSITIONS

Transities zijn het eerst opstapje naar een animatie. Een transitie bekent dat het element verandert van CSS-eigenschappen onder een bepaalde voorwaarde (*state*). De state van het element wordt bepaald door zijn pseudo-klassen. De transition property geeft een antwoord op de vraag: “hoe moet de overgang verlopen?”.



```
.myElement {  
    width: 100px;  
    height: 100px;  
    margin-top: 5em;  
    color: black;  
    background-color: pink;  
    transition: 1s; /* alle properties (van hover) worden veranderd binnen 1 seconde*/  
}  
  
.myElement:hover {  
    background-color: lightskyblue;  
    border-radius: 50% 50%;  
    transform: rotateY(360deg);  
}
```

Het bovenstaand voorbeeld toont aan dat we een element voorzien van een ‘standaard’ opmaak en een andere opmaak die geactiveerd wordt wanneer er over het element gehovered wordt (met behulp van de :hover pseudo-klasse). Wanneer er over .myElement gehovered wordt, verandert de background-color naar lightskyblue en dat over een tijdspanne van 1 seconde. Ook de border-radius en en transform-eigenschappen zullen veranderen over diezelfde tijdspanne.

Nu, misschien wil je wel dat de ene property er langer of korter over doet dan de andere. Indien je de properties individueel wil instellen dan kan je dat doen door de properties te scheiden met een komma:

```
.myElement {  
    ...  
    /*properties (van hover) individueel aanpassen*/  
    transition: background-color 1.5s, border-radius 0.8s, transform 1s;  
}  
  
.myElement:hover {  
    background-color: lightskyblue;  
    border-radius: 50% 50%;  
    transform: rotateY(360deg);  
}
```

### 16.2.1 VERTRAGINGEN (DELAY) EN TIMING FUNCTIONS

Aanvullend kan je ook bepalen om een vertraging in te stellen of een tijdsfunctie mee te geven. Met een vertraging, uitgedrukt in seconden, bedoelen we dat er eerst een opgegeven tijd gewacht wordt vóór de

transitie begint. In het voorbeeld hieronder duurt de transitie van de background-color dus in totaal 2,5 seconden: 1s vertraging + 1,5s om de eigenlijke overgang te maken.

```
transition: background-color 1.5s 1s, border-radius 0.8s, transform 1s; /* vertraging */
```

Alternatief kan je er voor kiezen om een tijdsfunctie mee te geven. Met een tijdsfunctie bepaal je hoe de tussenstappen berekend en weergegeven worden. In de voorbeelden hierboven hebben we steeds gezien dat de overgang gebeurt in een vloeiente beweging: de standaard optie, 'ease'.

```
transition: background-color 1.5s ease, border-radius 0.8s, transform 1s ; /* defaultfunctie */
```

Indien je wil instellen hoe de overgang gebeurt, dan kan je dat bepalen met de tijdsfuncties. In het voorbeeld hieronder geven we een functie mee om in 6 stappen de transform-property aan te passen. Geen vloeiente beweging meer, dus:

```
transition: background-color 1.5s, border-radius 0.8s, transform 1s steps(6, end);
```

Willen we zowel een tijdsfunctie als een vertraging meegeven, dan komt de vertraging achteraan te staan (*opmerking: eigenlijk kan deze overal komen te staan, zo lang ze maar gedefinieerd wordt ná de eigenlijke transitietijd*):

```
transition: background-color 1.5s, border-radius 0.8s, transform 1s steps(6, end) 1s;
```

### 1.1.1.2 TIMING FUNCTIONS

In deze cursus bespreken we niet alle timing function individueel. Hieronder enkele voorbeelden:

- Steps(4, jump-end)
- Ease, ease-in, ease-out, ease-in-out, linear, step-start, step-end
- Cubic-bezier(0.1, 0.7., 0.8, 1). Max-waarde is 1, min-waarde is 0

De laatste bepaalt in essentie de snelheid waarmee de transitie gebeurt, ten opzichte van de tijd. Een handige tool om curven te maken (en te visualiseren) vind je hier: <https://cubic-bezier.com/>;

### 16.2.2 SPECIFIEKE TRANSITION PROPERTIES

In de voorbeelden hierboven noteerden we steeds alles op één lijn. Het is echter ook mogelijk om elke property individueel aan te spreken. Lees de toewijzing van de properties als een rooster gescheiden door een komma.

```
/* Notatie met individuele properties, i.p.v. op één lijn */
transition-property: background-color, border-radius, transform;
transition-timing-function: linear, cubic-bezier(1, 0, 0, 1), steps(6, end);
transition-duration: 1s, 0.8s, 1s;
transition-delay: 1s, 0.2s, 0s;
```

### 16.3 ANIMATIONS

Met de `animations` property kunnen we bepalen hoe de animatie van de HTML-elementen exact moet gebeuren. CSS-animations zijn een elegante oplossing, omdat we geen JavaScript nodig hebben om de animatie te maken; dat werkt (mogelijk) gemakkelijker. Met behulp van de `@keyframes` at-rule kunnen we bepalen wat de start- en eindwaarden worden van het element dat we animeren. Een voorbeeld:

```


.cloud {
    overflow: hidden;
    width: 200px;
    animation-name: wind;
    animation-duration: 20s;
    animation-fill-mode: both;
    animation-timing-function: linear;
    animation-direction: both;
    animation-delay: 1s;
    animation-iteration-count: infinite;
}

@keyframes wind {
    from {
        transform: translateX(0);
    }

    to {
        transform: translateX(800px);
    }
}
```



We gebruiken dus de `animation` property om de gedeclareerde `@keyframes` in een CSS selector op te roepen. Animation kan meerdere properties bevatten:

- **animation-name:** de naam die verwijst naar de gewenste `@keyframes` die je wil gebruiken.
- **animation-duration:** de totale duurtijd van start tot eind van de animatie.
- **animation-timing-function:** de snelheid van de animatie, keuze uit vooraf gedefinieerde waarden.  
( `linear` | `ease` | `ease-in` | `ease-out` | `ease-in-out` | `cubic-bezier` ).
- **animation-delay:** eventuele vertraging vooraleer de animatie start
- **animation-iteration-count:** hoeveel keer zal deze animatie herhaald worden.
- **animation-direction:** bepaalt de richting van de animatie, start naar einde, einde naar start, of beiden.
- **animation-fill-mode:** verduidelijkt welke style er zal toegepast worden op het element na afloop van de animatie.  
( `none` | `forwards` | `backwards` | `both` )

De ‘to’ en ‘from’ keywords kan je ook beschouwen als 0% en 100%. Bekijk het volgend voorbeeld:

```
<h1>
  <span class="onzeSelectie">Introductie CSS</span>
  <span class="entry" >Animation</span>
</h1>
```

```
.onzeSelectie {
  animation-name: fadeOut;
  animation-duration: 4s;
  animation-delay: 1s;
  animation-iteration-count: infinite;
  animation-timing-function: linear;
  animation-direction: alternate;
}

@keyframes fade-out {
  0% {
    opacity: 1;
  }
  100% {
    opacity: 0;
  }
}
```

We kunnen echter ook kiezen voor een alternatieve notatie, zodat alles op één lijn genoteerd kan worden:

```
.onzeSelectie {
  animation: fade-out 4s 1s infinite linear alternate;
}
```

In bovenstaande voorbeelden hebben we telkens een enkele animation gekoppeld. Het is echter ook mogelijk om meerdere animations toe te kennen. Dat kunnen we doen door de animaties van elkaar te scheiden met een komma. We breiden het bovenstaand voorbeeld uit en passen het toe op een ander element, zodat we naast de fade-out, ook nog eens het element laten roteren:

```
<h3 class="nogEenSelectie">Gebruik van Animation</h3>

.nogEenSelectie {
  animation: fade-out 4s 1s infinite linear alternate,
             rotate 4s 1s infinite linear alternate;
}

@keyframes rotate {
  to {
    transform: rotate(180deg);
  }
}
```

Door de ‘to’ en ‘from’ te abstraheren naar ‘0%’ en ‘100%’ had je misschien al door dat we hierdoor extra tussenstappen kunnen toevoegen. Niets houdt ons namelijk tegen om tussendoor nog percenten te voorzien:

```
aside {  
    ...  
    animation: square-to-circle 2s 1s  
        cubic-bezier(0.600, -0.430, 0.500, 1.410) infinite alternate;  
}  
  
@keyframes square-to-circle {  
    0% {  
        border-radius:0 0 0 0;  
        background:coral;  
        transform:rotate(0deg);  
    }  
    25% {  
        border-radius:50% 0 0 0;  
        background:darksalmon;  
        transform:rotate(45deg);  
    }  
    50% {  
        border-radius:50% 50% 0 0;  
        background:indianred;  
        transform:rotate(90deg);  
    }  
    75% {  
        border-radius:50% 50% 50% 0;  
        background:lightcoral;  
        transform:rotate(135deg);  
    }  
    100% {  
        border-radius:50%;  
        background:darksalmon;  
        transform:rotate(180deg);  
    }  
}
```

### 16.3.1 HULPMIDDELEN

Mooie animations maken is niet evident. Het gevaar van animations is dat ze al snel overheersend zijn, waardoor ze de professionele look van je website teniet doen. Daarnaast kruipert er ook heel wat tijd in de creatie van goede animations.

Net omwille van die redenen kan het interessant zijn om bestaande CSS-‘bibliotheeken’ van animaties te gebruiken die tal van verschillende soorten animations voorzien. Dit zijn simpelweg .css-files die ter beschikking gesteld worden zodat je die (vrij) kan gebruiken. Een goed voorbeeld vind je op <https://daneden.github.io/animate.css/>.





# INTRODUCTIE TOT CSS FRAMEWORKS

Web Frontend Basics

## INHOUD

|   |          |
|---|----------|
| <b>17 INTRODUCTIE TOT CSS FRAMEWORKS</b>    | <b>3</b> |
| <b>17.1 Wat is een CSS Framework</b>        | <b>3</b> |
| <b>17.2 Welke CSS Frameworks bestaan er</b> | <b>4</b> |
| 17.2.1 Bootstrap                            | 4        |
| 17.2.2 Foundation                           | 4        |
| 17.2.3 Materialize                          | 4        |
| <b>17.3 inzoomen op Bootstrap</b>           | <b>5</b> |
| 17.3.1 Het grid systeem                     | 6        |
| 17.3.2 Bootstrap componenten                | 8        |
| alerts                                      | 8        |
| Buttons                                     | 9        |
| badge                                       | 10       |
| 17.3.3 Een volledige bootstrap-pagina       | 10       |

## 17 INTRODUCTIE TOT CSS FRAMEWORKS

### 17.1 WAT IS EEN CSS FRAMEWORK

Je hebt bij het schrijven van je CSS vast al gemerkt dat er veel werk kruip in het ontwerp van mooie componenten en dat je heel vaak dezelfde stijlregels moet schrijven. Om dat te vermijden, kun je gebruik maken van één van de vele CSS frameworks die ondertussen al bestaan.

Een CSS framework kan je zien als een verzameling van CSS stijlregels die je kan gebruiken om sneller mooiere en complexere websites te maken. Vaak gebeurt dit door een klasse aan je html element toe te voegen. Vaak bevat zo'n framework naast CSS stijlregels ook een heleboel JavaScript code die je componenten interactiever maken.

Het is natuurlijk nodig om het framework toe te voegen aan je website. Daarvoor zijn meestal twee mogelijkheden.

- 1 De nodige bestanden rechtstreeks toevoegen aan je website.
- 2 Gebruik maken van een zogenaamde CDN (Content Delivery Network).

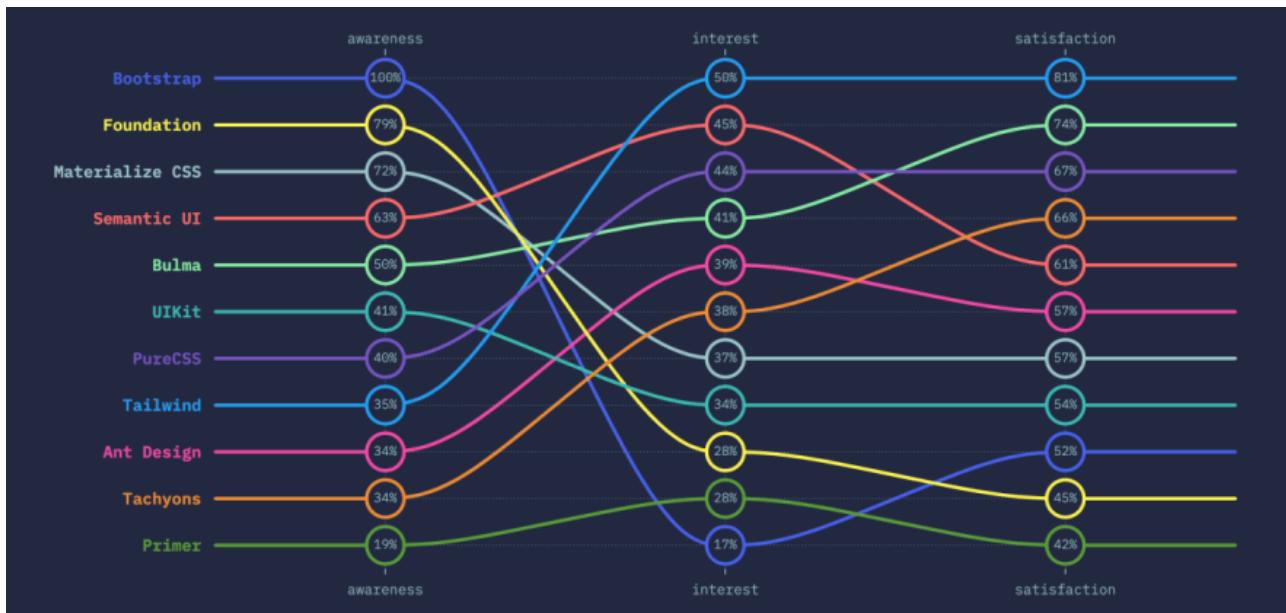
Voor de tweede optie moeten de nodige bestanden van het framework beschikbaar zijn via een CDN-url. De CSS voor het bekende Bootstrap framework kan bijvoorbeeld ingeladen worden via

```
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
      integrity="sha384-gg0yR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
      crossorigin="anonymous">
```

Let erop dat het vaak nodig zal zijn om meerdere bestanden in te laden. Zo moet de JavaScript-gedeelte van het framework altijd apart worden toegevoegd.

Het inladen via CDN heeft enkele voordelen. Zo moet je de bestanden van het framework niet op je eigen webserver zetten. Voor de populairste frameworks is het ook zo dat ze meestal al aanwezig zullen zijn in de cache van je browser (aangezien je browser die bestanden al nodig had voor een andere website). Als dat niet het geval is, zal er natuurlijk extra tijd nodig zijn om die in te laden.

## 17.2 WELKE CSS FRAMEWORKS BESTAAN ER



### 17.2.1 BOOTSTRAP

Ondertussen bestaan er al heel wat CSS frameworks. Het meest bekende is waarschijnlijk Bootstrap dat bij het schrijven van dit hoofdstuk al aan versie 4 zit, meer over Bootstrap vind je verder in dit hoofdstuk.



Deze cursus zal zeker niet alles van Bootstrap behandelen. Om meer te weten te komen, is het een goed idee om een tutorial te volgen. Kijk bijvoorbeeld eens naar de volgende link: <https://www.tutorialspoint.com/bootstrap/index.htm>.

### 17.2.2 FOUNDATION

Een ander bekend framework is Foundation (<https://foundation.zurb.com/>). Zoals de meeste frameworks is het een mobile-first responsive framework. Wat betekent dat je er websites mee kunt maken die er goed uit zien op elke schermgrootte. Daarvoor gebruikt het een CSS Grid Layout.



Er kan natuurlijk nog veel meer gezegd worden, maar om een goed idee te krijgen over wat Foundation is, is het een goed idee om er even mee te experimenteren door een tutorial te volgen. Kijk bijvoorbeeld eens naar de volgende link: <https://www.tutorialspoint.com/foundation>.

### 17.2.3 MATERIALIZE

Wie niet tevreden is over Foundation kan ook eens kijken naar Materialize. Materialize is een CSS framework dat gebaseerd is op Google Material Design (<https://material.io/>). Material design wordt omschreven als een "design"-taal en gaat veel verder dan enkel websites. Als je gebruik maakt van een android telefoon zullen veel zaken je bekend voorkomen. Materialize is een makkelijk te leren taal met een propere en moderne look. Het nadeel is de beperkte keuzevrijheid over hoe je website eruit ziet. Het is heel makkelijk om websites te herkennen die gemaakt zijn met Materialize.



Er kan natuurlijk nog veel meer gezegd worden, maar om een goed idee te krijgen over wat Materialize is, is het een goed idee om er even mee te experimenteren door een tutorial te volgen. Kijk bijvoorbeeld eens naar de volgende link: <https://www.tutorialspoint.com/materialize/index.htm>.

### 17.3 INZOOMEN OP BOOTSTRAP

Als er één framework iets grondiger behandeld moet worden, dan is het Bootstrap wel. Bootstrap is het meest bekende CSS Framework en is ontworpen door Twitter in 2011; versie 4 bestaat sinds 2014. Bij het schrijven van deze cursus wordt er geschat dat ongeveer 20% van alle websites gebruik maakt van Bootstrap. De meeste internetgebruikers komen dus dagelijks wel eens in contact met Bootstrap.

Zoals de meeste frameworks is Bootstrap een verzameling van HTML, CSS en JavaScript die gebruikt kan worden om responsive, mobile-first websites te maken.

De start HTML-code voor een bootstrap pagina ziet er als volgt uit:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
      integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
      crossorigin="anonymous">

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
      crossorigin="anonymous"></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
      integrity="sha384-U02eT0CpHqdSJQ6hJty5KVphtPhzWj9W01clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
      crossorigin="anonymous"></script>
    <script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
      integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
      crossorigin="anonymous"></script>
  </body>
</html>
```

Zoals je kan zien, laden we hier een heel aantal externe bestanden in. In het head-element laden we de Bootstrap CSS in via zijn CDN-url en in het body element laden we naast de Bootstrap JavaScript code ook enkele JavaScript bibliotheken in (jQuery en Popper) die door Bootstrap gebruikt worden.

Als je deze pagina opent in je browser, zul je zien dat er al heel wat extra opmaak toegevoegd is aan je h1-element. Door het inladen van die externe bestanden heeft Bootstrap namelijk enkele extra stijlregels toegevoegd.

### 17.3.1 HET GRID SYSTEEM

Het Bootstrap grid systeem gebruikt een reeks van containers, rijen en kolommen om de layout van je webpagina te bepalen. Het maakt gebruik van CSS Flexbox en is dus volledig *responsive*. Het belangrijkste dat je moet weten is dat in het bootstrap grid, iedere rij is opgebouwd uit 12 kolommen. Het meest eenvoudige voorbeeld dat gebruik maakt van het Bootstrap grid is dan ook het volgende:

```
<body>
  <div class="container">
    <div class="row">
      <div class="col">1</div>
      <div class="col">2</div>
      <div class="col">3</div>
      <div class="col">4</div>
      <div class="col">5</div>
      <div class="col">6</div>
      <div class="col">7</div>
      <div class="col">8</div>
      <div class="col">9</div>
      <div class="col">10</div>
      <div class="col">11</div>
      <div class="col">12</div>
    </div>
  </div>
</body>
html
```



in de browser

Zoals je kan zien, maken we gebruik van de 3 bootstrap klassen: *container*, *row* en *col*. Deze zorgen ervoor dat er al een aantal CSS regels worden toegepast. Zo zorgt de klasse *container* ervoor dat zijn inhoud gecentreerd wordt op de pagina (als je wil dat je *container* de volledige breedte gebruikt, kies dan voor de klasse *container-fluid* in plaats van *container*). Dit gebeurt door de *margin-left* en *margin-right* op *auto* te plaatsen. Kijk gerust eens via de inspector in je browser welke andere CSS stijlregels er nog worden toegepast.

Als je gewoon gebruik maakt van de klasse *col* zal iedere kolom dezelfde breedte hebben, maar bij complexere layouts wil je vaak dat sommige kolommen breder zijn. Vaak zul je ook willen dat een kolom dubbel zo breed is als de andere kolommen, denk maar aan de *fraction* die we gezien hebben bij CSS grid. Daarom heeft bootstrap naast de klasse *col* ook nog de klassen *col-1*, *col-2*, *col-3*, ..., *col-12*. Logischerwijs zullen elementen met die klassen de volgende breedte innemen. De breedte van een kolom met als klasse *col-1* zal altijd 1/12-de van de totale rij-breedte zijn, die van *col-2* zal twee keer zo breed zijn, die van *col-3* zal drie keer zo breed zijn, enz.

|        |       |       |        |       |       |       |       |       |       |       |       |  |  |  |  |
|--------|-------|-------|--------|-------|-------|-------|-------|-------|-------|-------|-------|--|--|--|--|
| col-1  | col-1 | col-1 | col-1  | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 | col-1 |  |  |  |  |
| col-2  |       | col-2 |        | col-2 |       | col-2 |       | col-2 |       | col-2 |       |  |  |  |  |
| col-3  |       |       | col-3  |       |       | col-3 |       |       | col-3 |       |       |  |  |  |  |
| col-4  |       |       | col-4  |       |       | col-4 |       |       | col-4 |       |       |  |  |  |  |
| col-5  |       |       | col-7  |       |       | col-6 |       |       | col-5 |       |       |  |  |  |  |
| col-6  |       |       | col-8  |       |       | col-4 |       |       | col-3 |       |       |  |  |  |  |
| col-7  |       |       | col-9  |       |       | col-4 |       |       | col-2 |       |       |  |  |  |  |
| col-8  |       |       | col-10 |       |       | col-3 |       |       | col-2 |       |       |  |  |  |  |
| col-9  |       |       | col-11 |       |       | col-1 |       |       | col-1 |       |       |  |  |  |  |
| col-10 |       |       | col-11 |       |       | col-1 |       |       | col-1 |       |       |  |  |  |  |
| col-11 |       |       | col-12 |       |       | col-1 |       |       | col-1 |       |       |  |  |  |  |

Normaal gezien zal de som van alle kolombreedtes dus gelijk zijn aan 12. Het is ook mogelijk om te mengen met elementen die gewoon klasse `col` hebben. Bootstrap zal eerst de breedte berekenen van de elementen die een breedte hebben meegekregen en de resterende breedte vervolgens gelijk verdelen over de elementen met als klasse `col`.

|     |       |     |       |     |
|-----|-------|-----|-------|-----|
| col | col-6 | col |       |     |
| col | col-2 | col |       |     |
| col | col-4 | col | col-3 | col |

Wat ook interessant om te weten is, is dat van zodra je aan meer dan 12 komt voor de breedtes van de kolommen, de extra kolom-elementen op een nieuwe rij zullen beginnen. Denk hierbij aan de CSS-property `flex-wrap` in flexbox.

Binnenin bootstrap zijn er ook 5 zogenaamde *breakpoints* ingebouwd die je kunt gebruiken om de grid aan te passen aan de hand van de schermbreedte van de gebruiker. Deze *breakpoints* zijn:

- extra small (xs): schermgrootte kleiner dan 576px
- small (sm): schermgrootte is groter dan 576px
- medium (md): schermgrootte is groter dan 768px
- large (lg): schermgrootte is groter dan 992px
- extra large (xl): schermgrootte is groter dan 1200px

Laat ons eens kijken naar hoe deze gebruikt kunnen worden binnennin Bootstrap.

```
<div class="container">
  <div class="row">
    <div class="col-sm-6">Column 1</div>
    <div class="col-sm-6">Column 2</div>
  </div>
</div>
```

In plaats van `col-*` maken we hier gebruik van `col-sm-*`. Dat betekent dat we vanaf een schermgrootte groter dan 576px, een rij met twee even grote kolommen krijgen. Aangezien er niets gedefineerd is voor kleinere schermen, zal daar de standaard van `col-12` genomen worden en zullen onze twee kolommen onder elkaar verschijnen (aangezien 12 de maximum breedte op 1 rij is).

Column 1

Column 2

*schermgrootte is groter dan 576px*

Column 1

Column 2

*schermgrootte is kleiner dan 576px*

Het is hierbij belangrijk op te merken dat grotere breakpoints lagere overschrijven. Als je dus een `col-sm` en een `col-lg` definieert, zal er van zodra het scherm groter is dan 992px niet meer naar de `col-sm` klasse gekeken worden.

### 17.3.2 BOOTSTRAP COMPONENTEN

De grote sterkte van Bootstrap is de aanwezigheid van heel veel voorgedefineerde componenten. Hier zullen we er enkele opsommen.

#### 1.1.1.1 ALERTS

Een alert kan gebruikt worden om feedback te geven aan de gebruiker van je website. Zo kun je bijvoorbeeld een alert-message gebruiken als een formulier verkeerd is ingevuld, als een actie succesvol is getoond, of gewoon om informatie aan de gebruiker te geven. Een alert kun je in veel verschillende kleuren te zien krijgen. Meestal hangt de kleur af van wat je de gebruiker wil vertellen. Zo geef je een foutmelding best in het rood weer en een geslaagde actie in het groen. De kleuren worden ingesteld door 1 van de vele CSS `alert-*` klassen.

```
<div class="alert alert-primary" role="alert">
  This is a primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert">
  This is a secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  This is a success alert—check it out!
</div>
<div class="alert alert-danger" role="alert">
  This is a danger alert—check it out!
</div>
<div class="alert alert-warning" role="alert">
  This is a warning alert—check it out!
</div>
<div class="alert alert-info" role="alert">
  This is a info alert—check it out!
</div>
<div class="alert alert-light" role="alert">
```

This is a primary alert—check it out!

This is a secondary alert—check it out!

This is a success alert—check it out!

This is a danger alert—check it out!

This is a warning alert—check it out!

This is a info alert—check it out!

This is a light alert—check it out!

This is a dark alert—check it out!

*In de browser*

```
This is a light alert—check it out!
</div>
<div class="alert alert-dark" role="alert">
  This is a dark alert—check it out!
</div>
```

De HTML-code

### 1.1.1.2



Meer info over alerts kan gevonden worden op  
<https://getbootstrap.com/docs/4.4/components/alerts/>

### 1.1.1.3 BUTTONS

Bootstrap heeft ook enkele voorgedefinieerde klassens die je kan toepassen op het button-element. Gebruik daarvoor de *btn* klasse. Daarnaast zijn er ook nog enkele klassen die je kan gebruiken om de kleur en de grootte in te stellen.

Een standaard knop zonder opmaak ziet er zo uit.

```
<button type="button" class="btn">
  Klik hier
</button>
```

Klik hier

Met de klassen *btn-primary*, *btn-secondary*, *btn-success*, *btn-danger*, *btn-warning*, *btn-info*, *btn-light*, *btn-dark* en *btn-link* kun je een semantische betekenis aan de knop geven en deze zal onmiddellijk ook de kleur van je knop instellen. De standaard kleuren voor deze klassen zijn de volgende:

Primary Secondary Success Danger Warning Info Light Dark Link

Als je liever een knop zonder achtergrondkleur wil, kun je bovenstaande klassen vervangen door *btn-outline-primary*, *btn-outline-secondary*, ...

Primary Secondary Success Danger Warning Info Light Dark

Daarnaast kun je de grootte van de knop ook instellen met de klassen *btn-sm* en *btn-lg*.

small normal large



Meer info over buttons kan gevonden worden op  
<https://getbootstrap.com/docs/4.4/components/buttons/>

#### 1.1.1.4 BADGE

Een badge kan gebruikt worden om extra informatie in weer te geven. Vaak wordt er een getal in weergegeven dat bijvoorbeeld het aantal ongelezen e-mails bevat.

```
<body style="margin: 20px;">
<button type="button" class="btn btn-primary">
    Inbox <span class="badge">4</span>
</button>
```

Inbox 4

Standaard wordt de achtergrondkleur overgenomen van het bovenliggen element, maar bootstrap heeft ook al enkele voorgedefinieerde klassen om de kleur afzonderlijk in te stellen.

```
<span class="badge badge-primary">4</span>
<span class="badge badge-secondary">4</span>
<span class="badge badge-success">4</span>
<span class="badge badge-danger">4</span>
<span class="badge badge-warning">4</span>
<span class="badge badge-info">4</span>
<span class="badge badge-light">4</span>
<span class="badge badge-dark">4</span>
```



#### 1.1.1.5



Meer info over badges kan gevonden worden op <https://getbootstrap.com/docs/4.4/components/badge/>

### 17.3.3 EEN VOLLEDIGE BOOTSTRAP-PAGINA

Om een aantal andere componenten te behandelen, zullen we een voorbeeld pagina opbouwen. Op het einde van dit hoofdstuk zou deze er zo moeten uit zien.

The screenshot shows a website layout with a dark header bar containing navigation links: 'Graduaat programmeren', 'Onze troeven', 'Curriculum', and two search bars. Below the header are six cards arranged in a grid:

- Programming Basics** by Deboosere D. A card describing the module's goal of providing a deep insight into .Net Framework and teaching how to develop programs.
- Programming Advanced** by Deboosere D. A card describing the module's goal of building on insights from Programming basics.
- Web Frontend Basics** by Soete B. A card describing the module's goal of teaching correct design for online user environments.
- Databases** by Di Marco M. A card describing the module's goal of teaching how to set up and query databases.
- Basic IT Skills** by Cools K. A card describing the module's goal of teaching logical and procedural thinking through simple database operations.
- Continuous Integration** by François J. A card describing the module's goal of teaching CI basics and using Git for version control.

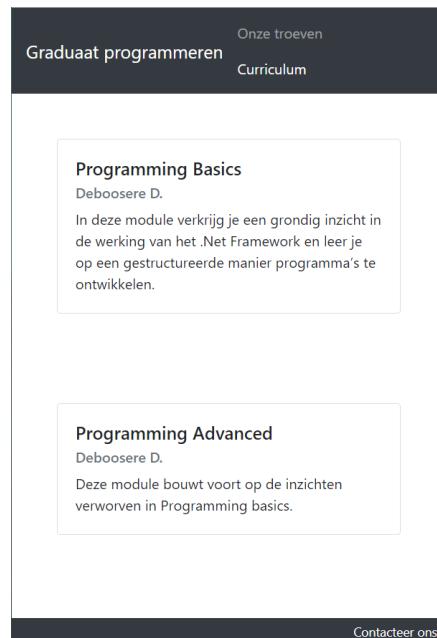
Contacteer ons

Laat ons beginnen met de navigatiebalk. De volledige html-code daarvoor ziet er zo uit:

```
<nav class="navbar navbar-expand-sm navbar-dark bg-dark">
  <a class="navbar-brand" href="#">Graduaat programmeren</a>
  <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="#">Onze troeven</a>
    </li>
    <li class="nav-item dropdown">
      <a class="nav-link active" href="#" data-toggle="dropdown">Curriculum</a>
      <div class="dropdown-menu">
        <a class="dropdown-item" href="#">Programming Basics</a>
        <a class="dropdown-item" href="#">Programming Advanced</a>
        <a class="dropdown-item" href="#">Web Frontend Basics</a>
        <a class="dropdown-item" href="#">Databases</a>
        <a class="dropdown-item" href="#">Continuous Integration</a>
        <a class="dropdown-item" href="#">Basic IT Skills</a>
      </div>
    </li>
  </ul>
  <form class="form-inline d-none d-md-inline">
    <input class="form-control mr-sm-2" placeholder="Zoek">
    <button class="btn btn-outline-light" type="submit">Zoek</button>
  </form>
</nav>
```

Je ziet dat we gebruik maken van een heleboel voorgedefinieerde css-klassen. Het *nav*-element zelf heeft de klasse *navbar* (dit is standaard voor alle navigatie-balken). We gebruiken de klassen *navbar-dark* en *bg-dark* om de kleuren in te stellen.

Daarnaast is er ook de klasse *navbar-expand-sm* die gebruikt wordt om onze navigatie aan te passen aan de schermgrootte. Van zodra de schermgrootte de ingestelde breedte voor small (*sm*) devices heeft bereikt, zullen de menu-items van onze navigatiebalk onder elkaar komen te staan waardoor de mobiele versie van onze website er als volgt uit ziet:



Het element met klasse `navbar-brand` wordt niet als een menu-item gezien, maar daar kun je bijvoorbeeld de naam van je website plaatsen. De werkelijke menu-items komen dan in het `ul`-element met als klasse `navbar-nav`. Deze hebben we ook nog de klasse `mr-auto` gegeven die de `margin-right` property van dat element op `auto` plaatst. Dit zorgt er voor dat onze zoekbalk rechts komt te staan. De menu-items plaatsen we dan binnen een `li`-element met als klasse `nav-item` in een `a`-element met als klasse `nav-link`.

In onze navigatiebalk hebben we van het Curriculum item een dropdown menu gemaakt.



Dit hebben we gedaan door de klasse `dropdown` toe te voegen aan het `li`-element. Daarnaast hebben we ook een `div`-element gecreëerd met de volgende inhoud.

```
<div class="dropdown-menu">
    <a class="dropdown-item" href="#">Programming Basics</a>
    <a class="dropdown-item" href="#">Programming Advanced</a>
    <a class="dropdown-item" href="#">Web Frontend Basics</a>
    <a class="dropdown-item" href="#">Databases</a>
    <a class="dropdown-item" href="#">Continuous Integration</a>
    <a class="dropdown-item" href="#">Basic IT Skills</a>
</div>
```

Dus 1 `a`-element voor ieder item van ons dropdown menu.

Als laatste hebben we ook nog een zoekbalk voorzien in ons navigatie-menu. Dit hebben we gedaan door de volgende html-code toe te voegen:

```
<form class="form-inline d-none d-md-inline">
    <input class="form-control mr-sm-2" placeholder="Zoek">
    <button class="btn btn-outline-light" type="submit">Zoek</button>
</form>
```

De klasse `form-inline` spreekt voor zich. Deze zorgt er gewoon voor dat ons formulier *inline* geplaatst wordt. De klassen `d-none` en `d-md-inline` zorgen er dan voor dat de `display` property van het element standaard op `none` staat, en pas zichtbaar wordt vanaf je scherm een medium (`md`) grootte heeft.

Nu onze navigatiebalk klaar is, kunnen we over gaan naar de inhoud van de pagina. Hiervoor gebruiken we de volgende html-code.

```
<main class="container-fluid">
    <div class="row">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Programming Basics</h5>
                <h6 class="card-subtitle mb-2">Deboosere D.</h6>
                <p class="card-text">
                    In deze module verkrijg je een grondig inzicht in de werking van het .Net Framework en leer je op een gestructureerde manier programma's te ontwikkelen.</p>
                </div>
```

```

        </div>
    <div class="card">
        <div class="card-body">
            <h5 class="card-title">Programming Advanced</h5>
            <h6 class="card-subtitle mb-2">Deboosere D.</h6>
            <p class="card-text">
                Deze module bouwt voort op de inzichten verworven in Programming basics.</p>
        </div>
    </div>
    <div class="card">
        <div class="card-body">
            <h5 class="card-title">Web Frontend Basics</h5>
            <h6 class="card-subtitle mb-2">Soete B.</h6>
            <p class="card-text">
                In de module Web Frontend Basics wordt de nadruk gelegd op het correct ontwerpen van online  
gebruikersomgevingen.</p>
        </div>
    </div>
    <div class="card">
        <div class="card-body">
            <h5 class="card-title">Databases</h5>
            <h6 class="card-subtitle mb-2">Di Marco M.</h6>
            <p class="card-text">
                In de module Databases leer je hoe je een database moet opstellen en bevragen.</p>
            </div>
        </div>
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Basic IT Skills</h5>
                <h6 class="card-subtitle mb-2">Cools K.</h6>
                <p class="card-text">
                    In de module Basic IT Skills wordt nadruk gelegd op het logisch en procedureel denken,  
door eenvoudige uitwerkingen van basisbewerkingen op een computersysteem (getallenken  
nis).</p>
            </div>
        </div>
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">Continuous Integration</h5>
                <h6 class="card-subtitle mb-2">François J.</h6>
                <p class="card-text">
                    In de module CI Basis leer je werken met het vrij gedistribueerd versiebeheer systeem  
genaamd  
Git. Deze module legt de basis voor het efficiënt samenwerken aan dezelfde programmaco  
de.</p>
            </div>
        </div>
    </div>
</main>
```

Om alles er goed te laten uit zien, voegen we ook nog de volgende css toe:

```

.card {
    width: 400px;
    margin: 50px;
}

.row {
    justify-content: space-around;
}
```

Onze inhoud staat dus in een *main*-element dat een bootstrap container is. We hebben de klasse op *container-fluid* ingesteld, zodat de volledige breedte van ons scherm wordt gebruikt. We hebben daarin dan 1 rij die 6 *div*-elementen met klasse *card* bevat.

Een card is een veelgebruikte component binnen bootstrap en kun je gebruiken om bepaalde inhoud afgezonderd weer te geven. Onze pagina bestaat dan ook uit 6 cards, die elk overeen komen met 1 vak binnen de opleiding programmeren.

```
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Programming Basics</h5>
    <h6 class="card-subtitle mb-2">Deboosere D.</h6>
    <p class="card-text">
      In deze module verkrijg je een grondig inzicht in de werking van het .Net Framework en
      leer je op een gestructureerde manier programma's te ontwikkelen.</p>
    </div>
  </div>
```

We hebben in onze CSS onze cards een breedte van 400px gegeven en de marge ingesteld op 50px zodat niet alles samenplakt. Aangezien onze row een flex-container is, kunnen we makkelijk de *justify-content* property op *space-around* plaatsen zodat onze cards mooi verdeeld op de pagina staan.

Dan hebben we ook nog de footer. Deze heeft de volgende html-gekregen.

```
<footer class="page-footer">
  <button class="btn btn-dark btn-sm" type="button" data-toggle="modal" data-
target="#contactModal">
    Contacteer ons
  </button>
</footer>
```

En de volgende css

```
footer {
  background-color: #343a40;
  position: fixed;
  bottom: 0px;
  width: 100%;
  color: white;
  text-align: right;
}
```

Onze footer bevat 1 knop die een pop-up met een contactformulier zal laten verschijnen. Na een klik op de knop, ziet de pagina er als volgt uit.

**Programming Basics**

Deboosere D.

In deze module verkrijg je een grondig inzicht in de werking van het .Net Framework en leer je op een gestructureerde manier programma's te ontwikkelen.

**Databases**

Di Marco M.

In de module Databases leer je hoe je een database moet opstellen en bevragen.

**Contacteer ons**

Email address

Opleidingsonderdeel

Opmerkingen

x

Sluit

Verzend

**Web Frontend Basics**

Soete B.

In de module Web Frontend Basics wordt de nadruk gelegd op het correct ontwerpen van online gebruikersomgevingen.

**Continuous Integration**

François J.

In de module CI Basis leer je werken met het vrij gedistribueerd versiebeheer systeem genaamd Git. Deze module legt de basis voor het efficiënt samenwerken aan dezelfde programmacode.

Contacteer ons

Door het attribuut *data-toggle* met value *modal* toe te voegen, samen met een attribuut *data-target*, kunnen we een dialoogvenster tonen. Hierbij is het belangrijk dat er een div bestaat wiens id overeen komt met de waarde van het *data-target*. Voor ons voorbeeld ziet deze div er zo uit:

```
<div class="modal fade" id="contactModal">
  <div class="modal-dialog">
    <form class="modal-content" method="POST" action="http://jkorpela.fi/cgi-bin/echo.cgi">
      <div class="modal-header">
        <h5 class="modal-title">Contacteer ons</h5>
        <button type="button" class="close" data-dismiss="modal">
          <span>&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <div class="form-group">
          <label for="emailField">Email address</label>
          <input name="email" type="email" class="form-control" id="emailField" placeholder="name@example.com">
        </div>
        <div class="form-group">
          <label for="vakField">Opleidingsonderdeel</label>
          <select name="vak" class="form-control" id="vakField">
            <option>Programming Basics</option>
            <option>Programming Advanced</option>
            <option>Web Frontend Basics</option>
            <option>Databases</option>
            <option>Basic IT Skills</option>
            <option>Continuous Integration</option>
          </select>
        </div>
        <div class="form-group">
          <label for="opmerkingenVak">Opmerkingen</label>
          <textarea name="opmerking" class="form-control" id="opmerkingenVak" rows="3"></textarea>
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Sluit</button>
        <button type="submit" class="btn btn-primary">Verzend</button>
      </div>
    </form>
  </div>
</div>
```

Hierbij moeten er een aantal dingen verteld worden. Door de klasse *fade* toe te voegen aan onze modal, zorgen we ervoor dat onze modal geleidelijk aan verschijnt en verdwijnt. Het kruisje in de rechterbovenhoek dat we kunnen gebruiken om onze modal te sluiten ziet er als volgt uit in html.

```
<button type="button" class="close" data-dismiss="modal">
  <span>&times;</span>
</button>
```

Belangrijk hierbij is het attribuut *data-dismiss* met als waarde *modal*. Daarnaast kun je ook zien dat we elk form-element binnenin een *div* met als klasse *form-group* hebben geplaatst. Dit is strikt genomen niet nodig, maar het wordt wel aangeraden. Om wat opmaak aan onze form-elementen toe te voegen hebben we ze ook de klasse *form-control* gegeven.

Op deze manier kom je met behulp van bootstrap snel een mooi gestijlde webpagina zonder dat je heel veel css moet schrijven.

