



Cursus Afstandstraject Web Essentials

Lector: Kimberly Willems

Inhoud

Inleiding.....	8
Opbouw en talen.....	8
HTML	8
CSS	8
HTTP en URL's.....	9
HTTP	9
URL	9
Benodigde software en accounts.....	10
JetBrains WebStorm	10
Chrome DevTools	11
Git SCM.....	11
GitHub	12
Git en GitHub	13
Algemene werking van Git en GitHub.....	14
Werkdirectory en Lokale Git repository	14
Wijzigingen aan de Lokale Git repository vastleggen	14
Lokale Git Repository en Remote Git Repository	15
Git software.....	16
Werken met Git Bash (Lokaal).....	17
Algemene commando's.....	17
Credentials ingeven	18
Een Git repository aanmaken of klonen	18
De status van een Git repository bekijken.....	19
Bestanden stagen en committen.....	19
De commit-geschiedenis bekijken.....	21
Harde reset naar een specifieke commit.....	21
Specifieke bestanden unstagen en unmodifieden	22

Bestanden uitsluiten	22
Werken met GitHub én Git Bash (Lokaal en remote)	22
Een remote Git repository aanmaken	23
De remote Git repository klonen als lokale Git repository.....	25
Terminalcommando's	26
Basis HTML.....	28
Starten met HTML	28
Elementen en tags	29
Elementen nesten	30
Attributen.....	30
Commentaar	31
Conventies en validatie	31
Structuur	32
Blokelementen en inline elementen	32
Structuurelementen	33
Hyperlinks	35
Absolute koppelingen.....	36
Relatieve koppelingen.....	36
Interne koppelingen.....	37
Koppeling naar een e-mailadres	39
Lijsten	39
Geneste lijsten	40
Afbeeldingen.....	41
Het figure-element.....	42
Afbeelding als koppeling	43
Youtube-video's	43
Basis CSS	44
Syntax van CSS.....	44
Commentaar	45

Toepassen van CSS	45
Lokale of inline stijl	45
Globale stijl of blokstijl	45
Gelinkte stijl met apart stijlblad.....	46
Volgorde van opmaak	46
Selectors.....	47
HTML tags.....	47
Classes	47
Id's.....	49
Speciale selectors	49
Eenheden	53
Opmaak	53
Kleuren.....	53
Tekstopmaak.....	57
Het box-model.....	60
Lijsten.....	65
Afbeeldingen	66
Tabellen.....	68
Structuur van een tabel.....	68
De tabelrij	68
De cellen	68
Het bijschrift	69
CSS	70
Rijgroepen	72
Overspannen van kolommen en rijen	73
Opeenvolgende tabellen.....	74
Geneste tabellen.....	75
CSS3 voor tabellen.....	76
Basis structuur en positionering	77

De structuurelementen	77
Stijleigenschap display	78
Stijleigenschap box-sizing	79
Standaardopmaak resetten	80
Stijleigenschap position	80
Absolute positionering	81
Relatieve positionering	83
Gefixeerde (vaste) positionering	85
Stijleigenschap z-index	86
Stijleigenschap overflow	88
Rekenfunctie calc()	89
Responsive webdesign.....	92
Responsiviteit controleren	93
Viewport.....	93
Minimale en maximale hoogtes en breedtes	94
Media queries	96
Media queries gebruiken in CSS.....	96
Media queries gebruiken in HTML	98
Werken met meeteenheid rem	99
Flexbox.....	100
Flex-container	100
Display	101
Flex-direction.....	101
Flex-wrap	102
Flex-flow	103
Justify-content	104
Align-items	105
Align-content	106
Flex items	107

Order	107
Flex-grow	108
Flex-shrink.....	109
Flex-basis.....	110
Flex	111
Align-self	112
Bootstrap	113
Bootstrap handleiding	114
Starten met Bootstrap.....	115
Bootstrap bestanden downloaden.....	115
Bootstrap bestanden insluiten.....	117
Standaardopmaak (reboot) van Bootstrap.....	118
Het Bootstrap gridsysteem	119
Containers.....	119
Rijen	120
Kolommen	121
Kolommen en rijen nesten	129
Padding en margin	130
Gutter	132
Webformulieren	134
Het form-element	135
Form-attributen.....	135
Labels	138
Invoerelementen	139
Input-elementen	139
Select-elementen	146
Tekstvak met meerdere regels.....	147
Knoppen.....	148
Structuur toevoegen	149

Toegankelijkheid verhogen	150
Extra	151
Meta-tags en meta-gegevens	151
Meta-tag met attribuut charset	151
Meta-tag met attributen name en content	151
Meta-tags via Open Graph protocol	152
Website online plaatsen.....	153
Webhosting	153
Hostingproviders	154
Domeinnaam.....	154
FTP	155

Inleiding

Voor we aan de slag kunnen gaan met het ontwikkelen van webpagina's, moeten we een stukje theorie doorlopen over de verschillende webtalen, de werking van het internet en de adressering van de webpagina's. Daarnaast installeren we in dit hoofdstuk ook alle benodigde software voor dit opleidingsonderdeel en maken we een account aan op GitHub.

Opbouw en talen

Als front-end webdeveloper gebruiken we vaak verschillende webtalen om onze webpagina op te bouwen. Deze talen hebben elk een specifieke functie.

Dit academiejaar gaan we, binnen 'Web Essentials' voornamelijk focussen op:

- HTML
- CSS



HTML

HTML staat voor 'HyperText Markup Language' en is een markeertaal. Met deze taal kunnen we een tekstdocument voorzien van annotaties. De annotaties zorgen voor de structuur van de inhoud.

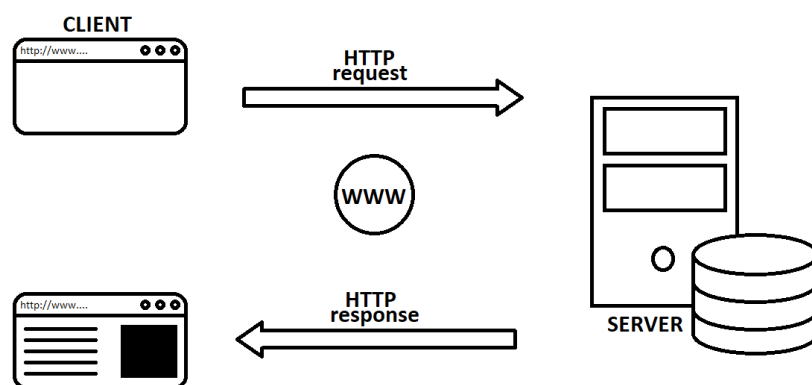
CSS

CSS staat voor 'Cascading Style Sheets' en beschrijft de opmaak of vormgeving van de elementen op een webpagina. CSS geeft de mogelijkheid om de vormgeving van webpagina's los te koppelen van hun feitelijke inhoud en centraal vast te leggen.

HTTP en URL's

HTTP

HTTP staat voor 'Hypertext Transfer Protocol' en is een vraag-antwoord protocol voor de communicatie tussen een client en een server. Dit protocol werkt platformonafhankelijk en wordt zowel op het wereldwijde web als op lokale netwerken gebruikt.



URL

URL staat voor 'Uniform Resource Locator' en geeft de unieke locatie van een document weer in de vorm van een adres of pad. Een URL kan volgende onderdelen bevatten:

- Protocol (http, https, ftp, file, ...)
- Authenticatiegegevens (gebruikersnaam en wachtwoord)
- Host (de domeinnaam of het IP-adres)
- Poortnummer
- Padnaam (hiërarchische data)
- Query (niet-hiërarchische data)
- Fragmentidentificer (subdocument of specifiek onderdeel van het document)

<u>http://www.spelconsoles.be:443/sony/playstation?model=ps4#specificaties</u>	
http	Protocol
www.spelconsoles.be	Host
443	Poortnummer
sony/playstation	Padnaam
model=ps4	Query
specificaties	Fragmentidentificer

Benodigde software en accounts

JetBrains WebStorm

Voor de lessen van Web Essentials gaan we gebruik maken van JetBrains WebStorm. Dit is een gespecialiseerde Integrated Development Environment (IDE) voor webontwikkelaars om broncode te schrijven.

OPGELET! Indien je liever met een andere gespecialiseerde IDE of teksteditor werkt, mag je deze gerust gebruiken. Let wel op, de ondersteuning van de lector beperkt zich dan enkel tot de code. Alle video-voorbeelden werden door de lector ook in bovenstaande teksteditor gemaakt.



Chrome DevTools

Chrome DevTools zit standaard ingebouwd in de webbrowser 'Google Chrome'. Het bevat een uitgebreide set van ontwikkelaarstools die rechtstreeks in Google Chrome zijn ingebouwd. Via de Chrome DevTools kunnen we:

- Onze code bekijken
- Onze code bewerken
- Problemen snel diagnosticeren
- ...

We kunnen de Chrome DevTools op meerdere manieren openen in onze Webbrowser:

- Ga naar het menu van de browser. Klik in het menu op 'Meer hulpprogramma's' en kies daarna 'Hulpprogramma's voor ontwikkelaars'.
- Plaats de muisaanwijzer binnen de geopende webpagina in de browser. Open met de rechtermuisknop het snelmenu en kies 'Inspecteren'.
- Druk op de knop F12.
- Druk op de knoppen Ctrl+shift+I (of Cmd+Opt+I voor Mac-gebruikers).



Git SCM

Git is een *Distributed* Version Control System (DVCS) oftewel een *vrij gedistribueerd versiebeheersysteem*. Git wordt voornamelijk gebruikt door developers om:

- wijzigingen aan bestanden bij te houden;
- wijzigingen in bestanden ongedaan te maken;
- Online samen te werken.

Via Git worden, per versie, enkel de wijzigingen in de bestanden opgeslagen. Dit zorgt voor een grote besparing aan data.

Git SCM (Git Bash) werkt via commando's die we ingeven in een terminal. Dit is de meest universele (softwareonafhankelijke) manier om met Git aan de slag te gaan.



GitHub

Met GitHub kunnen we projecten met andere mensen delen. Deze projecten, en alle versies van deze projecten, zijn dan beschikbaar via het internet voor een beperkte groep van betrokkenen of iedereen ter wereld (afhankelijk van de instellingen).

Om GitHub te kunnen gebruiken, hebben we een GitHub-account nodig.



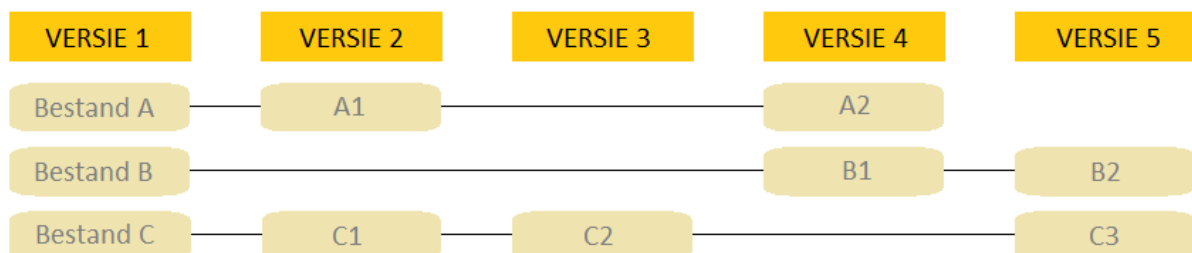
Git en GitHub

Git is een *Distributed* Version Control System (DVCS) oftewel een *vrij gedistribueerd versiebeheersysteem*. Git wordt voornamelijk gebruikt door developers om:

- wijzigingen aan bestanden bij te houden;
- wijzigingen in bestanden ongedaan te maken;
- Online samen te werken.

Via Git worden enkel de wijzigingen in de bestanden opgeslagen. Dit zorgt voor een grote besparing aan data. In een project zullen er vaak meer bestanden hetzelfde blijven tussen twee versies dan dat er bestanden wijzigen. Door enkel de gewijzigde bestanden te bewaren en een link te behouden naar de vorige versie van de niet gewijzigde bestanden moet er veel minder data worden opgeslagen.

Wijzigingen die we aan de bestanden doorvoeren overheen de versies:



Wijzigingen opgeslagen in de DVCS:



Algemene werking van Git en GitHub

Werkdirectory en Lokale Git repository

Tijdens het werken aan een project, maken we gebruik van een *werkdirectory*. In deze werkdirectory gaan we wijzigingen aan bestanden maken.

Elke keer dat we een punt in ons project bereiken dat we willen vastleggen, gaan we een snapshot van de werkdirectory *committen* naar onze *Git repository*. De lokale Git repository is een '*hidden directory*' (*staat een puntje voor de mapnaam*) op onze computer en bevat de volledige geschiedenis van de wijzigingen aan de bestanden.

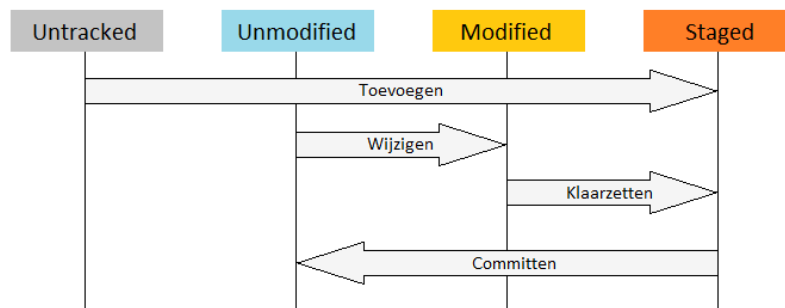
Wijzigingen aan de Lokale Git repository vastleggen

Elk bestand in de werkdirectory heeft ofwel de status *tracked* ofwel de status *untracked*.

- *Tracked*: de bestanden in de werkdirectory waren in de voorgaande snapshot aanwezig. Git weet af van het bestaan van deze bestanden.
- *Untracked*: de bestanden in de werkdirectory waren in de voorgaande snapshot niet aanwezig en zijn momenteel ook niet staged.

De tracked bestanden kunnen zich bevinden in de toestanden:

- *Unmodified* (ongewijzigd in de werkdirectory)
- *Modified* (gewijzigd in de werkdirectory)
- *Staged* (klaargezet voor de volgende commit)
- *Committed* (aanpassingen zijn doorgegeven aan de Git repository)

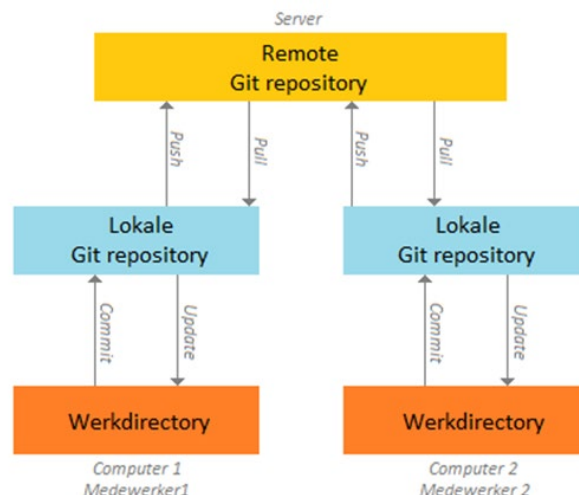


Wanneer we een werkdirectory voor het eerst *committen*, krijgen alle bestanden de status *tracked* en toestand *unmodified*, omdat we ze hebben ingecheckt, maar nog niets hebben gewijzigd. Zodra we de bestanden gaan wijzigen, gaat Git ze zien als *modified*. Bepaalde gewijzigde bestanden gaan we *stagen* en daarna *committen*.

De cyclus begint na het *committen* weer opnieuw.

Lokale Git Repository en Remote Git Repository

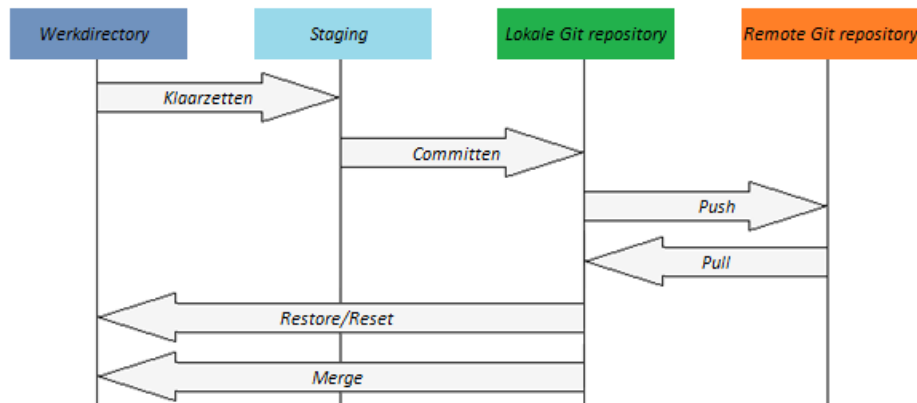
Naast onze lokale Git repository kunnen we ook gebruikmaken van een remote Git repository. Een remote Git repository is een Git repository vanop afstand.



Dit heeft enkele voordelen:

- Indien een computer of de server crasht, zijn we de bestanden niet kwijt. Deze staat namelijk op meerdere plaatsen.
- We kunnen eenvoudig samenwerken met meerdere personen aan één project.
- We kunnen sneller feedback vragen aan collega's.
- We hebben meer bewegingsvrijheid om nieuwe code lokaal toe te voegen en uit te proberen zonder dat dit meteen effect heeft op het project in de remote repository. Daarna kunnen we, wanneer we tevreden zijn, kiezen om alles in één keer te pushen.
- ...

De wijzigingen die we doen in onze lokale Git repository worden niet meteen doorgevoerd in onze remote Git repository. Omgekeerd wordt onze lokale Git repository niet automatisch geüpdatet met wijzigingen van onze remote repository. Om onze lokale Git repository en remote Git repository in sync te houden, moeten pushen, pullen en mergen.



Git software

Er bestaan verschillende Git-programma's die ons toelaten om op een vrij eenvoudige manier onze Git repository te beheren. Alle Git-programma's werken, al dan niet achterliggend, met behulp van dezelfde Git-terminalcommando's.

Wij gaan tijdens de lessen voornamelijk werken met *Git Bash* en *GitHub*.

Git Bash is een programma waarmee we rechtstreeks kunnen werken met de verschillende Git-terminalcommando's.

GitHub is een visuele *Git repository hosting service*. Via Github kunnen we gratis een openbare remote Git repository aanmaken. GitHub is handig wanneer we:

- onze code willen delen met de rest van de wereld;
- anderen willen toelaten om mee te werken aan onze code;
- anderen de mogelijkheid willen geven om zelf een nieuw project te starten op basis van onze code.



Werken met Git Bash (Lokaal)

Algemene commando's

Navigeren naar de juiste map:

```
cd 'pad/naar/map'  
cd 'C:/Users/KW/Documents/DVO/Web Essentials/Week 5-6/Project'
```

De inhoud van een map weergeven:

```
ls //Inhoud van de huidige map weergeven  
ls 'pad/naar/map' //Inhoud van een andere map weergeven
```

Een nieuwe map aanmaken:

```
mkdir 'Nieuwe map'  
mkdir 'pad/naar/Nieuwe map'
```

De versie van Git Bash controleren:

```
git --version
```

De handleiding van Git over een specifieke actie openen:

```
git config --help           //De git-config handleiding  
git add --help              //De git-add handleiding
```

Credentials ingeven

Vooraleer we aan de slag kunnen gaan, moeten we onze credentials ingeven in Git. De credentials worden gebruikt om bij te houden wie, welke aanpassingen heeft gemaakt. Om onze credentials in te stellen via de terminal, voeren we volgende commando's uit:

```
git config --global user.name "VoornaamNaamPXL"  
git config --global user.email voornaam.naam@student.px1.be
```

Een Git repository aanmaken of klonen

Een nieuwe Git repository aanmaken in onze werkdirectory:

```
git init
```

Een bestaande Git repository klonen:

```
git clone 'pad naar/bestaande Git repository' 'pad/naar/map voor kloon'
```

De status van een Git repository bekijken

Status van de Git repository controleren:

```
git status
```

Status (antwoord na bovenstaand commando) van een lege Git repository:

```
No commits yet  
  
nothing to commit (create/copy files and use "git add" to track)
```

We voegen een README.txt bestand toe aan de werkdirectory. De status (antwoord na bovenstaand commando) van de Git repository met een untracked README.txt bestand is:

```
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    README.txt  
  
nothing added to commit but untracked files present (use "git add" to track)
```

Bestanden stagen en committen

Nu we een Git repository hebben, kunnen we beginnen met werken. Dit doen we zoals we dat gewoon zijn. Onze werkdirectory is onze Webstorm-projectmap op de computer en de bestanden in de Projectmap maken we aan via Webstorm.

We gaan stagen en daarna committen naar onze Git repository:

- Wanneer we zelf vinden dat het tijd wordt om een back-up te maken
- Op het moment dat we een bug hebben opgelost
- Na het toevoegen van een nieuwe feature
- ...

We moeten dan eerst de bestanden stagen via volgend commando:

```
git add 'Bestandsnaam of mapnaam'           //alle bestanden in de map
                                              worden mee gestaged.
git add 'index.html'
git add *                                     //alle bestanden in het project
                                              worden gestaged.
```

Na het stagen kunnen we opnieuw de status van de Git repository controleren:

```
git status
```

Status (antwoord na bovenstaand commando) van de Git repository:

```
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html
```

De laatste stap is nu om alle wijzigingen te committen via volgend commando:

```
git commit -m 'Het commit bericht met de gemaakte wijzigingen'
```

Na het uitvoeren krijgen we automatisch volgende informatie te zien:

```
[master 4576055] Nieuwe startpagina toevoegen
1 file changed, 10 insertions(+)
create mode 100644 index.html
```

De commit-geschiedenis bekijken

Het kan handig zijn om, na een aantal commits of na het klonen van een Git repository, de commit-geschiedenis eens te bekijken. Dit kunnen we doen via volgend commando:

```
git log                //uitgebreide log
git log --oneline      //één lijn per commit
```

Logging (antwoord na bovenstaand commando) van de Git repository:

```
commit 4576055edd120e55e0b47d690f24348e7bd4ede5 (HEAD -> master)
Author: KimberlyWillemsPXL <kimberly.willems@pxl.be>
Date:   Sat Oct 17 17:29:11 2020 +0200

    Nieuwe startpagina toevoegen

4576055 (HEAD -> master) Nieuwe startpagina toevoegen
9b61f1c Lege readme.txt toegevoegd
```

Harde reset naar een specifieke commit

We kunnen de unieke code van een commit uit de commit-geschiedenis gebruiken om alle wijzigingen volgend op die commit ongedaan te maken. Dit noemen we een harde 'reset'.

```
git reset --hard 9b61f1c
```

Specifieke bestanden unstagen en unmodifieden

Het kan voorvallen dat we, na een vorige commit, een bestand wijzigen en/of stagen en daarna ontdekken dat we de nieuwe wijzigingen toch niet willen committen, maar helemaal ongedaan willen maken. Dit kunnen we doen via het restore commando:

```
git restore --staged index.html      //Bestand wordt unstaged
git restore index.html               //Bestand wordt unmodified
```

Bestanden uitsluiten

Wanneer we een project aanmaken in Webstorm, worden er ook automatisch (log)bestanden en tijdelijke bestanden aangemaakt die we niet willen meenemen in onze Git repository. Deze bestanden kunnen we uitsluiten door een extra bestand aan te maken in onze werkdirectory, genaamd '.gitignore'.

In dat bestand voegen we (minstens) volgende tekst toe:

```
.gitignore      //Het bestand '.gitignore' zelf
*.[oa]          //Bestanden die eindigen op '.o' of '.a'
*~              //Bestanden die eindigen met een tilde
.idea           //Een map die Webstorm automatisch aanmaakt
```

Werken met GitHub én Git Bash (Lokaal en remote)

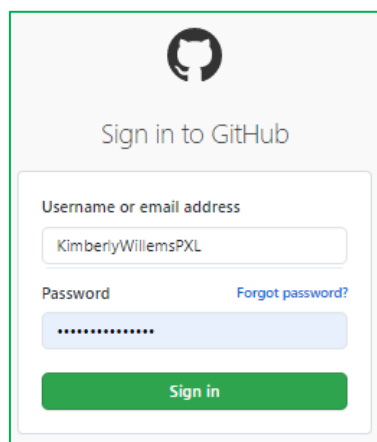
Wanneer we werken met zowel een lokale Git repository als een remote Git repository, moeten we er rekening mee houden dat niets automatisch zal gebeuren.

De wijzigingen die we doen in onze lokale Git repository worden niet meteen doorgevoerd in onze remote Git repository. Omgekeerd wordt onze lokale Git repository niet automatisch

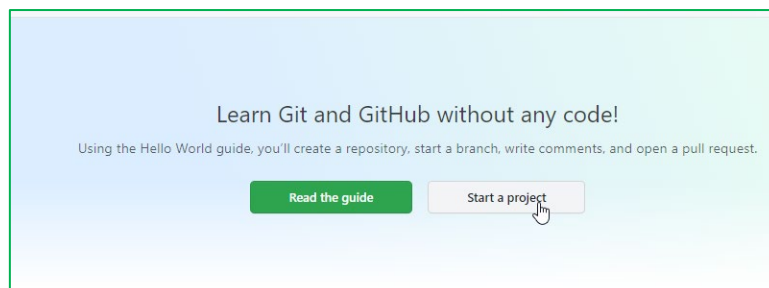
geüpdatet met wijzigingen van onze remote Git repository. Om onze lokale Git repository en remote Git repository in sync te houden, moeten we gebruik maken van specifieke git terminalcommando's.

Een remote Git repository aanmaken

Ga naar de website van GitHub (<https://github.com/>) en log in met je account.

A screenshot of the GitHub login page. At the top is the GitHub logo (Octocat) and the text "Sign in to GitHub". Below this is a form with two input fields: "Username or email address" containing the text "KimberlyWillemsPXL" and "Password" with masked characters. To the right of the password field is a link that says "Forgot password?". At the bottom of the form is a green button labeled "Sign in".

Klik op de knop 'Start a project'.




Vul de gegevens van de nieuwe repository in en kies de gewenste eigenschappen. Klik daarna op de knop 'Create repository'.


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

Repository name *


 KimberlyW30 ▾

/ project-x 


Great repository names are short and memorable. Need inspiration? How about [bookish-octo-telegram?](#)

Description (optional)

Eerste GIT project

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

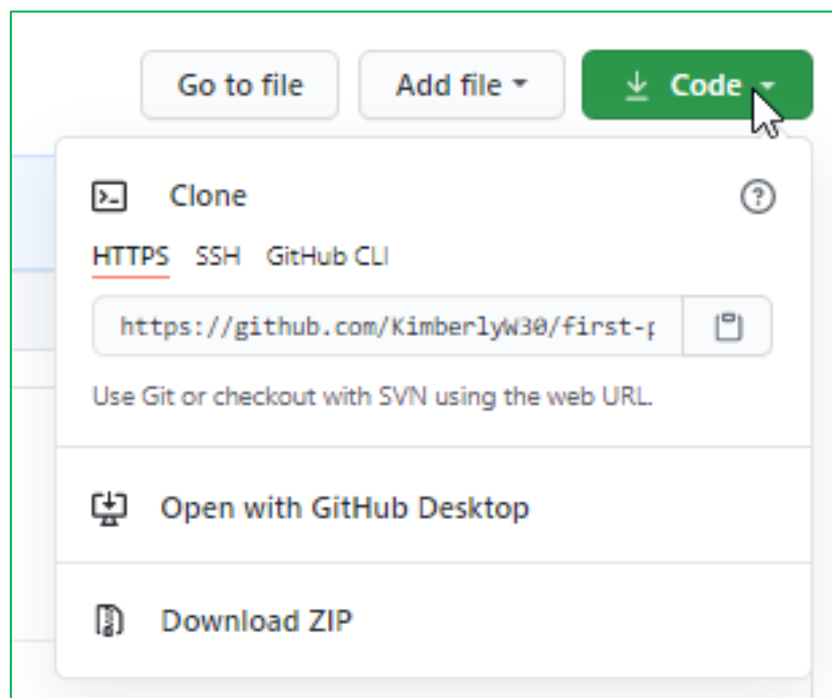
☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

De remote Git repository klonen als lokale Git repository

We gaan de lokale Git repository aanmaken door de remote Git repository te klonen. Zo hebben we, als beginpunt, alle informatie van de remote Git repository meteen ook op onze computer staan.

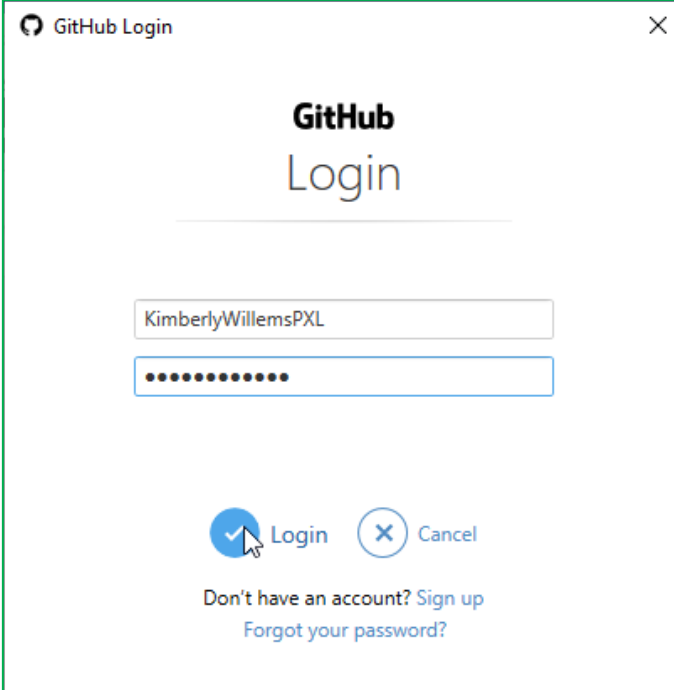
Open de repository via de GitHub website en klik rechtsboven op de knop 'Code'. Kopieer de URL naar het klembord.



Typ in Git Bash volgend commando in:

```
git clone https://github.com/KimberlyWillemsPXL/project-x.git
```

Er verschijnt een GitHub Login scherm. Vul de login-gegevens in.



Terminalcommando's

Pull commando

Om de lokale bestanden te wijzigen, moeten we de lokale branch en de remote branch mergen. Dit doen we via het Pull commando. In het beste geval is de merge eenvoudig en moeten we enkel updates binnenhalen.

```
git pull
git pull origin master
```


Indien er conflicten ontstaan, moeten we deze zelf oplossen.



Push commando

Via het push-commando kunnen we de lokale wijzigingen doorsturen naar de remote Git repository en daar een merge laten plaatsvinden.

```
git push  
git push origin master
```

🔗 master ▾ 🌿 1 branch 🏷️ 0 tags Go to file Add file ▾ 📄 Code ▾

 KimberlyW30 Added readme en index 21e193c 2 minutes ago 🕒 1 commits

 README.txt	Added readme en index	2 minutes ago
 index.html	Added readme en index	2 minutes ago

Add a README with an overview of your project.

Add a README

Indien er conflicten ontstaan tijdens het pushen, moeten we deze zelf manueel oplossen.

Basis HTML

In dit hoofdstuk maken we kennis met de ontwikkeling van webpagina's. We gaan dieper in op de structuur en opbouw van een eenvoudige HTML5-pagina en bekijken enkele belangrijke HTML5-elementen.

HTML5 is (sinds 2014) de nieuwste versie van de HTML-standaard en bevat alle functionaliteiten van zowel HTML als XHTML. In deze markeertaal werden enkele foutjes uit de voorgaande versie weggewerkt en een betere ondersteuning geïmplementeerd voor webapplicaties. Daarnaast introduceert HTML5 ook nieuwe elementen die zorgen voor meer structuur in het document.

Starten met HTML

Een lege webpagina bevat onderstaande basis.

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>Dit is de titel van de webpagina</title>
  </head>
  <body>
  </body>
</html>
```

- De `<!DOCTYPE html>` laat de browser weten welke versie (= HTML5) we gebruiken.
- Het `<html>`-element representeert de root van een HTML-document.
- Het `<head>`-element bevat alle informatie die betrekking heeft op het document, maar niet in het documentvenster wordt weergegeven. Het bevat:
 - alle `<meta>`-elementen met informatie voor de browser en zoekmachines.
 - een `<title>`-element
 - de link naar de stijlbladen en javascriptbestanden.
- Het `<body>`-element bevat alle verdere teksten, afbeeldingen en codes.

Elementen en tags

Aan een lege webpagina kunnen we HTML-elementen toevoegen.

HTML-elementen mét inhoud hebben een begin- en eindtag. Deze tags noemen we 'containertags'. Tussen de begintag en de eindtag staat de inhoud waarop het element betrekking heeft.

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>Dit is de titel van de webpagina</title>
  </head>
  <body>
    <p>Dit is een paragraaf.</p>
  </body>
</html>
```

Sommige elementen zijn 'void elementen' en hebben geen inhoud. Deze elementen hebben hierdoor ook geen eindtag. Deze tags noemen we 'lege tags'.

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>Dit is de titel van de webpagina</title>
  </head>
  <body>
    <p>Dit is een <br /> paragraaf.</p>
    
  </body>
</html>
```

Elementen nesten

Een HTML-element kan andere HTML-elementen bevatten. Dit noemen we 'nesten'. Tijdens het schrijven van onze code, gaan we de elementen laten inspringen om dit weer te geven. Hierdoor krijgen we een duidelijk beeld van het child-element en het parent-element.

Een element kan tegelijkertijd zowel een 'parent' als een 'child' zijn.

Attributen

Elk HTML-element kan in de begintag attributen meekrijgen. Attributen voorzien het element van extra eigenschappen en kenmerken. Er bestaan verschillende soorten attributen:

- *Globale attributen:* attributen die op alle HTML-elementen toepasbaar zijn.
- *Specifieke attributen:* attributen die op specifieke HTML-elementen toepasbaar zijn.

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>Dit is de titel van de webpagina</title>
  </head>
  <body>
    <p class="par" style="color:red">Dit is een paragraaf.</p>
    <a href="http://www.px1.be">Link naar de PXL-website</a>
  </body>
</html>
```

OPGELET! In het volgende hoofdstuk leren we hoe we de inhoud (HTML) van de opmaak (CSS) scheiden, vanaf dan mogen we het attribuut 'style' niet meer gebruiken. Alle attributen moeten dan een inhoudelijke betekenis hebben!

Commentaar

Soms kan het handig zijn om commentaar toe te voegen over onze code.

```
<!--Dit is een stukje commentaar-->
```

Conventies en validatie

Om ervoor te zorgen dat we een ‘goede’, ‘uniforme’, en ‘browseronafhankelijke’ HTML-code schrijven, houden we ons aan volgende conventies:

- We starten een HTML-document altijd met `<!DOCTYPE html>`.
- We plaatsen de `<meta charset="utf-8">` altijd in het `<head>`-element.
- We hanteren een correcte volgorde en nesting van de elementen.
- We schrijven alle HTML-elementen en attributen in kleine letters.
- We zetten de waardes van de attributen altijd tussen dubbele aanhalingstekens.
- We sluiten de elementen met inhoud altijd via een eindtag.
- We plaatsen algemene teksten altijd in paragrafen.

Browsers zijn zeer vergevingsgezind. Hierdoor leveren kleine inbreuken niet onmiddellijk problemen op bij weergave. Toch zorgen we er best voor dat onze webpagina's voldoen aan de standaard, want enkel zo kunnen we een identieke weergave krijgen in alle browsers. Deze controle ten opzichte van de standaard noemen we ‘validatie’.

Bij een validatie wordt een HTML5-document gecontroleerd op:

- Alle vereisten vastgelegd in de standaard
- Alle veelvoorkomende fouten
- Het bestaan van de gebruikte elementen en attributen

Om te valideren kunnen we gebruik maken van de ingebouwde validators in onze editor of van webbased validators. De bekendste webbased validator is <https://validator.w3.org/>. Via deze validator kunnen we zowel code als online en offline documenten controleren.

OPGELET! Een oefening is pas klaar als de code gevalideerd is zonder errors!

Structuur

Blokelementen en inline elementen

Er bestaan twee types van HTML-elementen:

- *Blokelementen*: deze elementen nemen altijd een nieuwe regel in beslag en maken gebruik van de volledige breedte van het scherm.

Dit is een blok element.

- *Inline elementen*: deze elementen worden in de huidige regel getoond en zijn even breed als de inhoud van het element.

Dit is een `inline element`.

Voorbeelden blokelementen

Paragraaf	<code><p>...</p></code>	Divisie	<code><div>...</div></code>
Kop 1	<code><h1>...</h1></code>	Blockquote/citaat	<code><blockquote>...</blockquote></code>
Kop 2	<code><h2>...</h2></code>	Preformatted	<code><pre>...</pre></code>
Kop 3	<code><h3>...</h3></code>	Ongeordende lijsten	<code></code>
Kop 4	<code><h4>...</h4></code>	Geordende lijsten	<code></code>
Kop 5	<code><h5>...</h5></code>		
Kop 6	<code><h6>...</h6></code>		

Voorbeelden inline elementen

Code blok	<code><code>...</code></code>	Afbeelding	<code></code>
Quote	<code><q>...</q></code>	Inputveld	<code><input /></code>
Hyperlink	<code><a>...</code>		<code><select></code>
Tekst-selectie	<code>...</code>	Line break	<code>
</code>
Urgent	<code>...</code>	Word break	<code><wbr /></code>
Superscript	<code><sup>...</sup></code>		
Subscript	<code><sub>...</sub></code>		
Uitgelicht	<code>...</code>		
Andere stemming	<code><i>...</i></code>		

Structuurelementen

In de vroegere HTML-versies werd er voor alles van structurele waarde het `<div>`-element gebruikt. Sinds HTML5 bestaan er specifieke structuurelementen.

- Hoofding `<header>...</header>`
- Navigatieblok `<nav>...</nav>`
- Hoofdgedeelte `<main>...</main>`
- Sectie (deel van een geheel) `<section>...</section>`
- Artikel (losstaand geheel) `<article>...</article>`
- Voetnoot of voettekst `<footer>...</footer>`
- Randinformatie `<aside>...</aside>`

Het is niet altijd eenvoudig om te bepalen wanneer we moeten kiezen voor welk structuurelement om een bepaalde inhoudsblok te definiëren. Vooral de keuze tussen sectie, artikel en randinformatie is voor interpretatie vatbaar. Daarom gebruiken we best de leidraad op de volgende pagina voor het kiezen van het juiste structuurelement.

Artikel	<p>De inhoud van dit element kan op zichzelf gelezen of gedeeld worden. De inhoud is onafhankelijk van de rest van de pagina.</p> <ul style="list-style-type: none"> • Bvb. Een recept in een kookboek.
Sectie	<p>De inhoud van dit element is een onderdeel van een geheel en is op zichzelf onvolledig.</p> <ul style="list-style-type: none"> • Bvb. Een hoofdstuk uit een boek.
Randinformatie	<p>De inhoud van dit element kan worden weggelaten. Het wordt gezien als iets 'extra', maar het is niet nodig om de inhoud te lezen of te begrijpen.</p> <ul style="list-style-type: none"> • Bvb. Een reclamebanner.
Hoofding	<p>De inhoud van dit element heeft een inleidend of verwijzend karakter.</p> <ul style="list-style-type: none"> • Bvb. De titel en subtitel van een blogpost.
Hoofdgedeelte	<p>De inhoud van dit element is dominant en essentieel.</p> <ul style="list-style-type: none"> • Bvb. De eigenlijke tekst van een blogpost.
Voettekst	<p>De inhoud van dit structuurelement omvat meta- of extra detailinformatie.</p> <ul style="list-style-type: none"> • Bvb. De copyrightinformatie onderaan een webpagina.
Navigatieblok	<p>De inhoud van dit element omvat navigatielinken.</p> <ul style="list-style-type: none"> • Bvb. De linken naar andere webpagina's binnen een website.

```

<body>
  <header>
    <h1>Paginatitel</h1>
    <p>Tekst</p>
  </header>
  <nav>
    <ul>
      <li><a href="link1.html">Link 1</a></li>
      <li><a href="link2.html">Link 2</a></li>
      <li><a href="link3.html">Link 3</a></li>
    </ul>
  </nav>
  <main>
    <section>

```

```

        <h2>Titel</h2>
        <p>Inhoud van de sectie</p>
    </section>
    <section>
        <h2>Titel</h2>
        <p>Inhoud van de sectie</p>
    </section>
</main>
<footer>
    <p>Paginavoetnoot</p>
</footer>
</body>

```

Hyperlinks

Via hyperlinks kunnen we navigeren naar andere webpagina's of bestanden. Om een hyperlink te bekomen, gebruiken we het `<a>`-element. Het meest gebruikte attribuut van het `<a>`-element is `href`, oftewel de 'hypertext reference'.

```

<a href="pagina1.html">Pagina 1</a>
<a href="http://www.andere-website.be/pagina2.html">Pagina 2</a>
<a href="/txt/tekst3.txt">Open tekstbestand in map txt</a>

```

Een ander veelvoorkomend attribuut van het `<a>`-element is `target`. Dit attribuut geeft aan waar de link geopend wordt.

<code>_self</code>	In het huidige venster (default).
<code>_blank</code>	In een nieuw venster of tabblad van de browser.

```
<a href="http://www.px1.be" target="_blank">Website van de PXL</a>
```

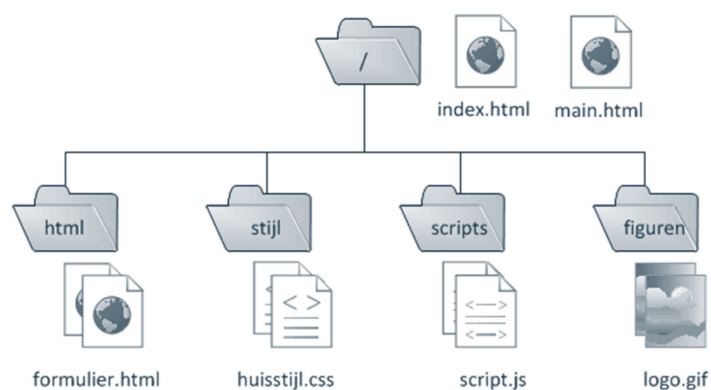
Absolute koppelingen

Een absolute koppeling verwijst naar een document op een webserver en gebruikt hiervoor een URL-adres.

```
<a href="http://www.andere-website.be/pagina2.html">Pagina 2</a>
```

Relatieve koppelingen

Een (document)relatieve koppeling verwijst naar een ander bestand, maar doet dit relatief ten opzichte van het bestand van waaruit de link wordt aangeklikt.



We maken een relatieve koppeling van 'index.html' naar 'main.html'. Het bestand waarnaar we linken staat in dezelfde map.

```
<a href="main.html">...</a>
```

We maken een relatieve koppeling van 'index.html' naar 'formulier.html'. Het bestand waarnaar we linken staat één map dieper in de folderhiërarchie (dalen).

```
<a href="html/formulier.html">...</a>
```

We maken een relatieve koppeling van 'formulier.html' naar 'index.html'. Het bestand waarnaar we linken staat één map hoger in de folderhiërarchie (stijgen).

```
<a href="../index.html">...</a>
```

Interne koppelingen

Een interne koppeling wordt gebruikt om naar een specifieke positie te gaan binnen eenzelfde of een ander document. Een interne koppeling bestaat altijd uit twee delen:

- Een *anker* die de positie binnen het document aangeeft.
- Een *hyperlink* die verwijst naar de positie binnen het document.

We maken een interne koppeling om te verwijzen naar een specifiek artikel op onze webpagina. Bij het klikken op de link, springt de webbrowser naar het artikel.

```
<h2>Inhoudsopgave</h2>
<ul>
  <li><a href="#deel1">deel 1</a></li>
  <li><a href="#deel2">deel 2</a></li>
</ul>
<article id="deel1">
  <h3>Deel 1: Inleiding</h3>
</article>
<article id="deel2">
  <h3>Deel 2: Structuur</h3>
</article>
```

We maken een interne koppeling om te verwijzen naar een specifiek artikel op een andere webpagina. Deze webpagina staat in dezelfde map als onze huidige webpagina. Het anker is reeds aanwezig op de andere webpagina.

```
<a href="index.html#deel2">deel 2</a>
```

We maken een interne koppeling om een top-link (knop op terug naar boven te gaan) toe te voegen aan een lange webpagina waarbij er een verticale scrollbar aanwezig is. Het plaatsen van een anker is niet nodig.

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

Koppeling naar een e-mailadres

Bij een koppeling naar een e-mailadres voegen we een mailto-opdracht toe aan de link. De mailto-opdracht kunnen we naar wens uitbreiden met een 'cc', 'bcc' en 'subject'.

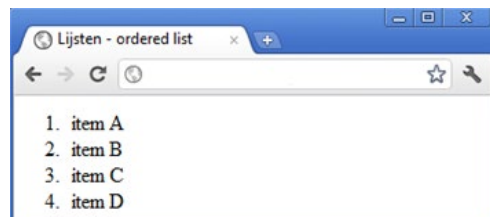
```
<a href="mailto:persoon1@bedrijf.be">...</a>  
  
<a href="mailto:persoon1@bedrijf.be?cc=persoon2@bedrijf.be&  
bcc=persoon3@bedrijf.be&subject=Nieuw%20bericht!">...</a>
```

Merk op dat de ampersand '&' geëncodeerd is als '&,' daar er geen speciale tekens kunnen gebruikt worden in HTML. Ook de spatie is geëncodeerd als '%20', omdat er geen spaties mogen voorkomen in een URL.

Lijsten

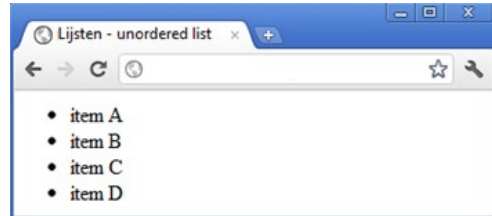
Geordende lijsten zijn lijsten met automatische nummering.

```
<ol>  
  <li>item A</li>  
  <li>item B</li>  
  <li>item C</li>  
  <li>item D</li>  
</ol>
```



Ongeordende lijsten zijn lijsten met opsommingstekens waarbij de volgorde van de items niet van belang is.

```
<ul>
  <li>item A</li>
  <li>item B</li>
  <li>item C</li>
  <li>item D</li>
</ul>
```



Definitielijsten zijn woordenlijsten met telkens een term gevolgd door de verklaring.

```
<dl>
  <dt>Term</dt>
  <dd>Definitie van woord</dd>
</dl>
```



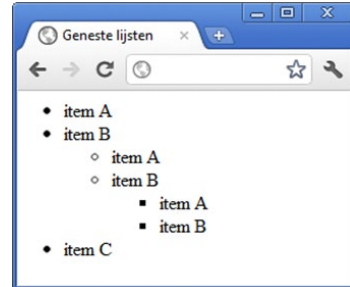
Geneste lijsten

Bij het nesten van een lijst binnen een andere lijst springen de verschillende niveaus automatisch in met telkens een ander opsomming (of nummering).


```

<ul>
  <li>item A</li>
  <li>item B
    <ul>
      <li>item A</li>
      <li>item B
        <ul>
          <li>item A</li>
          <li>item B</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>item C</li>
</ul>

```



Afbeeldingen

We kunnen een afbeelding op onze webpagina plaatsen via de open tag ``. Hierbij zijn twee attributen altijd verplicht:

- *Het src-attribuut* geeft het pad naar de afbeelding die getoond moet worden.
- *Het alt-attribuut* geeft een alternatieve tekst die getoond moet worden als de afbeelding niet kan weergegeven worden.

```

```

Er kunnen verschillende afbeeldingsformaten op een webpagina geplaatst worden. Het gebruik van formaat is afhankelijk van de wensen en noden.

	GIF	JPEG	PNG
Aantal kleuren	256	16 miljoen	16 miljoen
Bestandsgrootte	Groter	Kleiner	Kleiner
Downloadtijd	Langer	Korter	Korter
Decompressietijd	Korter	Langer	Langer
Animatie	Aanbevolen	Afgeraden	Afgeraden
Transparantie	Gif89a	Niet mogelijk	Mogelijk
Logo's	Aanbevolen	Afgeraden	Afgeraden
Computertekening	Aanbevolen	Afgeraden	Afgeraden
Foto's	Afgeraden	Aanbevolen	Aanbevolen

Het figure-element

Het `<figure>`-element is een nieuw structuurelement in HTML5 en is bedoeld om foto's, illustraties, diagrammen en/of codelistings te groeperen.

```
<figure>
  
  
</figure>
```

Afbeelding als koppeling

Wanneer we een ``-element plaatsen binnen een `<a>`-element, werkt de afbeelding als een koppeling.

```
<a href="http://www.px1.be"></a>
```

Youtube-video's

Met een `<iframe>`-element kunnen we eenvoudig een Youtube-video invoegen op onze webpagina.

```
<iframe src="https://www.youtube.com/embed/VIDEO_ID" width="680"  
height="385"></iframe>
```

<https://www.youtube.com/watch?v=uyA01nH72NI>

Basis CSS

In dit hoofdstuk gaan we aan de slag met Cascading Style Sheets (CSS). Deze taal gebruiken we om opmaak te geven aan onze HTML elementen. CSS zorgt voor een scheiding van de inhoud (HTML) en de stijl (CSS).

We leren hoe we CSS kunnen implementeren en hoe we één of meerdere elementen aanspreken aan de hand van verschillende selectoren. Daarnaast zien we enkele opmaakregels binnen CSS.

Syntax van CSS

Een CSS-regel bestaat altijd uit twee delen:

- De selector
- De stijldeclaraties (de stijleigenschappen met hun waarden).

```
selector {  
    stijldeclaratie;  
}
```

```
p {  
    font-size: 15pt;  
    line-height: 20pt;  
    color: red;  
}  
h1, h2 {  
    font-family: "Times New Roman", serif;  
}
```

Commentaar

Soms kan het handig zijn om commentaar toe te voegen.

```
/* opmaak voor alle paragrafen */
```

Toepassen van CSS

Lokale of inline stijl

Een stijldeclaratie kan binnen de begintag van een HTML-element worden geplaatst. Dit noemen we een *lokale of inline stijl*. De opgenomen stijldeclaratie geldt dan enkel voor dat specifieke HTML-element. **Dit gaan wij niet doen in deze cursus!**

```
<p style="color: red;">De tekst van deze paragraaf is rood.</p>
```

Globale stijl of blokstijl

Een *globale stijl of blokstijl* wordt opgenomen in het `<head>`-element van een webpagina en voorziet de opmaak van de HTML-elementen van deze webpagina. Telkens we de HTML-elementen op de webpagina gebruiken, wordt de opmaak overgenomen. **Ook dit gaan wij niet doen in deze cursus!**

```
<head>
  <style>
    body {
      background-color: #999999;
    }
    p {
      font-family: Verdana, Arial, Geneva, sans-serif;
    }
  </style>
</head>
```

Gelinkte stijl met apart stijlblad

We kunnen voor één of meerdere webpagina's ook een apart stijlbestand (.css) gebruiken.

De link naar het extern stijlblad staat dan in het `<head>`-element van de webpagina's.

Het `<link>`-element om het apart stijlblad te koppelen bevat altijd minstens twee attributen.

Het *rel-attribuut* bepaalt de relatie met het document en bevat de waarde "stylesheet". Het

href-attribuut bepaalt welk apart stijlblad dient gekoppeld te worden.

```
<head>
  <link rel="stylesheet" href="style.css" />
</head>
```

Een alternatief voor het `<link>`-element is het gebruik van `@import` om een apart stijlblad te koppelen.

```
<head>
  <style>
    @import url(style.css);
  </style>
</head>
```

Volgorde van opmaak

In de term 'Cascading Style Sheets' verwijst de term 'Cascading' naar het watervalsysteem dat geldt bij het toekennen van de stijlen. Volgende waterval-volgorde is van toepassing op de stijl van een webpagina:

- De gelinkte stijl
- De globale stijl
- De lokale stijl

Selectors

HTML tags

Alle HTML-tags kunnen gebruikt worden als CSS-selector. De gegeven opmaak is dan van toepassing op elk HTML-element met die HTML-tag.

```
body {  
    background-color: #cccccc;  
}  
h1 {  
    color: navy;  
}
```

Classes

Om herhaling te vermijden, kunnen we een reeks identieke stijleigenschappen onderbrengen in een klasse. Nadien kunnen we deze klasse aan willekeurige en meerdere HTML-elementen koppelen via het class-attribuut.

Een klasse-selector start in CSS met een . (punt).

```
.red {  
    color: red;  
}  
.green {  
    color: green;  
}
```

```
-----  
<head>  
    <link rel="stylesheet" href="style.css" />  
</head>  
<body>  
    <p class="red">Deze paragraaf heeft een rode tekstkleur.</p>
```

Deze paragraaf heeft een rode tekstkleur.
Deze paragraaf heeft een groene tekstkleur.
Deze paragraaf heeft ook een rode tekstkleur.

```
<p class="green">Deze paragraaf heeft een groene tekstkleur.</p>
<p class="red">Deze paragraaf heeft ook een rode tekstkleur.</p>
</body>
```

Klassen kunnen ook gekoppeld worden aan een bepaald HTML-element. Op die manier kunnen we stijlen definiëren die enkel voor dat element gelden. Deze klassen noemen we *afhankelijke klassen (fixed classes)*.

```
p {
    font-size: 12pt;
    font-family: Verdana, sans-serif;
}
.red {
    color: red;
    font-style: italic;
}
p.red {
    font-size: 14pt;
    font-weight: bold;
}
```

Hoofding met class red

Paragraaf met fixed class p.red

Paragraaf zonder class red

```
-----
<head>
    <link rel="stylesheet" href="style.css" />
</head>
<body>
    <h1 class="red">Hoofding met class red</h1>
    <p class="red">Paragraaf met fixed class p.red</p>
    <p>Paragraaf zonder class red</p>
</body>
```


Id's

Indien een stijldeclaratie slechts voor één element bedoeld is, is het niet logisch hiervoor een klasse te creëren. In dit geval gebruiken we een ID om een unieke stijl toe te kennen aan een element. Een ID kunnen we aan één uniek HTML-element koppelen via het id-attribuut.

Een id-selector start in CSS met een # (hashtag/hekje).

```
#red {  
    color: red;  
}
```

Enkel deze paragraaf heeft een rode tekstkleur.

Deze paragraaf heeft geen rode tekstkleur.

Deze paragraaf heeft ook geen rode tekstkleur.

```
-----  
<head>  
    <link rel="stylesheet" href="style.css" />  
</head>  
<body>  
    <p id="red">Enkel deze paragraaf heeft een rode tekstkleur.</p>  
    <p>Deze paragraaf heeft geen rode tekstkleur.</p>  
    <p>Deze paragraaf heeft ook geen rode tekstkleur.</p>  
</body>
```

Speciale selectors

Asterisk

De asterisk of universele selector werkt als 'wildcard' en is van toepassing op alle elementen.

```
* {  
    margin-top: 10px;  
}
```

Gecombineerde klassen

We kunnen meerdere klassen toepassen op een HTML-element door de klassen achter elkaar te schrijven.

```
.geel {  
    color: yellow;  
}  
.cursief {  
    font-style: italic;  
}  
-----  
<p class="geel cursief">Deze tekst is geel en cursief.</p>
```

Contextuele of descendant selectors

Alle HTML-elementen worden genest waardoor er een parent-child relatie ontstaat tussen de elementen onderling. Deze parent-child relatie kunnen we gebruiken om bepaalde elementen een opmaak mee te geven die afhankelijk is van het element waarin ze genest zijn.

```
section header h1 {  
    color: navy;  
    font-weight: bold;  
}
```

Attribuutselectors

We kunnen ook een opmaak vastleggen op basis van het attribuut of de attribuutwaarde van een element.

```
img[alt] {  
    margin: 10px;      /* <img>-element, alt-attribuut */  
}  
img[alt="figuur"] {  
    margin: 10px;      /* <img>-element, alt is "figuur" */  
}  
img[alt^="figuur"] {  
    margin: 10px;      /* <img>-element, alt beginnend met "figuur" */  
}  
img[alt$="figuur"] {  
    margin: 10px;      /* <img>-element, alt eindigend met "figuur" */  
}  
img[alt*="figuur"] {  
    margin: 10px;      /* <img>-element, "figuur" in alt */  
}
```

Pseudo-classes

Met een pseudo-klasse kunnen we de opmaak voor een specifieke toestand van een element vastleggen.

```
a:visited {  
    color: #00FF00;  
    text-decoration: none;  
}  
a:hover {  
    color: #FF00FF;  
    text-decoration: underline;  
}
```

Binnen CSS kunnen we verschillende psuedo-klassen gebruiken. Onderstaande tabel geeft slechts enkele mogelijkheden.

:link	Een niet-bezochte koppeling.
:visited	Een bezochte koppeling.
:hover	Een koppeling waar de muiscursor op staat.
:active	Een actief element (meestal een koppeling waarop geklikt is).
:focus	Een element met de focus.
:first-child	Het eerste kind-element van het parent-element.
:last-child	Het laatste kind-element van het parent-element.
:nth-child()	Het x kind-element van het parent-element.
:checked	Een invoerelement (radiobutton, checkbox) dat checked is.
:first-of-type	Het eerste element van dat type.
:last-of-type	Het laatste element van dat type.

Eenheden

We kunnen de afmetingen op onze website in verschillende eenheden uitdrukken.

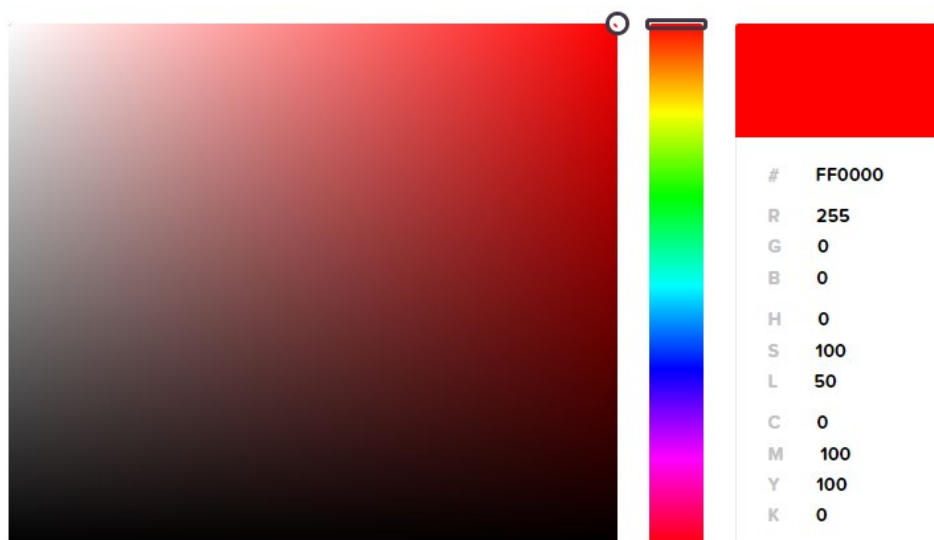
Onderstaande eenheden worden het meeste gebruikt.

em	De breedte van de letter "M" van het gebruikte lettertype.
px	In pixels, volgens de schermresolutie.
pt	In punten, 1pt is gelijk aan 1/72 inch.
%	Het percentage ten opzichte van het parent-element.

Opmaak

Kleuren

Als er niets wordt opgegeven is de standaardkleur die door de browsers wordt gebruikt wit voor de achtergrond en zwart voor de voorgrond. Om een goed ogende webpagina te maken zijn er natuurlijk meerdere kleuren nodig.



Kleurnamen

Origineel stonden er 17 naamkleuren in de specificaties van HTML en CSS.

Aqua	Black	Blue	Fuchsia	Gray
Green	Lime	Orange	Maroon	Navy
Olive	Purple	Red	Silver	Teal
White	Yellow			

Sinds CSS3 kunnen we 147 kleurnamen gebruiken in HTML en CSS.

```
body {  
    background-color: fuchsia;  
    color: darkslategrey;  
}
```

RGB-model

Naast de vastgelegde kleurnamen, kunnen we ook gebruikmaken van het RGB-model.

Binnen dit model kunnen we een kleur bepalen door intensiteiten van rood, groen en blauw licht te mengen.

We kunnen de RGB-kleuren op twee manieren noteren:

- Hexadecimaal als #rrggbb met waardes uitgedrukt in cijfers 0 tot en met 9 en letters A tot en met F.

```
body {  
    background-color: #ff77ff;  
    color: #2f4f4f;  
}
```

- Decimaal als rgb(r, g, b) met waardes tussen 0 en 255 of rgb(r%, g%, b%) met waardes uitgedrukt in percentage.

```
body {  
    background-color: rgb(255,119,255);  
    color: rgb(47, 79, 79);  
}  
  
body {  
    background-color: rgb(100%,46.7%,100%);  
    color: rgb(18.4%,31%,31%);  
}
```

In CSS3 kunnen we bij de decimale notatie nog een vierde waarde meegeven, namelijk de mate van transparantie (a = alpha).

- Waarde 0: volledig doorzichtig.
- Waarde 1: ondoorzichtig.

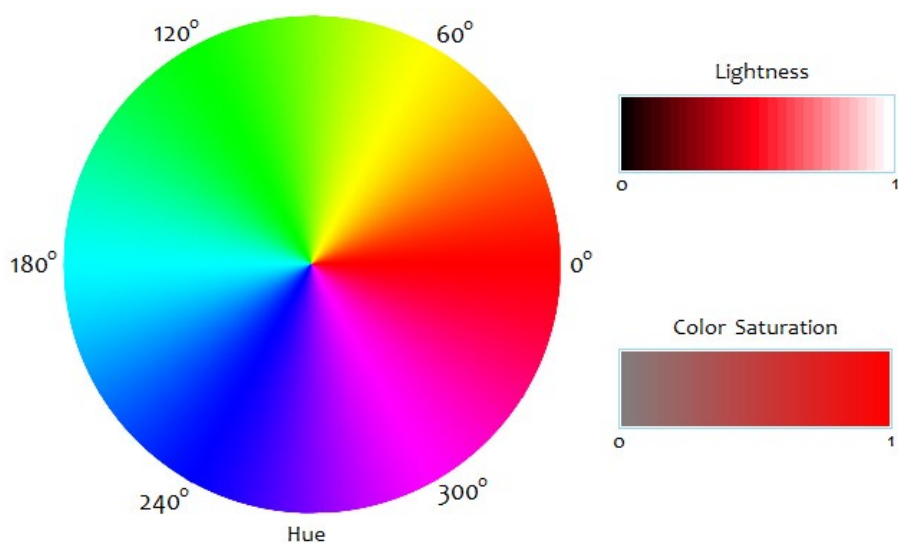
```
body {  
    background-color: rgba(255,119,255, 0.2);  
    color: rgba(47, 79, 79, 1);  
}
```

```
body {  
    background-color: rgba(100%,46.7%,100%, 0.2);  
    color: rgba(18.4%,31%,31%, 1);  
}
```

HSL-model

Via het HSL-model kunnen we een kleur bepalen door een waarde toe te kennen aan de tint (in graden), de verzadiging (in percentage) en de helderheid (in percentage).

```
body {  
    background-color: hsl(300, 100%, 50%);  
    color: hsl(180, 25%, 25%);  
}  
  
body {  
    background-color: hsla(300, 100%, 50%, 0.2);  
    color: hsla(180, 25%, 25%, 1);  
}
```



Tekstopmaak

Het lettertype

Met volgende CSS font-eigenschappen en bijhorende waardes kunnen we opmaak geven aan teksten.

CSS	Waarde
font-family	Naam van het lettertype
font-size	xx-small, x-small, small, medium, large, x-large, xx-large, larger, smaller in, cm, mm, pt, px, em, ex, %
font-style	normal, italic, oblique
font-variant	normal, small-caps
font-weight	normal, bold, bolder, lighter 100, 200, 300, 400, 500, 600, 700, 800, 900

Voor de CSS-eigenschap 'font-family' moeten we altijd de juiste benaming van het lettertype meegeven, maar helaas kan de benaming verschillen per besturingssysteem (Windows, Linux, Os). Om problemen te voorkomen geven we dus best altijd meerdere alternatieven én een fontfamilie op.

De font-families zijn:

- Sans-serif: schreefloze lettertypes.
- Serif: schreef lettertypes.
- Monospace: niet-proportionele lettertypes.
- Cursive: lettertypes die op een handschrift lijken.
- Fantasy: fantasievolle lettertypes.

Benamingen van lettertypes die bestaan uit meerdere woorden, moeten altijd tussen quotes geschreven worden.

```
p {  
    font-family: Times, "Times New Roman", serif;  
}
```

Het is mogelijk om meerdere font-eigenschappen te combineren en zo een verkorte schrijfwijze te gebruiken binnen CSS. Hierbij is de volgorde van de waardes belangrijk. Zo moet 'font-weight' en 'font-style' altijd als eerste staan.

```
p {  
    font-weight: bold;  
    font-style: italic;  
    font-size: 12pt;  
    line-height: 20pt;  
    font-family: "Times New Roman", serif;  
}  
p {  
    font: bold italic 12pt/20pt "Times New Roman", serif;  
}
```

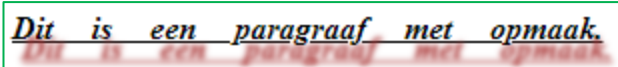
Dit is een paragraaf met opmaak.

Opmaak van de tekst

Naast het lettertype, de grootte en de kleur kunnen we met CSS ook het uitzicht van de tekst volledig aanpassen en de verticale en horizontale uitlijning bepalen.

CSS	Vertaling	Waarde
direction	Tekstrichting	ltr, rtl
line-height	Lijnhoogte	in, cm, mm, pt, px, em, ex, %
text-align	Tekstuitlijning	left, right, center, justify
text-decoration	Tekstdecoratie	none, underline, overline, line-through, blink
text-shadow	Tekstschaduw	kleur, x, y, vervaging
vertical-align	Verticale uitlijning	baseline, sub, super, top, text-top, middle, bottom, text-bottom, %
Word-spacing	Spaties	in, cm, mm, pt, px, em, ex, %

```
p {  
    font-weight: bold;  
    font-style: italic;  
    font-size: 12pt;  
    line-height: 20pt;  
    font-family: "Times New Roman", serif;  
    text-decoration: underline;  
    text-shadow: brown 5px 10px 3px;  
    word-spacing: 15px;  
}
```



Het box-model

Elk blokelement kan opgemaakt worden door stijldeclaraties die invloed hebben op de alineaering, de witruimte, de marges en de aan- en afwezigheid van een kader. Elk element kan hierdoor als een 'box' gezien worden in functie van het design.



Hierbij zit de inhoud (tekst, afbeelding, ...) van het element centraal. Om witruimte te bekomen binnen het element, dienen we padding toe te voegen. Daarna kan er een kader (border) rond het element worden geplaatst. Als laatste kunnen we witruimte buiten het element voorzien door margin toe te voegen.



Padding

De padding is de witruimte *binnen* het element. De padding kan apart opgegeven worden per zijde (boven, onder, links en rechts) of voor alle zijdes samen.

```
p {
    padding-top: 25px;
    padding-right: 50px;
    padding-bottom: 75px;
    padding-left: 60px;
}
h1 {
    /* volgorde kloksgewijs, top right bottom left */
    padding: 25px 50px 25px 50px;
}
img {
    /* alle zijdes 25px padding */
    padding: 25px;
}
```

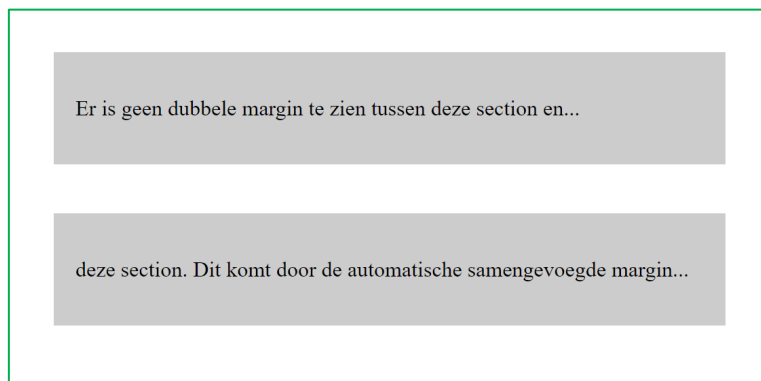
Margin

De margin is de witruimte (marge) *buiten* het element. De margin kan apart opgegeven worden per zijde (boven, recht, onder en links) of voor alle zijdes samen.

```
p {
    margin-top: 25px;
    margin-right: 50px;
    margin-bottom: 75px;
    margin-left: 60px;
}
h1 {
    /* volgorde kloksgewijs, top right bottom left */
    margin: 25px 50px 25px 50px;
}
```

```
img {  
    /* alle zijdes 50px margin */  
    margin: 50px;  
}
```

OPGELET! Om te voorkomen dat tussen meerdere structuurelementen onder elkaar dubbele margins ontstaan, worden de margins, in de standaard flow layout zonder specifiek ingestelde positionering, aan de boven- en onderkant **ALTIJD** automatisch samengevoegd (collapsing margins). De margins links en rechts worden **NOOIT** automatisch samengevoegd.



```
<section>  
    <p>Er is geen dubbele margin te zien tussen deze section en...</p>  
</section>  
<section>  
    <p>deze section. Dit komt door de automatische samengevoegde  
        margin...</p>  
</section>  
-----  
section {  
    margin: 36px;  
    padding: 16px;  
    background-color: #ccc;  
}
```

Border

De border zorgt voor een kader rond het element. Aan de border kunnen we een breedte, kleur, en stijl meegeven. De CSS-eigenschappen van de border kunnen apart of samen opgegeven worden.

```
p {
    border-top-width: 5px;
    border-right-width: 15px;
    border-bottom-width: 5px;
    border-left-width: 10px;
    border-color: green;
    border-style: dashed;
}
h1 {
    /* volgorde kloksgewijs, top right bottom left */
    border-width: 5px 10px 10px 15px;
    border-color: green;
    border-style: dotted;
}
img {
    /* alle zijdes 5px border, kleur groen en volle lijn */
    border: 5px green solid;
}
```

Overzicht

Met volgende CSS-eigenschappen en bijhorende waardes kunnen we de padding, de margin en de border van een blokelement instellen.

CSS	Waarde
margin	in, cm, mm, pt, px, em, ex, %
padding	in, cm, mm, pt, px, em, ex, %

border	breedte kleur stijl
border-width	thin, medium, thick in, cm, mm, pt, px, em, ex, %
border-color	kleur
border-style	dotted, dashed, solid, double, groove, ridge, inset, outset
border-radius	in, cm, mm, pt, px, em, ex, %
border-image	url()
box-shadow	kleur x y vervaging verspreiding
width	in, cm, mm, pt, px, em, ex, %
height	in, cm, mm, pt, px, em, ex, %

Dit stukje tekst staat in een grijze box met een groene rand en witruimte binnen en buiten het element.

```
<p>Dit stukje tekst staat in een grijze box met een groene rand en witruimte binnen en buiten het element.</p>
```

```
-----
```

```
#voorbeeldbox {
  width: 250px;
  padding: 20px;
  border: 5px solid green;
  margin: 35px;
  background-color: silver;
}
```


Lijsten

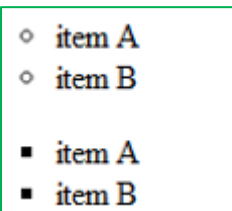
Met volgende CSS-eigenschappen en waardes kunnen we lijsten een opmaak geven.

CSS	Waarde
list-style-image	url()
list-style-position	inside, outside
list-style-type	disc, circle, square; decimal, lower-roman, upper-roman, lower-alpha, none

```
<ul id="cirkel">
  <li>item A</li>
  <li>item B</li>
</ul>
```

```
<ul id="rondje">
  <li>item A</li>
  <li>item B</li>
</ul>
```

```
#cirkel {
  list-style: circle;
}
#blokje {
  list-style: square;
}
```

- 
- item A
 - item B
 - item A
 - item B

Afbeeldingen

Met volgende CSS-eigenschappen en bijhorende waardes kunnen we afbeeldingen een opmaak geven.

CSS	Waarde
height	in, cm, mm, pt, px, em, ex, %
width	in, cm, mm, pt, px, em, ex, %
vertical-align	top, text-top, middle, bottom, text-bottom, baseline, sub, super
border	border-style border-width border-color
padding	in, cm, mm, pt, px, em, ex, %
margin	in, cm, mm, pt, px, em, ex, %

```

```

```
-----  
img {  
    width: 70px;  
}
```



Afbeelding of kleur als achtergrond

Met volgende CSS-eigenschappen en bijhorende waardes kunnen we een afbeelding als achtergrond instellen.

CSS	Waarde
background-attachment	scroll, fixed
background-color	kleur
background-image	url()
background-position	left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom
background-repeat	repeat, repeat-x, repeat-y, no-repeat
background-size	length, percentage, cover, contain, auto

```
body {  
    background-image: url(img/school.jpg);  
    background-attachment: fixed;  
}
```



Tabellen

Structuur van een tabel

Met een `<table>`-element kunnen we een tabel toevoegen aan onze webpagina. Een tabel wordt rij per rij opgebouwd en iedere rij kan een bepaalde hoeveelheid cellen met inhoud bevatten.

De tabelrij

Met een `<tr>`-element (table row) kunnen we een rij toevoegen aan een tabel.

```
<table>
  <tr>
</tr>
</table>
```

De cellen

Enkel binnen de cellen van een tabel mogen we inhoud plaatsen. Met een `<td>`-element (table data) kunnen we cellen aan onze rij toevoegen. Deze cellen verdelen de rij in het gewenste aantal kolommen.

```
<table>
  <tr>
    <td>...</td>
    <td>...</td>
  </tr>
</table>
```

Het <th>-element (table heading) is een variant van het <td>-element en zorgt ervoor dat de inhoud gecentreerd en vet wordt weergegeven. Dit element kunnen we gebruiken om de eerste rij of eerste kolom van een tabel te definiëren als veldnaam.

```
<table>
  <tr>
    <th>...</th>
    <th>...</th>
  </tr>
  <tr>
    <td>...</td>
    <td>...</td>
  </tr>
</table>
```

Het bijschrift

Met een <caption>-element kunnen we een tabel een titel of bijschrift geven.

```
<table border="1">
  <caption>Afbeeldingsformaten</caption>
  <tr>
    <th></th>
    <th>GIF</th>
    <th>JPEG</th>
  </tr>
  <tr>
    <th>Aantal kleuren</th>
    <td>256</td>
    <td>16 000 000</td>
  </tr>
  <tr>
    <th>Bestandsgrootte</th>
    <td>Groter</td>
    <td>Kleiner</td>
  </tr>
</table>
```

Afbeeldingsformaten		
	GIF	JPEG
Aantal kleuren	256	16 000 000
Bestandsgrootte	Groter	Kleiner

```

        <th>Bestandsgrootte</th>
        <td>Groter</td>
        <td>Kleiner</td>
    </tr>
</table>

```

OPGELET! Via het attribuut 'border' kunnen we onze tabel een 'tijdelijke' rand geven. Dit attribuut mag enkel blijven staan tijdens het ontwerpen van onze webpagina om de tabel beter te visualiseren. Net voor het finaliseren, moeten we het attribuut terug verwijderen.

CSS

Met volgende specifieke CSS-eigenschappen en bijhorende waardes kunnen we tabellen en cellen een opmaak geven.

CSS	Waarde
border-collapse	collapse (geen ruimte tussen cellen) separate (ruimte tussen cellen)
border-color	kleur
border-spacing	in, cm, mm, pt, px, em, ex, %
border-width	in, cm, mm, pt, px, em, ex, %
caption-side	top, bottom, initial, inherit
empty-cells	show (cellen zonder inhoud weergeven) hide (cellen zonder inhoud niet weergeven)
height	in, cm, mm, pt, px, em, ex, %

padding	in, cm, mm, pt, px, em, ex, %
text-align	left, right, center, justify
vertical-align	top, text-top, middle, bottom, text-bottom, baseline, sub, super
width	in, cm, mm, pt, px, em, ex, %

```

<table>
  <caption>Opleidingsonderdelen Graduaatsopleiding DVO</caption>
  <tr>
    <th>Opleidingsonderdeel</th>
    <th>Lector</th>
  </tr>
  <tr>
    <td>Web Essentials</td>
    <td>Kimberly Willems</td>
  </tr>
</table>

```

```

table {
  border-collapse: collapse;
  width: 500px;
  table-layout: fixed;
  text-align: left;
  caption-side: bottom;
}
th, td {
  border: 1px solid black;
  padding: 5px;
}

```

Opleidingsonderdeel	Lector
Web Essentials	Kimberly Willems

Opleidingsonderdelen Graduaatsopleiding DVO

Rijgroepen

Om de opmaak van een tabel te vergemakkelijken kan een tabel ingedeeld worden in drie verschillende zones of rijgroepen:

- Een tabelkop of hoofding via het `<thead>`-element
- Een zone met de eigenlijke data van de tabel via het `<tbody>`-element
- Een tabelvoet via het `<tfoot>`-element

```
<table>
  <caption>Sinterklaas Wenslijst</caption>
  <thead>
    <tr>
      <th>Product</th>
      <th>Beschrijving</th>
      <th>Kostprijs</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Playstation 4 Slim 500GB</td>
      <td>Spelconsole van Sony</td>
      <td>299,99 EUR</td>
    </tr>
    <tr>
      <td>Mafia Trilogy PS4</td>
      <td>Basisgames en DLC's Mafia</td>
      <td>54,99 EUR</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Totaalprijs</td>
      <td></td>
      <td>354,98 EUR</td>
    </tr>
  </tfoot>
</table>
```



```
thead {
    background-color: lightcoral;
}
tbody {
    background-color: lightgray;
}
tfoot{
    background-color:coral;
}
```

Sinterklaas Wenslijst		
Product	Beschrijving	Kostprijs
Playstation 4 Slim 500GB	Spelconsole van Sony	299,99 EUR
Mafia Trilogy PS4	Basisgames en DLC's Mafia	54,99 EUR
Totaalprijs		354,98 EUR

Overspannen van kolommen en rijen

Soms kan het handig zijn om tabelcellen samen te voegen. Via het attribuut *colspan* kunnen we een aantal kolommen overspannen en via het attribuut *rowspan* kunnen we een aantal rijen overspannen.

```
<table>
  <tr>
    <td rowspan="3"></td>
    <th colspan="2">Studentenlijst</th>
  </tr>
  <tr>
    <td>Tommy</td>
    <td>Janssens</td>
  </tr>
  <tr>
    <td>Bart</td>
    <td>Peetermans</td>
  </tr>
</table>
```

```
table {
    border-collapse: collapse;
    text-align: center;
}
th, td {
    border: 1px solid black;
    width: 200px;
}
```


PXL DIGITAL	Studentenlijst	
	Tommy	Janssens
	Bart	Peetermans


Opeenvolgende tabellen

Het is perfect mogelijk om meerdere tabellen na elkaar op te nemen.

```
<table>
  <tr>
    <td rowspan="3"></td>
    <th colspan="2">Studentenlijst</th>
  </tr>
  <tr>
    <td>Tommy</td>
    <td>Janssens</td>
  </tr>
  <tr>
    <td>Bart</td>
    <td>Peetermans</td>
  </tr>
</table>
<table>
  <tr>
    <td rowspan="4"></td>
    <th colspan="3">Lectorenlijst</th>
  </tr>
  <tr>
    <td>Kimberly</td>
    <td>Willems</td>
    <td>Web Essentials</td>
  </tr>
  <tr>
    <td>Nicolas</td>
    <td>Schepers</td>
    <td>Digital Branding</td>
  </tr>
  <tr>
    <td>Kim</td>
    <td>Gijbels</td>
    <td>IT-organisation</td>
  </tr>
</table>
```

```
table {
  border-collapse: collapse;
  text-align: center;
  margin-bottom: 10px;
}
th, td {
  border: 1px solid black;
  width: 200px;
}
```

	Studentenlijst	
	Tommy	Janssens
	Bart	Peetermans

	Lectorenlijst		
	Kimberly	Willems	Web Essentials
	Nicolas	Schepers	Digital Branding
	Kim	Gijbels	IT-organisation

Geneste tabellen

Ook het nesten van tabellen is mogelijk. Hierbij is het aaneensluiten van de randen vaak wel een groot probleem, vandaar dat het weinig wordt gedaan in de praktijk.

```
<table>
  <tr>
    <th colspan="2">Activiteitenkalender</th>
  </tr>
  <tr>
    <td>Vergaderingen</td>
    <td>*</td>
  </tr>
  <tr>
    <td>Uitstappen</td>
    <td>**</td>
  </tr>
  <tr>
    <td>Vrije dag</td>
    <td>***</td>
  </tr>
  <tr>
    <td colspan="2">
      <table id="rooster">
        <tr>
          <td>MA</td>
          <td>DI</td>
          <td>WO</td>
          <td>DO</td>
          <td>VR</td>
        </tr>
        <tr>
          <td>*</td>
          <td>***</td>
          <td></td>
          <td>**</td>
          <td></td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

```
-----
table {
  border: 1px solid black;
  text-align: left;
  width: 500px;
  table-layout: fixed;
}
#rooster{
  width: 100%;
}
th, td {
  border: 1px solid black;
}
```

Activiteitenkalender				
Vergaderingen	*			
Uitstappen	**			
Vrije dag	***			
MA	DI	WO	DO	VR
*	***		**	

CSS3 voor tabellen

Via psuedo-klassen kunnen we tabellen eenvoudig een alternerende opmaak geven.

```
<table>
  <tr>
    <th>1</th>
    <th colspan="2">Alternerende opmaak in een tabel</th>
  </tr>
  <tr>
    <th>2</th>
    <td>Rij 2 kolom 2</td>
    <td>Rij 2 kolom 3</td>
  </tr>
  <tr>
    <th>3</th>
    <td>Rij 3 kolom 2</td>
    <td>Rij 3 kolom 3</td>
  </tr>
  <tr>
    <th>4</th>
    <td>Rij 4 kolom 2</td>
    <td>Rij 4 kolom 3</td>
  </tr>
  <tr>
    <th>5</th>
    <td>Rij 5 kolom 2</td>
    <td>Rij 5 kolom 3</td>
  </tr>
</table>
```

```
-----

table {
  border-collapse: collapse;
  width: 300px;
  text-align: center;
}
th, td {
  border: 1px solid black;
}
table tr:first-of-type {
  background-color: gray;
  color: white;
}
tr:nth-child(even) {
  background-color: lightpink;
}
tr:nth-child(odd) {
  background-color: lightblue;
}
tr > th:first-child {
  background-color: gray;
  color: white;
}
```

1	Alternerende opmaak in een tabel	
2	Rij 2 kolom 2	Rij 2 kolom 3
3	Rij 3 kolom 2	Rij 3 kolom 3
4	Rij 4 kolom 2	Rij 4 kolom 3
5	Rij 5 kolom 2	Rij 5 kolom 3

Basis structuur en positionering

Vroeger werden webpagina's ingedeeld in deelvensters met behulp van tabellen, maar hier zijn veel nadelen aan verbonden:

- Er is veel (extra) HTML-code nodig
- Er zijn veel minder mogelijkheden omdat alles in een tabelvorm moet passen
- ...

Tegenwoordig worden layers gebruikt om te komen tot de juiste positionering van de (structuur)elementen op een webpagina. Deze techniek positioneert elementen zonder veel interactie van of met andere elementen. Dit zorgt voor een onbegrensde vrijheid.

Alvorens over te gaan tot het algemeen positioneren van (structuur)elementen, gaan we eerst nog enkele onderdelen uit de voorgaande hoofdstukken herhalen én verdiepen. Daarnaast gaan we ook leren hoe we de standaardopmaak van een webpagina kunnen resetten om zo volledig zelf de opmaak te bepalen.

Tenslotte gaan we ook al enkele speciale manieren en mogelijkheden van positioneren leren via de stijleigenschap 'position'.

De structuurelementen

Structuurelementen kunnen ons helpen om de indeling van een webpagina vast te leggen. We hebben in het hoofdstuk 'Basis HTML' deze standaard HTML5-structuurelementen gezien:

- | | |
|------------------------------------|-------------------------------------------------|
| • Hoofding | <code><header>...</header></code> |
| • Navigatieblok | <code><nav>...</nav></code> |
| • Hoofdgedeelte | <code><main>...</main></code> |
| • Sectie (deel van een geheel) | <code><section>...</section></code> |
| • Artikel (losstaand geheel) | <code><article>...</article></code> |
| • Voetnoot of voettekst | <code><footer>...</footer></code> |
| • Randinformatie | <code><aside>...</aside></code> |
| • (Verzamelingen van) afbeeldingen | <code><figure>...</figure></code> |

Naast de standaard structuurelementen kunnen we ook een `<div>`-element gebruiken. Dit element gebruiken we voornamelijk voor blokelementen zonder specifieke betekenis.

```
<div id="geenspecifiekebetekenis">
    ...
</div>
```

Stijleigenschap display

In het hoofdstuk 'Basis HTML' hebben we gezien dat er twee types van HTML-elementen bestaan:

1. *Blokelementen*: deze elementen nemen altijd een nieuwe regel in beslag en maken gebruik van de volledige breedte van het scherm. Deze elementen staan in een blokcontainer of blokbox.
2. *Inline elementen*: deze elementen worden in de huidige regel getoond en zijn even breed als de inhoud van het element.

Het is belangrijk dat we het verschil tussen een blok- en inline-elementen kunnen vaststellen en, indien nodig voor de gewenste positionering, kunnen aanpassen. Dit kunnen we doen via de stijleigenschap 'display'.

Waarde	Uitleg
block	Het element wordt een blok-element in een blokcontainer en neemt een volledig nieuwe regel in beslag.
inline-block	Het element blijft een inline-element, maar kan zonder problemen met blokcontainer stijleigenschappen overweg.
inline	Het element wordt een inline-element en wordt in de huidige regel getoond.
none	Het element wordt onzichtbaar en neemt geen plaats meer in op de webpagina.

Stijleigenschap box-sizing

In het hoofdstuk 'Basis CSS' hebben we het box-model gezien. Elk blokelement zit in een blokcontainer en kan voorzien worden van padding, border en margin.

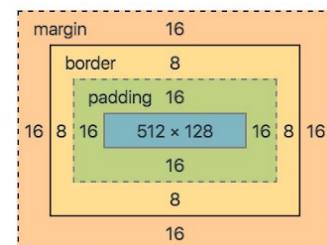
De stijleigenschappen padding, border en margin hebben invloed op de totale breedte of hoogte van een blokelement.

Eigenlijke breedte = breedte element + padding + border + margin

$$592 = 512 + (16 \times 2) + (8 \times 2) + (16 \times 2)$$

Eigenlijke hoogte = hoogte element + padding + border + margin

$$208 = 128 + (16 \times 2) + (8 \times 2) + (16 \times 2)$$



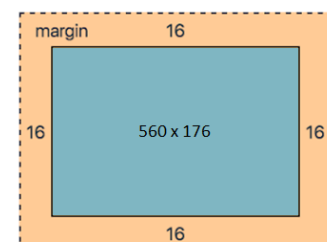
Via stijleigenschap 'box-sizing' en waarde 'border-box' kunnen we van deze standaard optelsom afwijken door in de breedte van het element, ook de padding en de border op te nemen. Deze afwijking maakt het makkelijker om met meerdere meeteenheden tegelijkertijd te werken op een webpagina. Enkel de margin moet dan nog apart worden berekend.

Eigenlijke breedte = breedte element + margin

$$592 = 560 + (16 \times 2)$$

Eigenlijke hoogte = hoogte element + margin

$$208 = 176 + (16 \times 2)$$



Standaardopmaak resetten

Door volgend stukje CSS-code toe te voegen bovenaan ieder CSS-bestand, kunnen we het onszelf wat makkelijker maken om onze eigen, unieke CSS-code te schrijven.

Ieder HTML-element heeft een standaardopmaak. Deze standaardopmaak is verschillend per webbrowser. Dit stukje CSS-code verwijdert alle standaard ingestelde witruimte binnen en buiten de HTML-elementen en maakt de berekening van de breedtes, ongeacht de gebruikte meeteenheden, eenvoudiger.

```
* {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}
```

Stijleigenschap position

Om specifieke elementen een speciale positionering te geven, kunnen we werken met de stijleigenschap 'position' en volgende waarden:

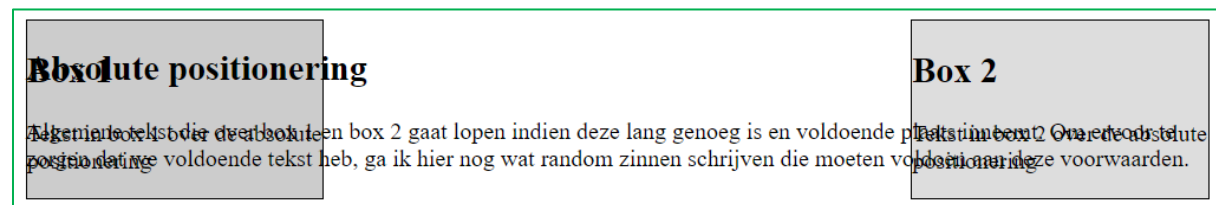
- *Static*: dit is de standaardwaarde. Dit zorgt ervoor dat het HTML-element op zijn normale plaats komt te staan volgens de standaard flow layout.
- *Absolute*: dit zorgt ervoor dat het HTML-element op een aangegeven plaats binnen het parent HTML-element of het browservenster vastgezet wordt.
- *Relative*: dit zorgt ervoor dat het HTML-element verschoven wordt te opzichte van zijn originele plaats.
- *Fixed*: dit zorgt ervoor dat het HTML-element vastgelijmd wordt aan een plaats binnen het zichtbare browservenster. Het effect is alleen zichtbaar als er een schuifbalk aanwezig is.

OPGELET! We passen de standaardwaarde 'static' enkel aan wanneer we specifieke elementen een speciale of unieke positionering willen geven. De stijleigenschap 'position' gebruiken we niet om alle (structuur)elementen van een webpagina te voorzien van positionering.

Absolute positionering

Een absoluut gepositioneerd element gedraagt zich als een aparte laag en oefent geen invloed uit op andere elementen. Hierdoor speelt de volgorde van de absoluut gepositioneerde elementen in de HTML-code helemaal geen rol.

De afstanden of positie van een absoluut gepositioneerd element worden bepaald door de stijleigenschappen 'top', 'right', 'bottom' en 'left'. Bij het wijzigen van de grootte van het browservenster blijft deze afstand ongewijzigd.



```
#box1 {
    position: absolute;
    top: 10px;
    left: 10px;
    width: 200px;
    border: 1px solid #000000;
    background-color: #cccccc;
}
#box2 {
    position: absolute;
    top: 10px;
    right: 10px;
    width: 200px;
}
```

```
border: 1px solid #000000;
background-color: #dddddd;

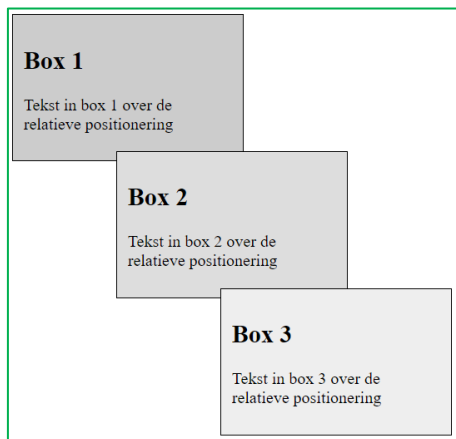
}
#inhoud {
    position: absolute;
    top: 10px;
    right: 10px;
    left: 10px;
}

-----

<aside id="box1">
    <h1>Box 1</h1>
    <p>Tekst in box 1 over de absolute positionering</p>
</aside>
<aside id="box2">
    <h1>Box 2</h1>
    <p>Tekst in box 2 over de absolute positionering</p>
</aside>
<article id="inhoud">
    <h1>Absolute positionering</h1>
    <p>Algemene tekst die over box 1 en box 2 gaat lopen indien deze lang
        genoeg is en voldoende plaats inneemt. Om ervoor te zorgen dat we
        voldoende tekst heb, ga ik hier nog wat random zinnen schrijven
        die moeten voldoen aan deze voorwaarden. </p>
</article>
```

Relatieve positionering

De positie van een relatief gepositioneerd element is relatief ten opzichte van de normale positie die dit element zou hebben zonder positiebepaling. De normale positie wordt bepaald door de plaats van dit element in de HTML-code. Zo komen alle opeenvolgende blokelementen onder elkaar en alle inline-elementen naast elkaar.



```
#box1 {
    position: relative;
    border: 1px solid #000000;
    background-color: #cccccc;
    padding: 10px;
    width: 200px;
}
#box2 {
    position: relative;
    top: -10px;
    left: 100px;
    border: 1px solid #000000;
    background-color: #dddddd;
    padding: 10px;
    width: 200px;
}
#box3 {
    position: relative;
    top: -20px;
```

```

    left: 200px;
    border: 1px solid #000000;
    background-color: #eeeeee;
    padding: 10px;
    width: 200px;
}
-----
<aside id="box1">
    <h1>Box 1</h1>
    <p>Tekst in box 1 over de relatieve positionering</p>
</aside>
<aside id="box2">
    <h1>Box 2</h1>
    <p>Tekst in box 2 over de relatieve positionering</p>
</aside>
<aside id="box3">
    <h1>Box 3</h1>
    <p>Tekst in box 3 over de relatieve positionering</p>
</aside>

```

Door het opgeven van de stijleigenschappen 'top', 'right', 'bottom' en 'left' kan de positie gewijzigd worden. Een relatief gepositioneerd element blijft wel ruimte innemen op de normale positie.

Met de instelling `position: relative` kunnen inline elementen bijvoorbeeld hoger of lager dan de normale tekst geplaatst worden. Dit kan natuurlijk ook meer naar links of meer naar rechts of met een combinatie van beiden.

```

span {
    font-family: Arial, Helvetica, sans-serif;
    color: #999999;
    font-size: 1.2em;
}
.hoger {

```

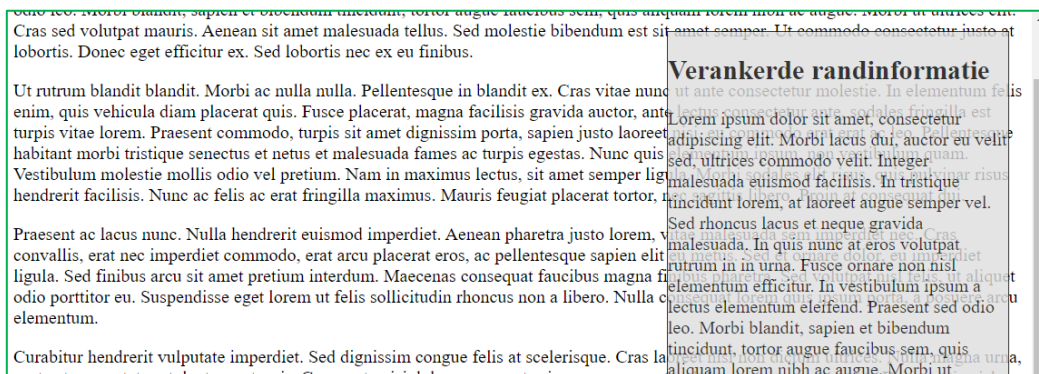
```

        position: relative;
        bottom: 10px;
    }
    .lager {
        position: relative;
        bottom: -10px;
    }
    .links{
        position: relative;
        left: -15px;
    }
    .rechts {
        position: relative;
        right: -15px;
    }
}
-----
<p>Met de instelling position: relative kunnen inline elementen
    bijvoorbeeld <span class="hoger">hoger</span> of <span
        class="lager">lager</span> dan de normale tekst geplaatst worden.</p>
<p>Dit kan natuurlijk ook meer naar <span class="links">links</span> of
    meer naar <span class="rechts">rechts</span> of met een <span
        class="links lager">combinatie</span> van beiden.</p>

```

Gefixeerde (vaste) positionering

Een fixed gepositioneerd element wordt vast verankerd binnen het zichtbare veld van het browservenster en scrolt niet mee met de rest van de inhoud.



```

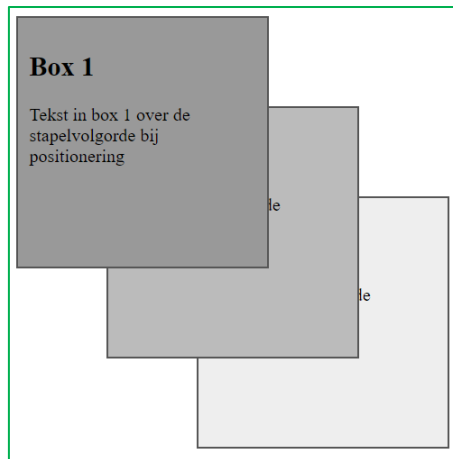
aside#box1 {
    position: fixed;
    top: 20px;
    right: 20px;
    width: 300px;
    border: 1px solid #000000;
    background-color: #dddddd;
    opacity: 0.8;
}
-----
...
...
<p>Curabitur hendrerit vulputate imperdiet. Sed dignissim congue ...</p>
<p>Suspendisse pretium, enim vel viverra pretium, ex ex pharetra ...</p>
<aside id="box1">
    <h1>Verankerde randinformatie</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
</aside>

```

Stijleigenschap z-index

Elementen kunnen op elkaar gestapeld worden. De laag die als eerste staat in de HTML-code, komt dan onderaan te liggen en alle volgende lagen zullen hierop gepositioneerd worden in de volgorde waarop ze in de code staan. Dit geeft niet altijd het gewenste effect.

Via de stijleigenschap 'z-index' kunnen we, bij absolute positionering en fixed positionering, de manier van stapelen aanpassen aan onze wensen. Hierbij ligt een hoger z-nummer bovenop een object met een lager z-nummer. Er kunnen ook negatieve getallen gebruikt worden.



```
aside {  
    position: absolute;  
    border: 2px solid #555555;  
    padding: 10px;  
    width: 200px;  
    min-height: 200px;  
}  
#box1 {  
    top: 10px;  
    left: 10px;  
    background-color: #999999;  
    z-index: 3;  
}  
#box2 {  
    top: 90px;  
    left: 90px;  
    background-color: #BBBBBB;  
    z-index: 2;  
}  
#box3 {  
    top: 170px;  
    left: 170px;  
    background-color: #EEEEEE;  
    z-index: 1;  
}  
-----
```

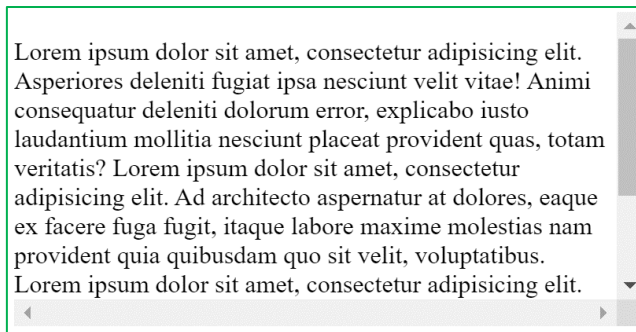
```
<aside id="box1">
  <h1>Box 1</h1>
  <p>Tekst in box 1 over de stapelvolgorde bij positionering</p>
</aside>
<aside id="box2">
  <h1>Box 2</h1>
  <p>Tekst in box 2 over de stapelvolgorde bij positionering</p>
</aside>
<aside id="box3">
  <h1>Box 3</h1>
  <p>Tekst in box 3 over de stapelvolgorde bij positionering</p>
</aside>
```

Stijleigenschap overflow

De hoogte en breedte van een structuurelement wordt standaard bepaald door de inhoud van dat structuurelement. Echter kan het voorvallen dat we een structuurelement een vaste breedte en hoogte wensen te geven, ongeacht de inhoud. Dit kan overlopende inhoud, ook wel overflow genoemd, veroorzaken.

Met de opmaakeigenschappen 'overflow', 'overflow-x' en 'overflow-y', kunnen we bepalen wat er moet gebeuren indien er overlopende inhoud aanwezig is.

Waarde	Uitleg
visible	De overlopende inhoud wordt getoond buiten het structuurelement.
hidden	De overlopende inhoud wordt afgekapt en is onzichtbaar.
scroll	Ongeacht of er al dan niet overlopende inhoud is, komt er een schuifbalk in het structuurelement te staan. De overlopende inhoud wordt afgekapt, maar kan met de schuifbalk in beeld worden gebracht.
auto	Bij overlopende inhoud, komt er een schuifbalk in het structuurelement te staan. De overlopende inhoud wordt afgekapt, maar kan met de schuifbalk in beeld worden gebracht.



```
<section>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Asperiores
    deleniti fugiat ipsa nesciunt velit vitae! Animi consequatur ...</p>
</section>
-----
section {
  height: 200px;
  width: 400px;
  overflow: scroll;
}
```

Rekenfunctie calc()

Sinds CSS3 is het mogelijk om de rekenfunctie `calc()` te gebruiken in de CSS-code. Met `calc()` kunnen we eenvoudig waardes, bestaande uit verschillende meeteenheden, optellen (+), aftrekken (-), vermenigvuldigen (*) en delen (/).

Dit maakt het mogelijk om, bijvoorbeeld, een even of oneven aantal structuurelementen, met een gelijke breedte of hoogte, naast of onder elkaar te plaatsen.

Deze drie sections passen samen perfect binnen de main en zijn alle drie even hoog. Dit komt door de rekenfunctie $\text{calc}((100\% / 3) - ((4 * 36\text{px}) / 3))$.

Voor de berekening van de hoogte van de sections nemen we de hoogte van het parent-element als 100% en deze delen we door het aantal sections ($100\% / 3$). Je kan ook werken met de ingestelde hoogte van het parent-element in pixels ($600\text{px} / 3$).

Daarna kijken we naar de ingestelde margin van 36px op de sections. Deze doen we maal vier (drie sections met samengevoegde verticale margins is vier keer een margin van 36px) en delen we door het aantal sections ($((4 * 36\text{px}) / 3)$).

```
<main>
  <section>
    <p>Deze drie sections passen samen perfect binnen de main en zijn
      alle drie even hoog. Dit komt door de rekenfunctie
       $\text{calc}((100\% / 3) - ((4 * 36\text{px}) / 3))$ .</p>
  </section>
  <section>
    <p>Voor de berekening van de hoogte van de sections nemen we de
      hoogte van het parent-element als 100% en deze delen we door het
      aantal sections ( $100\% / 3$ ). Je kan ook werken met de ingestelde
      hoogte van het parent-element in pixels ( $600\text{px} / 3$ ).</p>
  </section>
  <section>
    <p>Daarna kijken we naar de ingestelde margin van 36px op de
      sections. Deze doen we maal vier (drie sections met samengevoegde
      verticale margins is vier keer een margin van 36px) en delen we
      door het aantal sections ( $((4 * 36\text{px}) / 3)$ ).</p>
  </section>
</main>

-----
* {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
```

```
}  
main {  
    width: 800px;  
    height: 600px;  
    background-color: #ccc;  
    overflow: hidden;  
}  
section {  
    height: calc((100% / 3) - ((36px * 4) / 3));  
    margin: 36px;  
    padding: 16px;  
    background-color: #fff;  
}
```

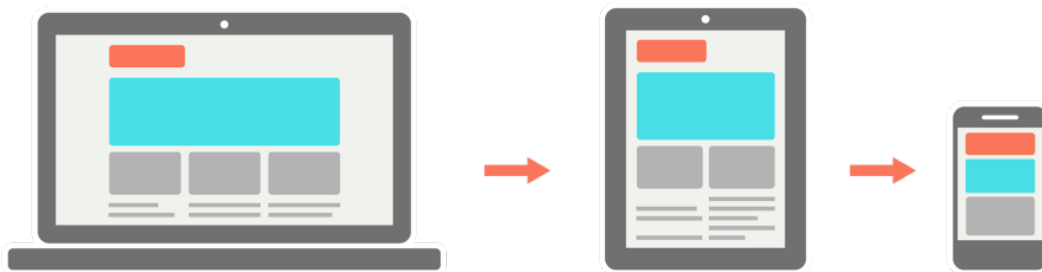
OPGELET! In bovenstaand voorbeeld is op het structuurelement 'main' een overflow 'hidden' ingesteld. Dit is één van de mogelijke noodoplossingen om het vervelende probleem met de automatisch samengevoegde verticale margin tussen de child-structuurelementen en het parent-structuurelement in de standaard flow layout op te lossen.

Dit probleem lost zich vanzelf op wanneer we, later in deze cursus, gaan leren om Flexbox en Bootstrap Grid te gebruiken om af te wijken van de standaard flow layout. De automatisch samengevoegde verticale margin verdwijnt dan. Het instellen van een overflow is dan niet meer nodig en de berekening van de hoogte via calc() is dan ook veel eenvoudiger.

Responsive webdesign

Bij responsive webdesign gaan we de webpagina zo ontwerpen dat deze toegankelijk is via een breed scala van apparaten (computer, tablet, smartphone, ...). De webpagina zal op basis van de grootte van het scherm geschaald worden om zo te komen tot een optimale leesbaarheid en gebruiksvriendelijkheid.

Elementen kunnen, afhankelijk van het apparaat en de schermgrootte, verplaatst, gewijzigd of verwijderd worden.



Responsive Web Design

Mobile First Web Design




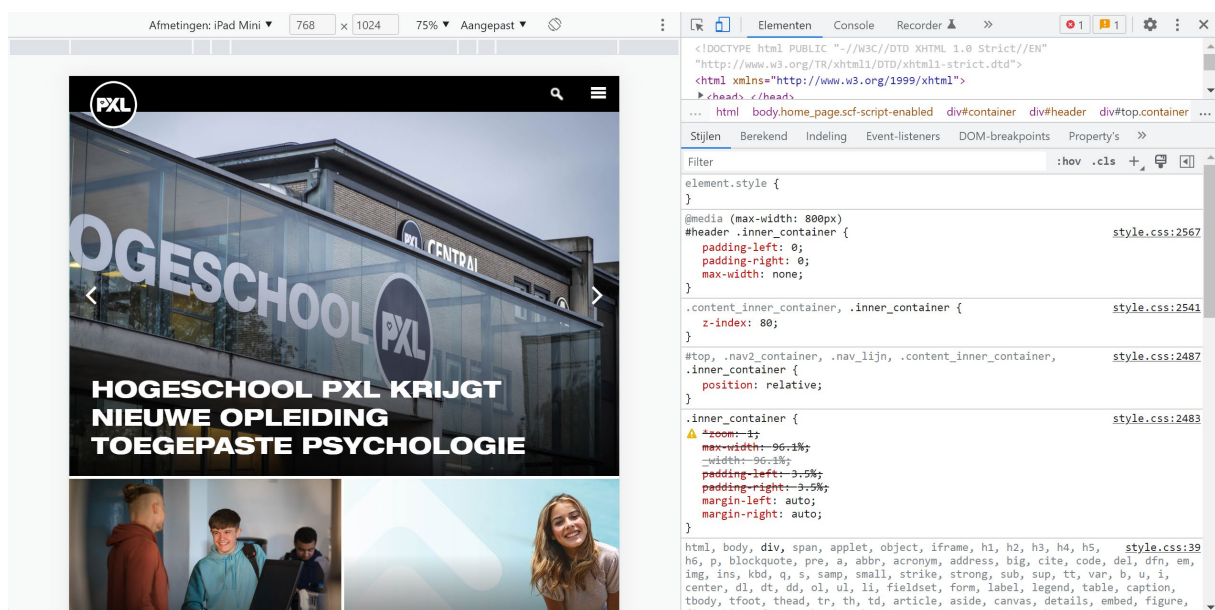
We kunnen op twee verschillende manieren 'responsive' schrijven in de CSS-code:

- *Desktop First Responsive Web Design*: we schrijven in de CSS-code eerst de opmaak voor het grootste scherm en daarna voor de kleinere schermen.
- *Mobile First Web Design*: we schrijven in de CSS-code eerst de opmaak voor het kleinste scherm en daarna voor de grotere schermen.

Responsiviteit controleren

Via de Chrome DevTools kunnen we de responsiviteit van een webpagina controleren:

- Open de Chrome DevTools.
- Klik op de knop  of gebruik de toetsencombinatie Ctrl+Shift+M.
- Kies de gewenste schermafmetingen of apparaat.
- Controleer de responsiviteit van de webpagina.
- Bekijk, indien er elementen afwijken, de bijhorende HTML- en CSS-code.



Viewport

De viewport is het 'door de gebruiker zichtbare deel van een webpagina'. De viewport varieert per apparaat. Zo is de viewport van een smartphonescherm kleiner dan die van een computerscherm.

Vroeger werden webpagina's voornamelijk op computerschermen bekeken. Apparaten met een kleiner scherm herschaalden de webpagina om op het scherm te passen, maar dat was niet altijd wenselijk voor de leesbaarheid en de gebruiksvriendelijkheid van de webpagina.



De viewport geeft de mogelijkheid om de breedte van de content aan te passen aan de grootte van het browservenster. De viewport is een specifiek <meta>-element in het <head>-element.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

OPGELET! Vanaf nu gaan we op iedere webpagina de viewport in onze HTML-code meegeven.

Minimale en maximale hoogtes en breedtes

We kunnen HTML-elementen een breedte en een hoogte gegeven via de stijleigenschappen 'width' en 'height'.

Om gedeeltelijk tegemoet te komen aan eventuele verschillen in resoluties en groottes van digitale schermen, kunnen we ook een minimum- of maximumbreedte en een minimum- of maximumhoogte opgeven via de stijleigenschappen 'min-width', 'max-width', 'min-height' en 'max-height'.

Deze box heeft een vaste breedte van 600px.

Deze box heeft een maximale breedte van 600px.

Deze box heeft een minimale breedte van 600px.

Deze box heeft een vaste breedte van 600px.

Deze box heeft een maximale breedte van 600px.

Deze box heeft een minimale breedte van 600px.

```
aside {
  border: 3px solid #1ba96c;
}
#box1 {
  width: 600px;
}
#box2 {
  max-width: 600px;
}
#box3 {
  min-width: 600px;
}
-----
<aside id="box1">
  <p>Deze box heeft een vaste breedte van 600px.</p>
</aside>
<aside id="box2">
  <p>Deze box heeft een maximale breedte van 600px.</p>
</aside>
<aside id="box3">
  <p>Deze box heeft een minimale breedte van 600px.</p>
</aside>
```

Media queries

CSS3 media queries zijn de basisoplossing om webpagina's responsive te maken. Via media queries kunnen we opmaakeigenschappen koppelen aan het weergavemedium. Mogelijke weergavemedia zijn hierbij:

- *All*: alle soorten apparaten
- *Screen*: beeldschermen, tablets, smartphones, ...
- *Print*: printers
- *Speech*: spraakbrowsers of voorleessoftware

Een media query is een vergelijking (if-conditie). In de media query geeft u het weergavemedium en de mediakenmerken op. Als het weergavemedium overeenkomt en de mediakenmerken kloppen, dan worden de bijhorende stijlregels uitgevoerd.

Media queries gebruiken in CSS

Een media query heeft in de CSS volgende algemene opbouw:

```
@media WEERGAVEMEDIUM and (MEDIAKENMERKEN) {  
    selector {  
        stijldeclaraties  
    }  
    selector {  
        stijldeclaraties  
    }  
    ...  
}
```


Hierbij zijn mogelijke mediakenmerken:

- *Min-width*: minimale breedte
- *Max-width*: maximale breedte
- *Orientation*: landschap- of portret-weergave
- *Min-height*: minimale hoogte
- *Max-height*: maximale hoogte

Met media queries kunnen we dus diverse breekpunten in de lay-out vastleggen. Op deze manier kan de opmaak worden afgestemd op de verschillende schermresoluties.

De meest gangbare schermresoluties zijn momenteel:

- 480 x 320
- 640 x 360
- 1024 x 768
- 1280 x 1024
- 1366 x 768
- 1400 x 1050
- 1680 x 1050
- 1920 x 1080
- 3840 x 2160
- ...

```
@media screen and (min-width: 480px) {  
    header {  
        background-image: url(bg-small.jpg);  
    }  
}  
  
@media screen and (min-width: 1024px) {  
    header {  
        background-image: url(bg-big.jpg);  
    }  
}
```

Daarnaast kunnen we ook een verschil maken in de lay-out op basis van de oriëntatie van het scherm. Zo kan de website er anders uitzien in portretweergave dan in landschap weergave.

```
@media screen and (orientation: landscape) {  
    p {  
        font-size: 1.25rem;  
    }  
}  
@media screen and (orientation: portrait) {  
    p {  
        font-size: 1rem;  
    }  
}
```

Media queries gebruiken in HTML

We kunnen voor iedere mediaweergave een apart stijlblad maken. Daarna geven we via media queries in de HTML-code aan welk stijlblad bij welke mediaweergave hoort.

```
<link rel="stylesheet" href="stylescreen.css" media="screen">  
<link rel="stylesheet" href="styleprint.css" media="print">
```

OPGELET! Wij gaan in de oefeningen enkel de media queries gebruiken in een CSS-stijlblad en niet in de HTML-code.

Werken met meeteenheid rem

De meeteenheid 'rem' heeft de laatste jaren enorm veel aan populariteit gewonnen. Rem staat voor 'root em' en refereert naar de tekstgrootte van het rootelement. Rem is ideaal om te gebruiken in combinatie met een responsief design.

De standaard tekstgrootte van het rootelement is voor de meeste apparaten en hun webbrowsers 16px, maar dat kunnen we dus aanpassen door op het rootelement een andere tekstgrootte in te stellen. 1rem is dan gelijk aan de ingestelde tekstgrootte van het rootelement.

```
// rootelement met lettergrootte 1.25 x 16px = 20px.  
html {  
    font-size: 1.25rem;  
}  
  
// hoofding van niveau 2 met lettergrootte 1.50 x 20px = 30px.  
h2 {  
    font-size: 1.50rem;  
}
```

Flexbox

Met CSS3 Flexbox kunnen we makkelijker een responsief design mét positionering voorzien, zonder gebruik te maken van verouderde manieren van positionering.

Wanneer we aan de slag gaan met CSS3 Flexbox, moeten we minstens volgende onderdelen voorzien in onze HTML- en CSS-code met behulp van onze gekende structuurelementen:

- Een flex-container
- Flex-items in de flex-container

```
<main class="flex-container">
  <article id="deel1"></article>
  <article id="deel2"></article>
  <article id="deel3"></article>
</main>
```

Flex-container

Aan de flex-container kunnen we volgende stijleigenschappen toevoegen:

- display
- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

Display

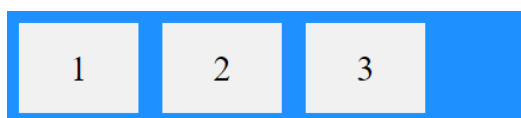
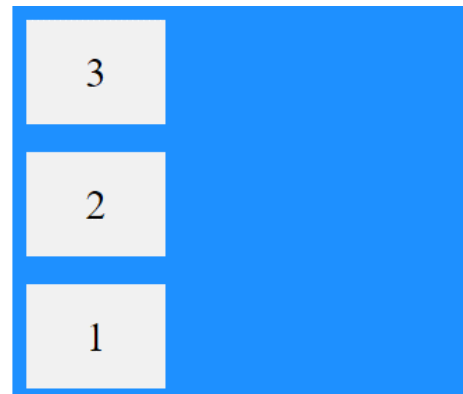
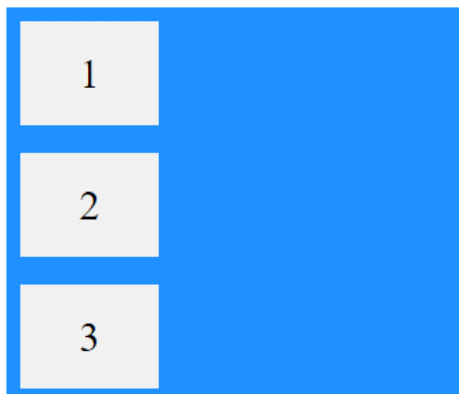
Via de display eigenschap kunnen we ervoor zorgen dat de flex-container flexibel wordt.

```
.flex-container {  
    display: flex;  
}
```

Flex-direction

Via de flex-direction eigenschap kunnen we de richting van de flex-items instellen. Volgende waarden zijn hierbij mogelijk:

- *column*: de flex-items staan verticaal van boven naar onder.
- *column-reverse*: de flex-items staan verticaal van onder naar boven.
- *row*: de flex-items staan horizontaal van links naar rechts.
- *row-reverse*: de flex-items staan horizontaal van rechts naar links.

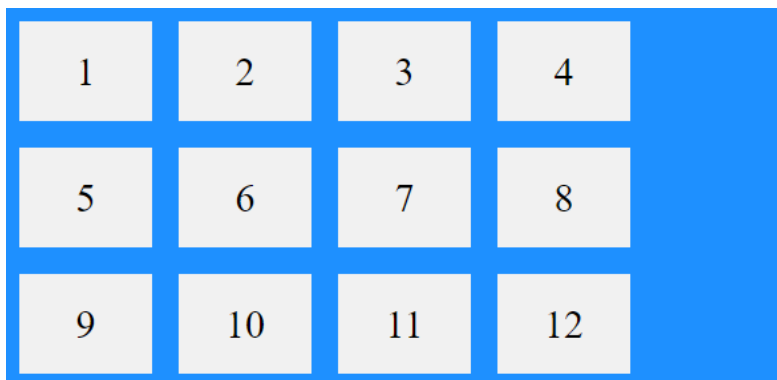
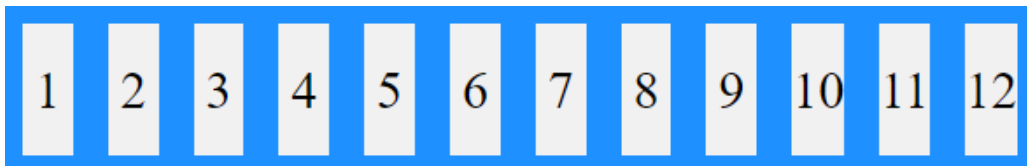


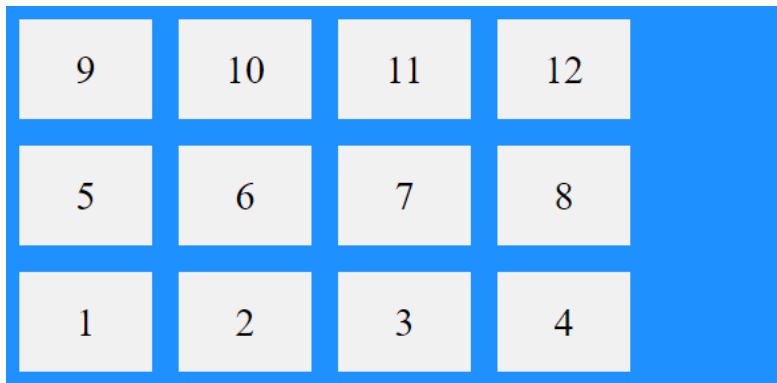
```
.flex-container {  
    display: flex;  
    flex-direction: row;  
}
```

Flex-wrap

Via de `flex-wrap` eigenschap kunnen we aangeven of alle flex-items op één of meerdere rijen of kolommen moeten komen te staan. Volgende waarden zijn hierbij mogelijk:

- *nowrap*: dit is de standaardwaarde. De flex-items moeten op dezelfde rij of kolom blijven staan en mogen niet naar een volgende rij of kolom gaan.
- *wrap*: de flex-items mogen naar de volgende rij of kolom gaan indien de flex-container niet hoog of breed genoeg is.
- *wrap-reverse*: de flex-items mogen naar de volgende rij of kolom gaan indien de flex-container niet hoog of breed genoeg is, maar doen dit in de omgekeerde volgorde.





```
.flex-container {  
  display: flex;  
  flex-direction: row;  
  flex-wrap: wrap;  
}
```

Flex-flow

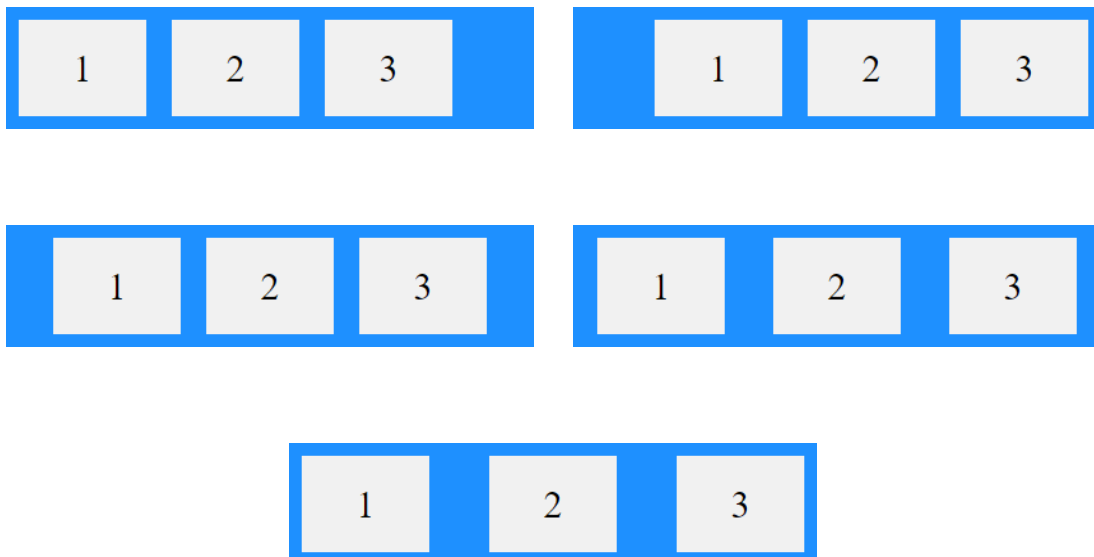
Via de `flex-flow` eigenschap kunnen we de `flex-direction` en `flex-wrap` op basis van één regel instellen.

```
.flex-container {  
  display: flex;  
  flex-flow: row wrap;  
}
```

Justify-content

Via de `justify-content` eigenschap kunnen we de lay-out van de flex-items in de container bepalen. Volgende waarden zijn hierbij mogelijk:

- *flex-start*: dit is de standaardwaarde. De flex-items staan aan het begin van de flex-container.
- *flex-end*: de flex-items staan aan het einde van de flex-container.
- *center*: de flex-items staan in het midden van de flex-container.
- *space-around*: de flex-items hebben ruimte ervoor, ertussen en erachter.
- *space-between*: de flex-items hebben ruimte tussen ieder item.
- *space-evenly*: de flex-items hebben een gelijke verdeling van ruimte.



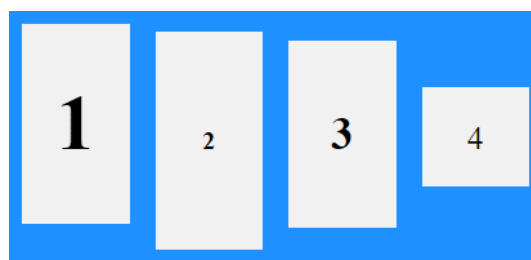
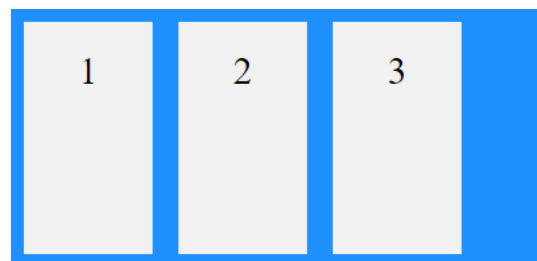
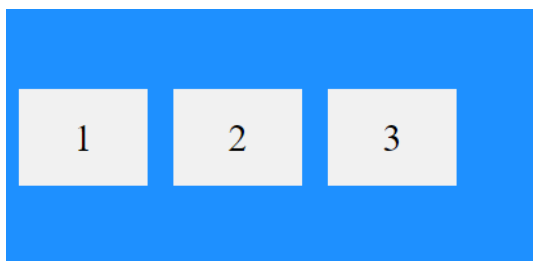
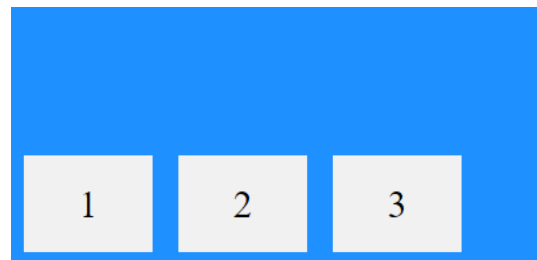
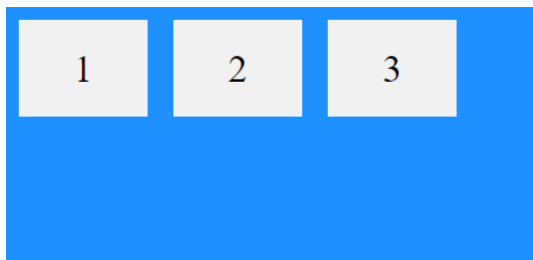
```
.flex-container {  
  display: flex;  
  justify-content: center;  
}
```


Align-items

Via de align-items eigenschap kunnen we de verticale uitlijning van de flex-items instellen.

Volgende waarden zijn hierbij mogelijk:

- *flex-start*: dit is de standaardwaarde. De flex-items staan bovenaan.
- *flex-end*: de flex-items staan onderaan.
- *center*: de flex-items worden verticaal gecentreerd.
- *stretch*: de flex-items worden uitgetrokken van boven tot onder.
- *baseline*: de flex-items worden over hun baseline verticaal gecentreerd.



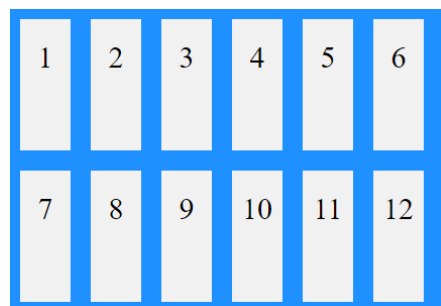
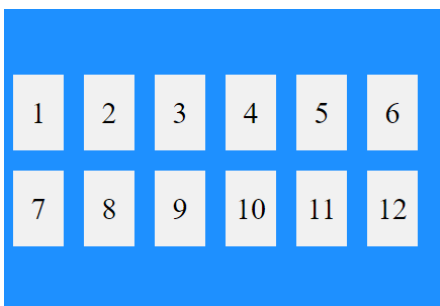
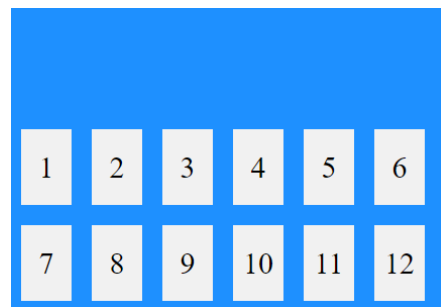
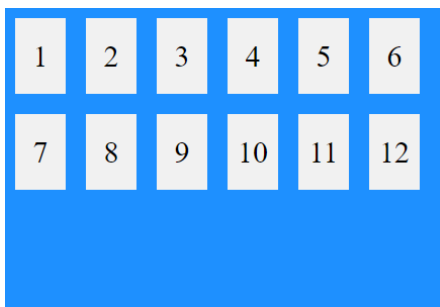
```
.flex-container {  
    display: flex;  
    align-items: center;  
}
```

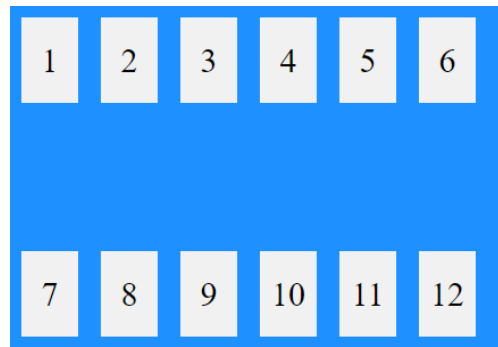
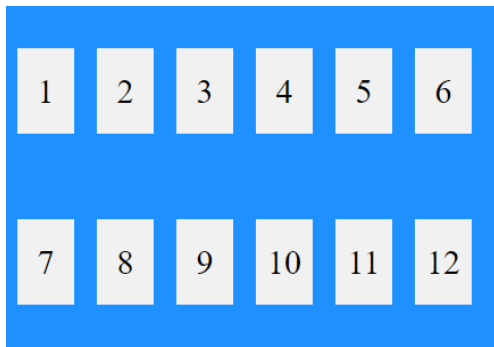
Align-content

Via de align-content eigenschap kunnen we de regels met flex-items verticaal uitlijnen.

Volgende waarden zijn hierbij mogelijk:

- *flex-start*: dit is de standaardwaarde. De regels met flex-items staan aan het begin van de flex-container.
- *flex-end*: de regels met flex-items staan aan het einde van de flex-container.
- *center*: de regels met flex-items staan in het midden van de flex-container.
- *stretch*: de regels met flex-items worden uitgetrokken over de hele de flex-container.
- *space-around*: de regels met flex-items hebben ruimte ervoor, ertussen en erachter.
- *space-between*: de regels met flex-items hebben ruimte tussen ieder item.





```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
  align-items: center;  
}
```

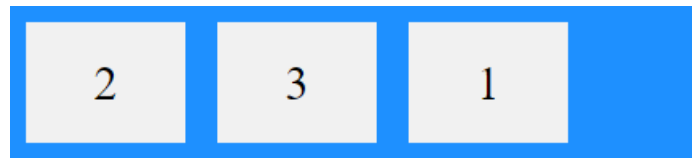
Flex items

Flex-items worden automatisch flexibel in een flex-container. Aan de flex-items kunnen we volgende stijleigenschappen toevoegen:

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

Order

Via de order eigenschap kunnen we de volgorde van de flex-items in de flex-container aanpassen.



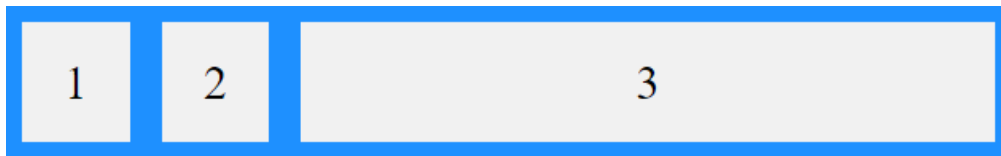
```
<main class="flex-container">
  <article id="deel1">1</article>
  <article id="deel2">2</article>
  <article id="deel3">3</article>
</main>

-----
#deel1 {
  order: 3;
}
#deel2 {
  order: 1;
}
#deel3 {
  order: 2;
}
```

Flex-grow

We kunnen flex-items een vaste hoogte of breedte geven via de CSS-code, maar we kunnen er ook voor kiezen om de hoogte en/of breedte flexibel in te stellen en de flex-items te laten groeien of krimpen in relatie tot de andere items. Dit kan via `flex`, `flex-grow`, `flex-shrink` en `flex-basis`.

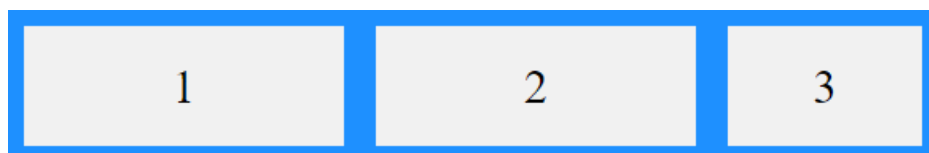
Via de `flex-grow` eigenschap kunnen we aangeven hoeveel een flex-item relatief moet groeien tegenover de rest van de flex-items.



```
<main class="flex-container">
  <article id="deel1">1</article>
  <article id="deel2">2</article>
  <article id="deel3">3</article>
</main>
-----
#deel1 {
  flex-grow: 1;
}
#deel2 {
  flex-grow: 1;
}
#deel3 {
  flex-grow: 8;
}
```

Flex-shrink

De flex-shrink eigenschap is het tegenovergestelde van de flex-grow eigenschap. Via de flex-shrink eigenschap kunnen we aangeven hoeveel een flex-item relatief moet krimpen tegenover de rest van de flex-items.



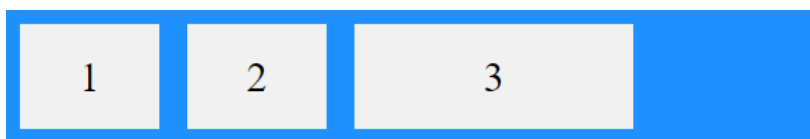
```
<main class="flex-container">
  <article id="deel1">1</article>
  <article id="deel2">2</article>
  <article id="deel3">3</article>
</main>

-----

#deel3 {
  flex-shrink: 2;
}
```

Flex-basis

Via de flex-basis eigenschap kunnen we de initiële lengte van een flex-item vastleggen.



```
<main class="flex-container">
  <article id="deel1">1</article>
  <article id="deel2">2</article>
  <article id="deel3">3</article>
</main>

-----

#deel3 {
  flex-basis: 200px;
}
```

Flex

Via de flex eigenschap kunnen we de flex-grow, flex-shrink en flex-basis verkort instellen. Met deze verkorte notatie krijgen de niet-ingestelde eigenschappen automatisch de meest optimale waarden.

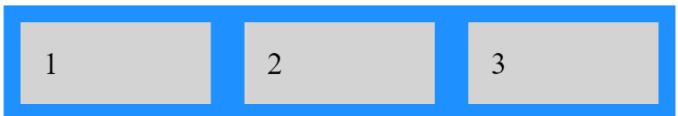
Volgende waarden zijn hierbij mogelijk:

- *<flex-grow> <flex-shrink> <flex-basis>*: het item krijgt een initiële lengte en specifieke waarde voor groeien en krimpen.
- *initial*: verkorte instelling voor '0 1 auto'. Het item mag niet groeien, maar wel krimpen. De basisafmeting wordt bepaald door de ingestelde breedte/hoogte.
- *auto*: verkorte instelling voor '1 1 auto'. De beschikbare ruimte in de container wordt volledig benut. Groeien en krimpen gebeurt naar behoefte.
- *none*: verkorte instelling voor '0 0 auto'. Het item krijgt de waarde van de ingestelde breedte/hoogte en is niet flexibel. Er is geen groei en geen krimp.
- *<getal>*: verkorte instelling voor '<getal> 1 0'. Het item groeit met de vrije ruimte in de container. Het getal geeft het proportionele deel van de vrije ruimte aan. Als alle items hetzelfde getal hebben, krijgen ze een gelijk deel van de vrije ruimte.

```
#deel1, #deel2, #deel3 {  
    flex: initial;  
}
```



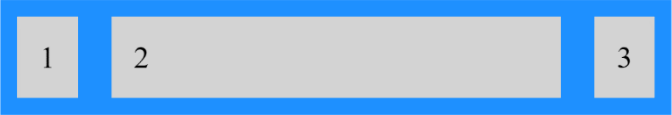
```
#deel1, #deel2, #deel3 {  
    flex: auto;  
}
```



```
#deel1, #deel2, #deel3 {  
    flex: none;  
}
```



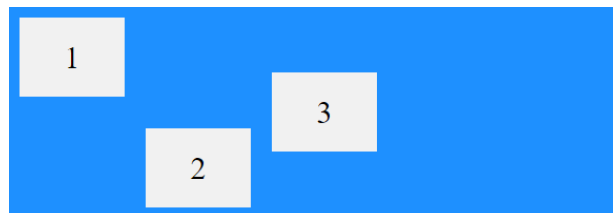
```
#deel2 {  
    flex: 1;  
}
```



Align-self

Via de `align-self` eigenschap kunnen we de verticale uitlijning van één of meerdere flex-items apart aanpassen. De `align-self` eigenschap overschrijft de `align-items` eigenschap van de flex-container. Volgende waarden zijn hierbij mogelijk:

- *flex-start*: dit is de standaardwaarde. De flex-items staan bovenaan.
- *flex-end*: de flex-items staan onderaan.
- *center*: de flex-items worden verticaal gecentreerd.
- *stretch*: de flex-items worden uitgetrokken van boven tot onder.
- *baseline*: de flex-items worden over hun baseline verticaal gecentreerd.



```
#deel1 {  
    align-self: flex-start;  
}  
#deel2 {  
    align-self: flex-end;  
}  
#deel3 {  
    align-self: center;  
}
```


Bootstrap

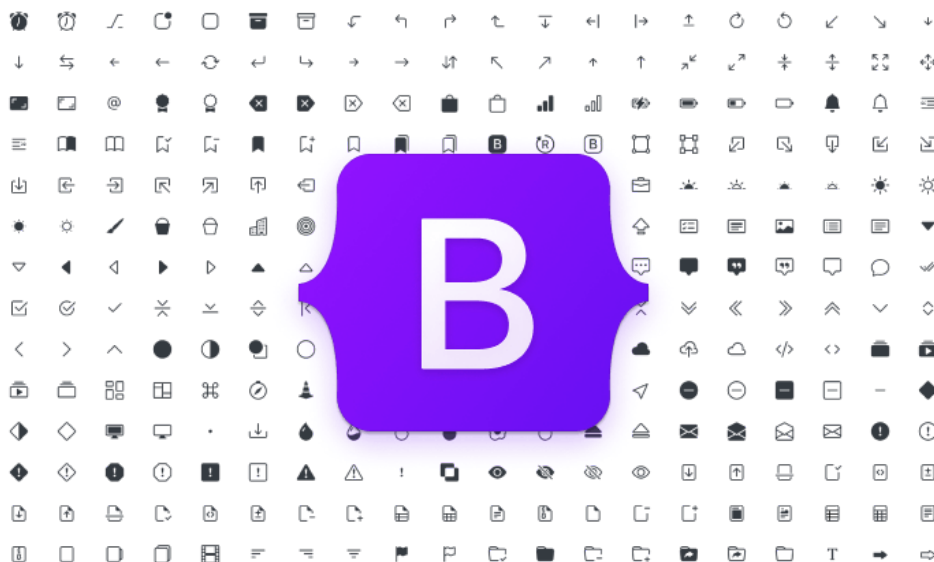
Bootstrap is een populair, open-source (gratis) HTML, CSS en Javascript Framework oftewel 'Front-end Framework' voor het bouwen van responsieve, mobile-first websites.

Bootstrap bestaat uit een groot aantal voorgedefinieerde CSS-basisstijlen en CSS-classes die we eenvoudig, mits het volgen van enkele richtlijnen, op onze HTML-code kunnen toepassen. Daarnaast bevat het ook Javascript-plugins. Met Javascript kunnen we dynamische elementen aan webpagina's toevoegen.

De webtaal Javascript zien we pas in het opleidingsonderdeel 'Web Advanced', dus voorlopig gaan we ons enkel richten op de basisfunctionaliteiten van Bootstrap. In deze cursus gaan we dieper in op:

- de modernere standaardopmaak na het toevoegen van het Bootstrap-framework;
- het positioneren van structuurelementen via het Bootstrap gridsysteem.

We gaan in deze cursus niet in op het gebruiken en aanpassen van Javascript-plugins en het gebruiken van Bootstrap-componenten. Wens je hier toch meer informatie over, kan je altijd terecht op de officiële website van Bootstrap: <https://getbootstrap.com/>.



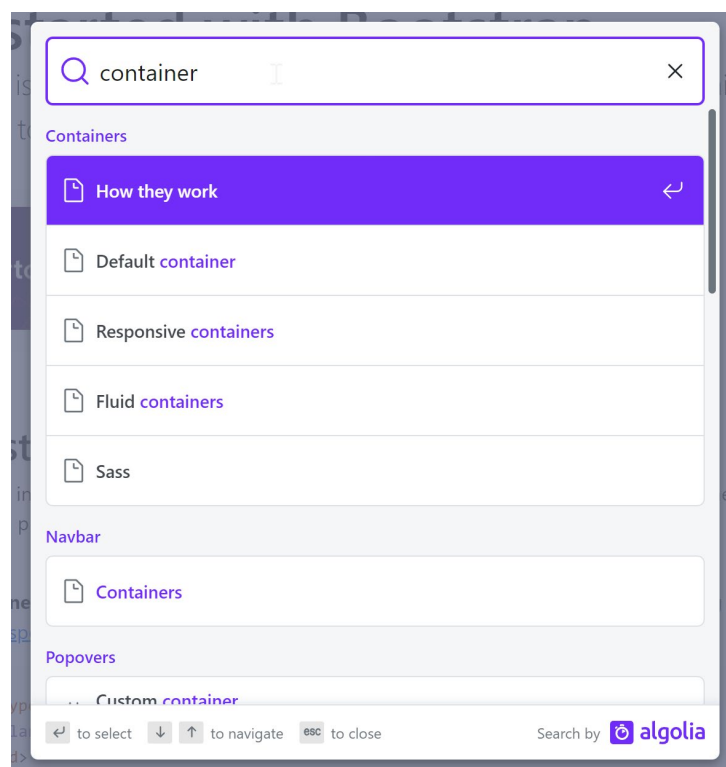
Bootstrap handleiding

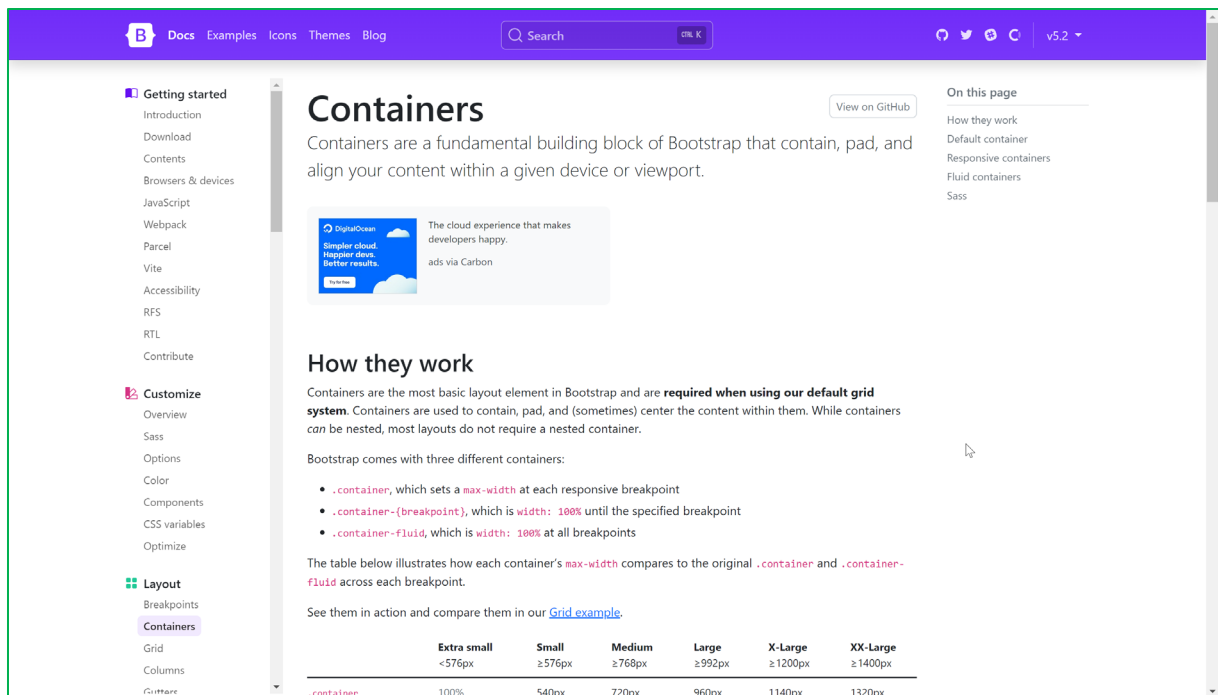
Het Bootstrap-framework bestaat uit meerdere CSS- en JS-bestanden. Deze bestanden kunnen we downloaden via de officiële Bootstrap website en toevoegen aan ons project in Webstorm.

Het basis CSS-bestand van het Bootstrap framework bevat meer dan tienduizend regels CSS-code met diverse CSS-classes die we kunnen toevoegen aan onze HTML-elementen op onze webpagina. Wanneer we Bootstrap gaan gebruiken, is het niet de bedoeling dat we alle aangeboden CSS-code van buiten leren. Zoals bij alle webtalen is vooral het ‘begrijpen’ en het ‘kunnen toepassen’ van de code belangrijk!

Op de website van Bootstrap vind je daarom een volledige online handleiding met een handige zoekfunctie terug. Deze handleiding sla je best op in je favorieten en open je tijdens het werken met Bootstrap.

<https://getbootstrap.com/docs/5.2/getting-started/introduction/>





Starten met Bootstrap

Bootstrap bestanden downloaden

Alvorens met het Bootstrap framework aan de slag te gaan, moeten we eerst de nodige bestanden van het framework downloaden en toevoegen aan ons Webstorm-project.

1. Ga naar de website van Bootstrap en download de laatste versie.
2. Ga naar de Downloadmap op je computer en unzip de Bootstrap-bestanden.
3. Open de map 'CSS'.
4. Kopieer de bestanden 'bootstrap-grid.min.css' en 'bootstrap-reboot.min.css' (of 'bootstrap.min.css' indien je het volledige framework wenst te gebruiken) naar je WebStorm-project op je computer.
5. Open je project in WebStorm.
6. Open de HTML-bestanden waaraan je Bootstrap wenst toe te voegen.
7. Voeg de linken naar de Bootstrap CSS-bestanden toe in het <head>-element.
8. Voeg je eigen CSS-bestand en de meta-gegevens van de viewport toe in het <head>-element.

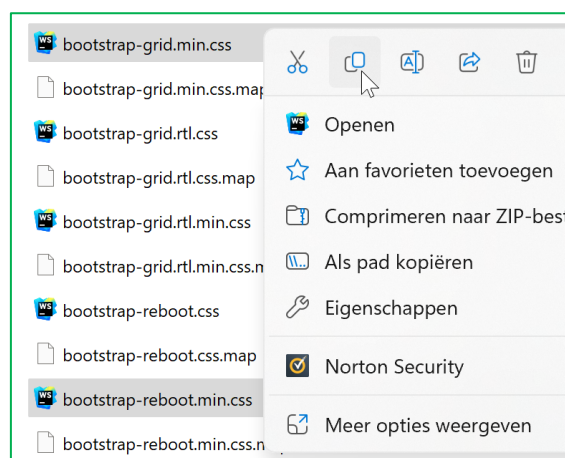
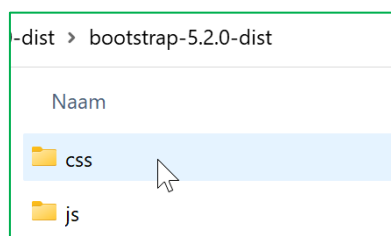
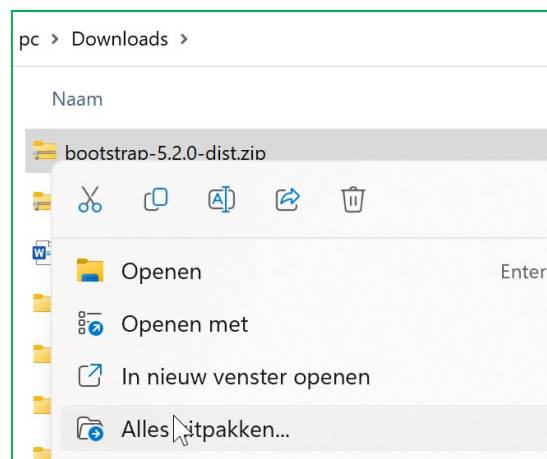
Compiled CSS and JS

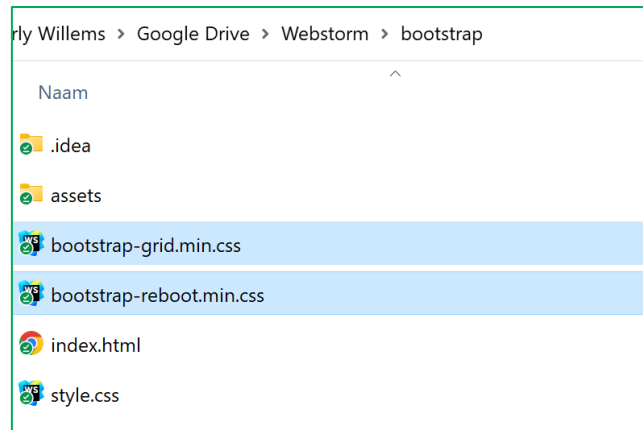
Download ready-to-use compiled code for **Bootstrap v5.2.0** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins (see [JS files comparison](#))

This doesn't include documentation, source files, or any optional JavaScript dependencies like Popper.

Download





```
<meta charset="UTF-8">
<title>Startpagina</title>
<link rel="stylesheet" href="bootstrap-reboot.min.css">
<link rel="stylesheet" href="bootstrap-grid.min.css">
<link rel="stylesheet" href="style.css">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

OPGELET! De volgorde van de CSS-bestanden is enorm belangrijk. Ons eigen CSS-bestand moet als laatste CSS-bestand staan.

Bootstrap bestanden insluiten

Wensen we de Bootstrap-bestanden liever niet te downloaden, dan kunnen we ook de CDN-manier (Content Delivery Network) van insluiten gebruiken. We plaatsen dan onderstaande code in het <head>-element van de webpagina.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/
bootstrap.min.css" rel="stylesheet" crossorigin="anonymous">
<link rel="stylesheet" href="style.css">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Standaardopmaak (reboot) van Bootstrap

De 'reboot' is een ingebouwde collectie van opmaak voor standaard HTML-elementen. Een groot deel van de bekende HTML-elementen hebben via de tag-selector in het CSS-bestand van het Bootstrap framework een nieuwe, modernere opmaak gekregen.

Via de reboot zorgt Bootstrap ervoor dat we een goed ogende, moderne website kunnen maken die er in iedere browser identiek uitziet. De reboot heeft invloed op de standaardopmaak van:

- De globale HTML-elementen en instellingen
- De hoofdingen
- De paragrafen
- De lijsten
- De afbeeldingen
- ...

Willen we tijdens het ontwikkelen van de website afwijken van de reboot-opmaak en aan de HTML-elementen huisstijl-specifieke opmaak geven, dan kunnen we dat via het tweede (lege) CSS-bestand 'style.css' dat we hebben toegevoegd aan onze HTML-code.

Dit is een hoofding van niveau 1

Dit is een hoofding van niveau 2

Dit is een hoofding van niveau 3

Dit is een hoofding van niveau 4

Dit is een hoofding van niveau 5

Dit is een hoofding van niveau 6

Dit is een stukje paragraaftekst met lorem ipsum dolor sit amet, consectetur adipisicing elit. Ab at aut, consequatur exercitationem illo iusto nostrum omnis quia! Aliquid aperiam dolorem, eius illo illum laudantium maxime nulla provident unde voluptas.

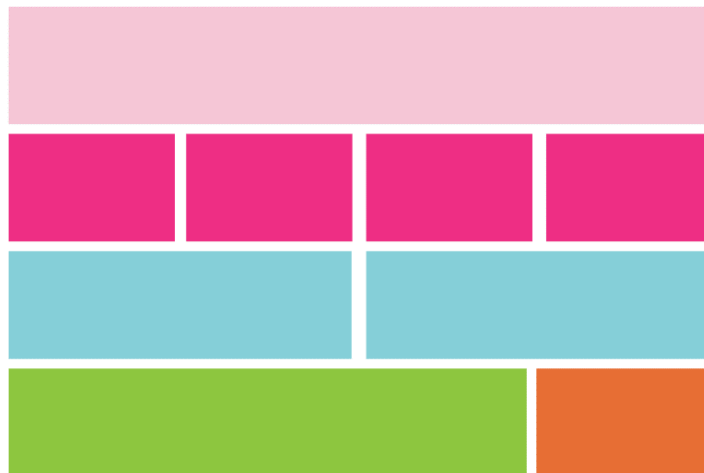
- Item 1
- Item 2
- Item 3

1. Item 1
2. Item 2
3. Item 3

Het Bootstrap gridsysteem

Het basis Bootstrap gridsysteem is een responsief Flexbox gridsysteem en bestaat uit containers, rijen en kolommen:

- Het positioneren van structuurelementen start met containers.
- In de containers staan de rijen.
- In de rijen staan de kolommen.
- In de kolommen staat de inhoud van de webpagina (of nieuwe rijen genest).



Containers

Containers zijn een enorm belangrijk onderdeel van het gridsysteem. Er bestaan drie soorten container-classes die we kunnen toevoegen aan een structuurelement:

- *container-fluid*: deze container is een full-width container en neemt de volledige breedte van de viewport in, ongeacht de schermresolutie.
- *container*: deze container is een responsief fixed-width container. Bij deze container is de maximale breedte afhankelijk van de schermresolutie.
- *container-{sm|md|lg|x|xxl}*: deze containers zijn responsieve containers met een afwijkende maximale ingestelde breedte afhankelijk van de schermresolutie.

	XS	SM	MD	LG	XL	XXL
Breekpunt	< 576px	≥ 576px	≥ 768px	≥ 992px	≥ 1200px	≥ 1400px
Class/max-width						
container-fluid	100%	100%	100%	100%	100%	100%
container	100%	540px	720px	960px	1140px	1320px
container-sm	100%	540px	720px	960px	1140px	1320px
container-md	100%	100%	720px	960px	1140px	1320px
container-lg	100%	100%	100%	960px	1140px	1320px
container-xl	100%	100%	100%	100%	1140px	1320px
container-xxl	100%	100%	100%	100%	100%	1320px

```

<body>
  <div class="container">

    </div>
</body>

```

Rijen

Rijen staan binnen containers. In één rij kunnen we tot maximaal 12 kolommen plaatsen. Het is niet verplicht dat we in een rij alle 12 beschikbare plaatsen gebruiken.

```

<body>
  <div class="container">
    <div class="row">

      </div>
    </div>
  </body>

```


Kolommen

De kolommen bevatten de inhoud van de webpagina. De kolommen zijn responsief en passen zich, op basis van de schermresolutie en het gekozen breekpunt, automatisch aan.

In één rij kunnen we maximaal 12 kolommen plaatsen. Gaan we over het maximum van 12 kolommen, worden de kolommen die te veel zijn, automatisch op een volgende regel binnen de rij geplaatst.

Elke kolom heeft links en rechts standaard 1.5 rem padding om afstand te creëren. Deze padding zorgt, ter compensatie, voor een negatieve margin op de rijen. Dit noemen we de gutter.

COL-3				COL-3				COL-3				COL-3					
COL-4						COL-4						COL-4					
COL-6								COL-6									
COL-2			COL-2			COL-2			COL-2			COL-2			COL-2		
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-1	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 1	col 1	col 1
col 4				col 4				col 4			
col 4				col 8							
col 6						col 6					
col 12											

Er bestaan twee soorten kolom-classes die we kunnen toevoegen aan een structuurelement:

- `col-{sm|md|lg|x|xxl}-{*}`: bij deze class moeten we eerst de soort responsiviteit bepalen en via het sterretje de inname van het aantal kolommen. Laten we het aantal kolommen weg, dan bepaald Bootstrap zelf de inname van het aantal kolommen. Wensen we de breedte van een kolom af te laten hangen van de inhoud van de kolommen, dan kunnen we bij het tweede sterretje ook 'auto' gebruiken als waarde.
- `col-{*}`: in deze class is de soort responsiviteit automatisch 'xs'. Het sterretje kunnen we optioneel invullen voor de inname van het aantal kolommen.

De soort responsiviteit van de kolommen kunnen we in onderstaande tabel nagaan.

	XS	SM	MD	LG	XL	XXL
Breekpunt	< 576px	≥ 576px	≥ 768px	≥ 992px	≥ 1200px	≥ 1400px
Classes	col-*	col-sm-*	col-md-*	col-lg-*	col-xl-*	col-xxl-*
Gedrag	Horizontaal naast elkaar	Gestapeld bij resoluties kleiner dan het breekpunt, horizontaal langs elkaar bij resoluties boven het breekpunt.				

```

<body>
  <div class="container">
    <!-- Rij met 3 kolommen van responsiviteit md en breedte 33,33% -->
    <div class="row">
      <div class="col-md-4">
        Dit is de inhoud van de eerste kolom.
      </div>
      <div class="col-md-4">
        Dit is de inhoud van de tweede kolom.
      </div>
      <div class="col-md-4">
        Dit is de inhoud van de derde kolom.
      </div>
    </div>
  </div>

  <div class="container">
    <!--Rij met 2 kolommen van responsiviteit xs en breedte 50% -->
    <div class="row">
      <div class="col">
        Dit is de inhoud van de eerste kolom.
      </div>
      <div class="col">
        Dit is de inhoud van de tweede kolom.
      </div>
    </div>
  </div>
</body>

```

Het is mogelijk om meerdere kolom-classes toe te voegen aan één structuurelement om de responsiviteit te verhogen. Dit kunnen we doen door een spatie tussen de twee classes te plaatsen in de HTML-code (speciale selector - gecombineerde klassen).

```

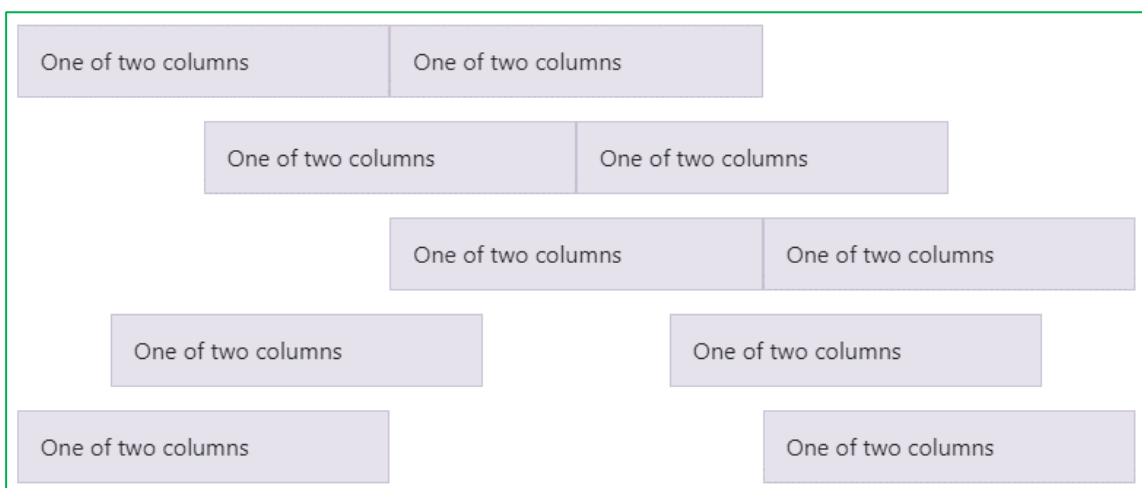
<body>
  <div class="container">
    <div class="row">
      <!--De kolommen zijn op smartphonescherf 50% breed en worden
      33,33% breed op een computer -->
      <div class="col-6 col-md-4">
        Dit is de inhoud van de eerste kolom.
      </div>
      <div class="col-6 col-md-4">
        Dit is de inhoud van de tweede kolom.
      </div>
      <div class="col-6 col-md-4">
        Dit is de inhoud van de derde kolom.
      </div>
    </div>
  </div>
</body>

```

Kolommen horizontaal uitlijnen

Indien we in een rij niet het maximum aantal kolommen hebben gebruikt, kan het handig zijn om deze kolommen horizontaal uit te lijnen binnen de rij. Hiervoor bestaan verschillende classes die we aan de rij kunnen toevoegen:

- justify-content-start
- justify-content-center
- justify-content-end
- justify-content-around
- justify-content-between
- justify-content-evenly



```
<div class="container">
  <div class="row justify-content-start">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-center">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-end">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-around">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
      One of two columns
    </div>
  </div>
  <div class="row justify-content-between">
    <div class="col-4">
      One of two columns
    </div>
    <div class="col-4">
```

```
    One of two columns  
  </div>  
</div>  
</div>
```

Kolommen verticaal uitlijnen

We kunnen, net zoals bij flexbox, verticale uitlijning instellen op een rij en op een kolom. De verticale uitlijning die we instellen op een rij zorgt ervoor dat deze van toepassing is op alle kolommen in die rij. De verticale uitlijning die we instellen op één specifieke kolom is enkel van toepassing op deze kolom.

Classes om verticale uitlijning in te stellen op een rij (in volgorde van weergave):

- align-items-start
- align-items-center
- align-items-end

One of three columns	One of three columns	One of three columns
One of three columns	One of three columns	One of three columns
One of three columns	One of three columns	One of three columns

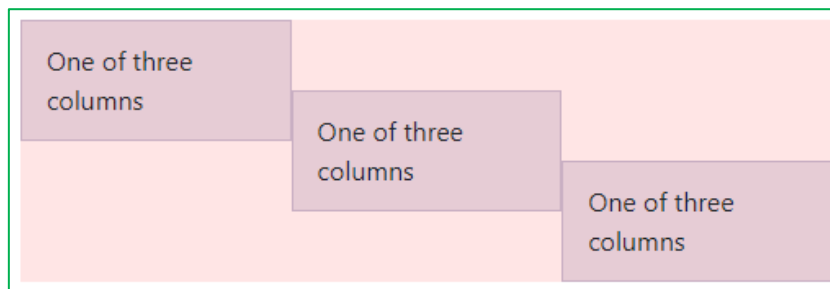
```

<div class="row align-items-center">
  <div class="col">
    <p>One of three columns</p>
  </div>
  <div class="col">
    <p>One of three columns</p>
  </div>
  <div class="col">
    <p>One of three columns</p>
  </div>
</div>

```

Classes om verticale uitlijning in te stellen op één specifieke kolom:

- align-self-start
- align-self-center
- align-self-end



```

<div class="row">
  <div class="col align-self-start">
    <p>One of three columns</p>
  </div>
  <div class="col align-self-center">
    <p>One of three columns</p>
  </div>
  <div class="col align-self-end">
    <p>One of three columns</p>
  </div>
</div>

```

Kolommen ordenen

Het is mogelijk om de volgorde van de kolommen aan te passen. Dit kunnen we doen door de class `.order-{volgnummer}` in te stellen op de kolommen.

Willen we de volgorde veranderen op een specifiek breekpunt, dan moeten we de soort responsiviteit meenemen via de class `.order-{sm|md|lg|x|xxl}-{volgnummer}`.

Dit is de inhoud van de derde kolom.	Dit is de inhoud van de eerste kolom.	Dit is de inhoud van de tweede kolom.
--------------------------------------	---------------------------------------	---------------------------------------

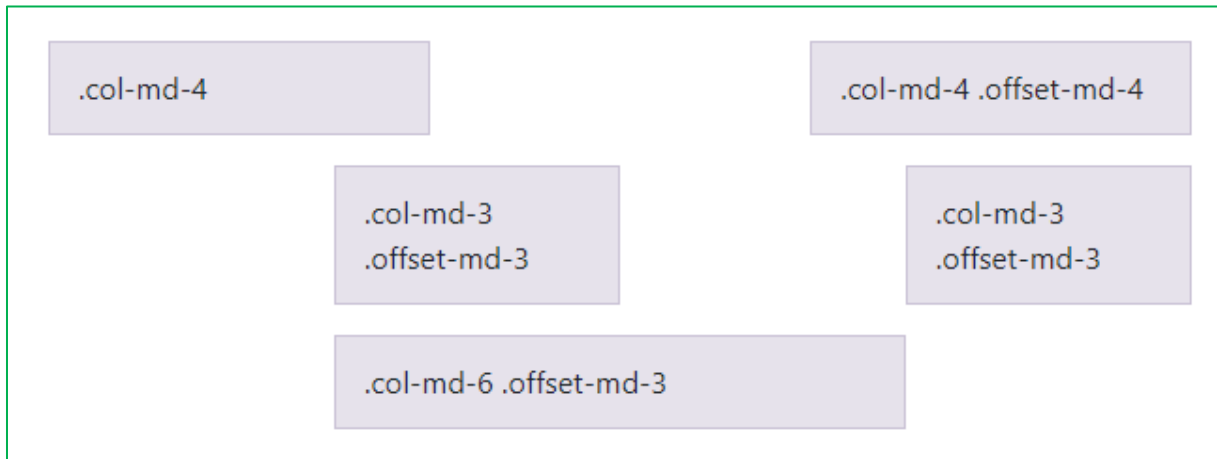
Dit is de inhoud van de eerste kolom.	Dit is de inhoud van de tweede kolom.
Dit is de inhoud van de derde kolom.	

```
<div class="row">
  <!--De kolommen zijn op smartphonescherf 50% breed en worden
       33,33% breed op een computer -->
  <div class="col-6 col-md-4 order-md-2">
    <p>Dit is de inhoud van de eerste kolom.</p>
  </div>
  <div class="col-6 col-md-4 order-md-3">
    <p>Dit is de inhoud van de tweede kolom.</p>
  </div>
  <div class="col-6 col-md-4 order-md-1">
    <p>Dit is de inhoud van de derde kolom.</p>
  </div>
</div>
```

Lege ruimtes in rijen opvullen met offset

Door een offset-class toe te voegen, kunnen we een specifieke kolom meer naar rechts plaatsen binnen een rij.

Aan de offset-class kunnen we, net zoals bij de verschillende col-classes, de soort responsiviteit en de inname van het aantal kolommen meegeven.



```
<div class="container">
  <div class="row">
    <div class="col-md-4">.col-md-4</div>
    <div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>
  </div>
  <div class="row">
    <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
    <div class="col-md-3 offset-md-3">.col-md-3 .offset-md-3</div>
  </div>
  <div class="row">
    <div class="col-md-6 offset-md-3">.col-md-6 .offset-md-3</div>
  </div>
</div>
```


Kolommen en rijen nesten

We kunnen een meer geavanceerde positionering maken door nieuwe rijen en kolommen in een kolom te nesten.

Dit is de inhoud van de eerste kolom.	Dit is de inhoud van de tweede kolom. met een opsplitsing in nog twee extra rijen met twee kolommen.	
	Dit is de inhoud van de eerste kolom op de eerste rij.	Dit is de inhoud van de tweede kolom op de eerste rij.
	Dit is de inhoud van de eerste kolom op de tweede rij.	Dit is de inhoud van de tweede kolom op de tweede rij.

```
<div class="row">
  <div class="col-6">
    Dit is de inhoud van de eerste kolom.
  </div>
  <div class="col-6">
    Dit is de inhoud van de tweede kolom. met een opsplitsing in
    nog twee extra rijen met twee kolommen.
    <div class="row">
      <div class="col-6">Dit is de inhoud van de eerste kolom op
      de eerste rij.</div>
      <div class="col-6">Dit is de inhoud van de tweede kolom op
      de eerste rij.</div>
    </div>
    <div class="row">
      <div class="col-6">Dit is de inhoud van de eerste kolom op
      de tweede rij.</div>
      <div class="col-6">Dit is de inhoud van de tweede kolom op
      de tweede rij.</div>
    </div>
  </div>
</div>
```

Padding en margin

Het Bootstrap framework bevat voorgedefinieerde classes om op een snelle en responsieve manier margin en padding te voorzien op de verschillende HTML-elementen van een webpagina.

Afhankelijk van de classes die we instellen, kunnen we een margin en padding verkrijgen dat varieert van 0rem tot 3rem. Willen we meer witruimte creëren, kunnen we nog altijd zelf margin en padding schrijven in ons eigen CSS-bestand.

Notatie

Willen we een spatiëring instellen die voor alle breekpunten geldig is, dan definiëren we de soort responsiviteit niet. De class wordt dan genoteerd als `.{eigenschap}{zijde}-{grootte}`.

Willen we de spatiëring instellen op een specifiek breekpunt, dan moeten we de soort responsiviteit wel meenemen in onze notatie. De class wordt dan genoteerd als `.{eigenschap}{zijde}-{sm|md|lg|xl|xxl}-{grootte}`.

We kunnen kiezen uit twee **eigenschappen**:

- m - om de margin (witruimte buiten het element) in te stellen.
- p - om de padding (witruimte binnen het element) in te stellen.

Daarna kunnen we de **zijdes** waarop we de margin of padding willen instellen kiezen:

- t - de bovenzijde, 'top'
- b - de onderzijde, 'bottom'
- s - de linkerzijde (of rechterzijde bij RTL-werking), 'start'
- e - de rechterzijde (of linkerzijde bij RTL-werking), 'end'
- x - de horizontale zijdes, de linker- en rechterzijde, 'left' en 'right'
- y - de verticale zijdes, de boven- en onderzijde, 'top' en 'bottom'
- geen zijde opgegeven – alle zijdes

Na de optionele responsiviteit (*sm|md|lg|xl|xxl*) volgt de **grootte**:

- 0 - reset naar waarde 0 (geen padding of margin)
- 1 - 0.25rem
- 2 - 0.50rem
- 3 – 1.00rem
- 4 - 1.50rem
- 5 – 3.00rem
- auto

```
<!-- Op alle zijdes 1.50rem padding -->
<h1 class="p-4">Hoofding niveau 1</h1>

<!-- Op de boven- en onderzijde 0.50rem padding -->
<h1 class="py-2">Hoofding niveau 1</h1>

<!-- Enkel aan de bovenzijde 1.00rem padding -->
<h1 class="pt-3">Hoofding niveau 1</h1>

<!-- Op geen enkele zijde margin (reset) -->
<h1 class="m-0">Hoofding niveau 1</h1>

<!-- Op de linker- en rechterzijde 3.00rem margin -->
<h1 class="mx-5">Hoofding niveau 1</h1>

<!-- Enkel aan de linkerzijde 0.25rem margin -->
<h1 class="ml-1">Hoofding niveau 1</h1>

<!--Het <div>-element staat gecentreerd in het midden -->
<div class="mx-auto">Gecentreerd element</div>
```

Hoofding niveau 1

Hoofding niveau 1

Hoofding niveau 1

Hoofding niveau 1

Hoofding niveau 1

Hoofding niveau 1

Gecentreerd element

Gutter

Elke kolom heeft links en rechts standaard 1.5 rem padding om afstand te creëren. Deze padding creëert, ter compensatie, een negatieve margin op de rijen. Dit noemen we gutter. Wensen we meer of minder gutter, dan kunnen we dit aanpassen met behulp van voorgedefinieerde classes.

Notatie

Willen we een gutter instellen die voor alle breekpunten geldig is, dan definiëren we de soort responsiviteit niet. De class wordt dan genoteerd als `.g{zijde}-{grootte}`.

Willen we de gutter instellen op een specifiek breekpunt, dan moeten we de soort responsiviteit wel meenemen in onze notatie. De class wordt dan genoteerd als `.g{zijde}-{sm|md|lg|x|xxl}-{grootte}`.

We kunnen volgende **zijdes** instellen:

- x - de horizontale gutter (start en einde)
- y - de verticale gutter (boven en onder)
- geen zijde opgegeven – alle zijdes

Na de optionele responsiviteit (`sm|md|lg|x|xxl`) volgt de instelling van de **grootte**:

- 0 - reset naar waarde 0 (geen gutter)
- 1 - 0.25rem
- 2 - 0.50rem
- 3 – 1.00rem
- 4 - 1.50rem
- 5 – 3.00rem

```

<!-- De volledige breedte benutten met drie afbeeldingen -->
<!-- Aan de start (links) en het einde (rechts) de padding verwijderen -->
<div class="container-fluid px-0">
    <!-- De horizontale gutter verwijderen -->
    <div class="row gx-0">
        <div class="col-md-4">
            
        </div>
        <div class="col-md-4">
            
        </div>
        <div class="col-md-4">
            
        </div>
    </div>
</div>

```

```

// Van de afbeelding een block-element maken en de volledige breedte geven
    binnen de kolom.
img {
    display: block;
    width: 100%;
}

```



Webformulieren

Formulieren verhogen de interactiviteit van een website. Ze worden niet enkel gebruikt voor het opgeven van contactgegevens, maar ook voor het invullen van vragenlijsten.

Bij het klassieke gebruik van een formulier stuurt de bezoeker van de website gegevens naar een webserver. De gegevens worden dan verwerkt door een webapplicatie via een achterliggend script. De verwerkte gegevens kunnen:

- in een databank worden bewaard
- in een respons aan de bezoeker worden getoond
- in een e-mail worden verzonden

Om ten volle van de meerwaarde van een formulier te genieten, moeten we deze doordacht opbouwen in onze code. We bedenken dus best op voorhand welke gegevens we aan de bezoekers willen vragen.

Formulier:

Naam:

Straat:

Postnummer: Gemeente:

Paswoord:

Land:

Geslacht:

☒ Mannelijk

☐ Vrouwelijk

Operating system:

☐ Windows

☐ Linux

Commentaar:

Het form-element

Formulieren kunnen zonder problemen in een HTML-pagina worden opgenomen. Alle formulierspecifieke elementen, moeten hierbij wel binnen een `<form>`-element staan.

```
<form>
    ...
</form>
```

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

Form-attributen

Method

Het method-attribuut geeft aan welke HTTP-methode (GET of POST) we wensen te gebruiken om de formuliergegevens te verzenden. Dit attribuut is verplicht.

GET	<p>De formuliergegevens worden toegevoegd aan het einde van de URL.</p> <p>Deze methode zorgt ervoor dat de browser de gegevens kan opslaan in het cachegeheugen en de bezoeker een bladwijzer kan maken naar de pagina met het ingevulde formulier.</p> <p>Deze methode gebruiken we best enkel voor kleine, eenvoudige formulieren waarbij beveiliging niet belangrijk is.</p>
POST	<p>De formuliergegevens worden naar een webserver gestuurd.</p> <p>Deze methode is ideaal voor (grotere) formulieren met (al dan niet) gevoelige of persoonlijke informatie.</p>

Action

Het action-attribuut geeft aan hoe we de formuliergegevens wensen te verwerken. In de meeste gevallen gaan we verwijzen naar een server-side script (PHP, ASP.NET, CGI, ...) voor de verwerking van onze formuliergegevens.

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

Om onze oefeningen in dit opleidingsonderdeel te testen, gaan we altijd bovenstaande action gebruiken. Deze action verwijst naar een simpel echo-script dat de verzonden formuliergegevens gewoon opnieuw gaat weergeven.

We kunnen ook een mailto-koppeling gebruiken om ons formulier te testen. Het formulier stuurt dan een mail naar een opgegeven e-mailadres via een mailclient. Dit is echter enorm verouderd, makkelijk te misbruiken en dus niet meer aan te raden om te gebruiken.

```
<form method="post" action="mailto:kimberly.willems@pxl.be">
```

Enctype

Het enctype-attribuut geeft aan hoe de formuliergegevens moeten worden gecodeerd bij het verzenden. Hierbij zijn er meerdere mogelijkheden, maar onderstaande mogelijkheden worden het meeste gebruikt:

- *application/x-www-form-urlencoded*: dit is de default-waarde.
- *multipart/form-data*: voor het verzenden van bestanden.
- *text/plain*: voor het verzenden van platte tekst.

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


Name

Het name-attribuut geeft een naam aan het formulier. Wanneer we meerdere formulieren op één webpagina plaatsen, moet de naam per formulier uniek zijn.

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

Novalidate

Het novalidate-attribuut geeft aan dat het formulier niet gevalideerd moet worden bij verzenden. Met dit attribuut wordt de standaard validatie van de browser niet getriggerd.

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

Target

Het target-attribuut geeft het doelvenster aan waarin we het antwoord wensen te ontvangen na het verzenden van de formuliergegevens.

_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.

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

Accept-charset

Het accept-charset attribuut geeft aan welke charsets er door de server waarnaar we het formulier versturen, aanvaard zullen worden.

```
<form method="post" action="https://jkorpela.fi/cgi-bin/echo.cgi"
  enctype="plain/text" name="klachtenformulier" target="_blank" novalidate
  accept-charset="utf-8">
```

Autocomplete

Het autocomplete-attribuut zorgt ervoor dat eerder ingevulde gegevens niet verloren gaan wanneer de bezoeker de webpagina verlaat.

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

Labels

Bij ieder invoerelement in een webformulier, kunnen we een label plaatsen met behulp van het <label>-element. Via het for-attribuut geven we mee op welk invoerelement het label van toepassing is. Het for-attribuut linkt met het id-attribuut van het bijhorende invoerelement.

```
<label for="voornaam">Voornaam</label>
```

Invoerelementen

Input-elementen

De input-elementen vormen de basisbestandsdelen van een formulier. In een input-element definiëren we verplicht het type invoer via het type-attribuut. Mogelijke types zijn:

- Tekstvelden: text, email, url, password, hidden
- Datumvelden: date, month, week, time, datetime, datetime-local
- Nummervelden: number, range
- Keuzelijsten: checkbox, radio
- Bestandsvelden: file
- Knoppen: submit, reset, image en button.
- ...

Wanneer bepaalde invoerelementen niet worden ondersteund door de browser, werken ze als een invoerveld van het type tekst.

Volgende attributen zijn, afhankelijk van het gekozen type input-element, toepasbaar:

- *Id*: geeft een id aan het input-element.
- *Placeholder*: toont een invoerhint.
- *Minlength*: geeft het minimaal aantal tekens aan.
- *Min*: zorgt voor een ondergrens (minimale waarde).
- *Maxlength*: geeft het maximale aantal tekens aan.
- *Max*: zorgt voor een bovengrens (maximale waarde).
- *Step*: geeft de verhoging van een waarde aan in stappen.
- *Required*: maakt het input-element verplicht om aan te vinken of in te vullen.
- *Disabled*: zorgt dat we het input-element niet kunnen aanklikken/aanvinken/invullen.
- *List*: zorgt voor een koppeling naar een datalist.
- *Name*: geeft een naam aan het input-element.
- *Pattern*: zorgt dat de invoer volgens een reguliere expressie moet verlopen.
- *Value*: specificeert de (begin)waarde.
- ...

Input type text

Input-elementen van het type 'text' zijn het meest geschikt voor het verzamelen van algemene informatie. Het is het eenvoudigste veld om als bezoeker tekstuele of numerieke informatie door te geven.

```
<label for="voornaam">Voornaam</label>
<input type="text" name="voornaam" id="voornaam" />
```

Voornaam

Met HTML5 kunnen we vrijblijvende opties toevoegen aan ons tekst inputveld. Dit doen we door gebruik te maken van een `<datalist>`-element. De bezoeker kan dan kiezen uit een optie of zelf een waarde ingeven.

```
<label for="gemeente">Gemeente</label>
<input list="gemeentelijst" name="gemeente" id="gemeente" />
<datalist id="gemeentelijst">
  <option label="3500" value="Hasselt"></option>
  <option label="3600" value="Genk"></option>
  <option label="3700" value="Sint-truiden"></option>
</datalist>
```

Gemeente

- Hasselt
3500
- Genk
3600
- Sint-truiden
3700

Input type password

Input-elementen van het type 'password' werken analoog aan input-elementen van het type 'text'. Het enige verschil bestaat erin dat er in het wachtwoordveld bolletjes verschijnen bij het intypen van een waarde.

```
<label for="wachtwoord">Wachtwoord</label>  
<input type="password" name="wachtwoord" id="wachtwoord" />
```

Paswoord

Input type hidden

Input-elementen van het type 'hidden' zijn onzichtbaar voor de bezoekers van de webpagina. Deze velden zijn vooral handig bij het gebruik van achterliggende scripts.

```
<input type="hidden" name="scriptwaarde" id="scriptwaarde" />
```

Input type email

Een input-element van het type 'email' is bedoeld om een e-mailadres in te geven. Dit veld controleert automatisch op de aanwezigheid van de at (@) en de punten (.) in een e-mailadres.

```
<label for="mailadres">Mailadres</label>  
<input type="email" name="mailadres" id="mailadres" />
```

E-mailadres

Input type number

Een input-element van het type 'number' laat toe een getal in te geven. Met de attributen 'min' en 'max' kunnen we de range van het getal bepalen.

```
<label for="getal">Random getal tussen 0 en 10</label>  
<input type="number" name="getal" id="getal" min="0" max="10" />
```

Getal

Input type URL

Een input-element van het type 'url' laat toe een pad naar een andere website of bestand in te geven. Dit veld controleert automatisch op de vormgeving van de URL.

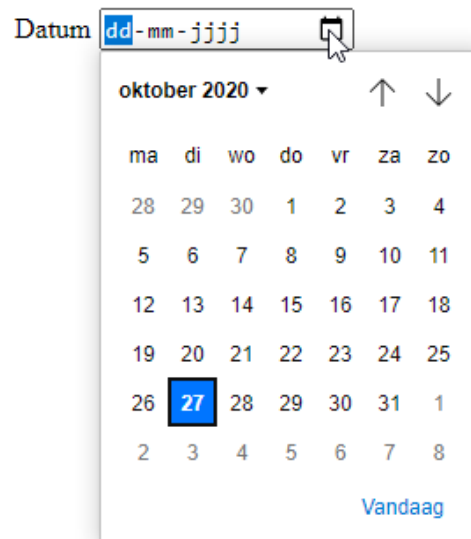
```
<label for="siteadres">URL</label>  
<input type="url" name="siteadres" id="siteadres" />
```

Website URL

Input type date

Een input-element van het type 'date' kan gebruikt worden om een datum te kiezen via een datumkiezer.

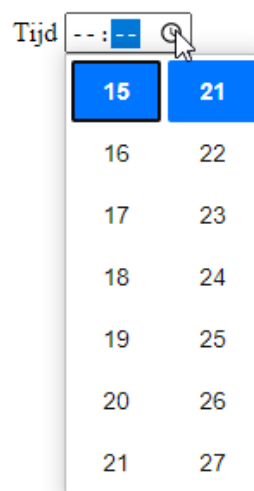
```
<label for="datum">Datum</label>  
<input type="date" name="datum" id="datum" />
```



Input type time

Een input-element van het type 'time' kan gebruikt worden om een tijd te kiezen via een tijdskiezer.

```
<label for="tijdstip">Tijd</label>
<input type="time" name="tijdstip" id="tijdstip" />
```



Input type range

Een input-element van het type 'range' kan gebruikt worden om een range in te geven. Met de attributen 'min' en 'max' kan het minimum en het maximum van het veld worden bepaald.

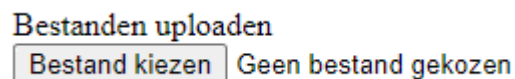
```
<label for="range">Range</label>
<input type="range" name="range" id="range" min="0" max="5" />
```



Input type file

Wanneer bestanden dienen verzonden te worden naar de webserver, kan de bezoeker gebruik maken van het input-element van het type 'file' om een bestand te uploaden.

```
<label for="bijvoegsel">Bestanden uploaden</label>
<input type="file" name="bijvoegsel" id="bijvoegsel" />
```



Input type checkbox

Wanneer de bezoeker een keuze dient te maken uit één of meerdere mogelijkheden maken we gebruik van selectievakken. Dit zijn input-elementen van het type 'checkbox'. Het attribuut 'checked' wordt gebruikt om een vakje standaard aan te vinken.


```
<input type="checkbox" name="windows" id="windows" value="Windows 10" />
<label for="windows">Windows 10</label>
<input type="checkbox" name="mac" id="mac" value="Mac OS" checked />
<label for="mac">Mac OS</label>
```

☐ Windows 10
☐ Mac OS

☒ Windows 10
☐ Mac OS

Input type radio

Radiobuttons zijn input-elementen van het type 'radio'. Radiobuttons werken analoog aan de selectievakken, maar met het verschil dat de bezoeker slechts één van de mogelijkheden kan selecteren. Dit bereiken we door alle keuzerondjes dezelfde 'name' te geven. Het attribuut 'value' zorgt voor het onderscheiden van de verschillende waardes.

```
<input type="radio" name="besturing" id="windows" value="windows" />
<label for="windows">Windows 10</label>
<input type="radio" name="besturing" id="macos" value="macos" />
<label for="macos">macOS</label>
<input type="radio" name="besturing" id="chromeos" value="chromeos" />
<label for="chromeos">Chrome OS</label>
```

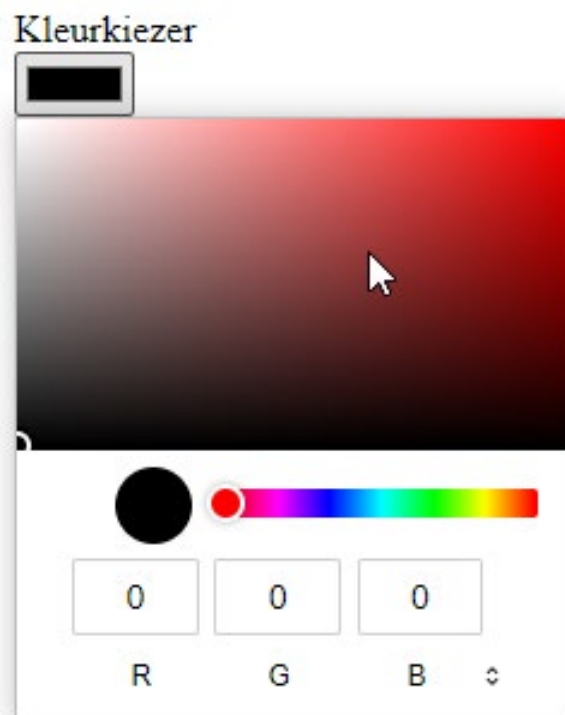
☐ Windows 10
☐ Mac OS
☐ Chrome OS

☒ Windows 10
☐ Mac OS
☐ Chrome OS

Input type color

Input-elementen van het type 'color' laten de bezoeker toe om een kleur te kiezen of een kleurwaarde in te geven.

```
<label for="kleur">Kies een kleur</label>  
<input type="color" name="kleur" id="kleur" />
```



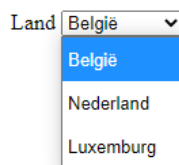
Select-elementen

Via een select-element met daarin geneste optie-elementen kunnen we een keuzelijst maken. Een keuzelijst stelt een bezoeker in staat om één of meerdere opties te selecteren uit een lijst.

Op select-elementen kunnen we volgende attributen toepassen:

- *Disabled*: voorkomt dat een optie gekozen kan worden of dat de optie wordt verzonden naar de server.
- *Multiple*: zorgt ervoor dat de gebruiker meerdere items tegelijkertijd kan selecteren.
- *Name*: geeft een naam aan de keuzelijst.
- *Id*: geeft een id aan de keuzelijst.
- ...

```
<label for="land">Land</label>
<select name="land" id="land">
  <option selected>België</option>
  <option>Nederland</option>
  <option>Luxemburg</option>
</select>
```



Tekstvak met meerdere regels

Voor het versturen van lange teksten, maken we best gebruik van een textarea-element.

Via het attribuut 'wrap' kunnen we het toevoegen van lijneinden bij het verzenden regelen. Bij 'soft' wrapping wordt er geen nieuwe regel meegezonden als deze er is in het tekstvak, bij 'hard' wrapping wordt er wel een combinatie van linefeed en carriage return meegezonden.

Met de CSS-stijleigenschap 'white-space' kan het uitzicht in het tekstvak worden aangepast:

- *Normal*: dit is de standaardinstelling, de tekst breekt vanzelf af en begeeft zich op de volgende regel als het eind van het tekstvak bereikt wordt.
- *Pre*: de tekst komt tot aan het einde en gaat dan naar een volgende regel.
- *Nowrap*: de tekst breekt niet af. Het tekstvak krijgt een schuifbalk als de bezoeker blijft doortypen zonder een enter in te geven.

```
<label for="commentaar">Commentaar</label>
<textarea id="commentaar" wrap="hard" name="commentaar">
  Deze tekst staat klaar in het tekstvak
</textarea>
```

Commentaar

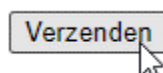
Deze tekst staat klaar in het tekstvak

Knoppen

Verzendknop

De waarden die de bezoekers ingeven, moeten naar de server worden verstuurd. Hiervoor moeten we gebruik maken van een submit-knop. Een submit-knop is een input-element van het type 'submit'. Via het attribuut 'value' kunnen we de nodige tekst op de knop plaatsen.

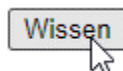
```
<input type="submit" value="Verzenden" />
```



Wisknop

Om alle invoervelden in het formulier terug leeg te maken, kunnen we gebruik maken van een reset-knop. Een reset-knop is een input-element van het type 'reset'. Via het attribuut 'value' kunnen we de nodige tekst op de knop plaatsen.

```
<input type="reset" value="Alle gegevens wissen" />
```



Knop zonder voorgedefinieerde werking

Via het button-element of via het input-element van het type 'button' kunnen we een knop zonder voorgedefinieerde werking in het formulier plaatsen.

Deze knop kunnen we via een event handler in Javascript een actie geven. Dit zien we in het opleidingsonderdeel 'Web Advanced'.

```
<button>Klik op mij</button>  
<input type="button" value="Klik op mij" />
```

Structuur toevoegen

Soms kan het handig zijn om invoerelementen die samenhangen te groeperen. Dit kunnen we doen door een fieldset-element rond deze invoerelementen te plaatsen.

Via het legend-element kunnen we de groep ook een titel geven.

```

<fieldset>
  <legend>Persoonlijke gegevens</legend>
  <label for="voornaam">Voornaam</label>
  <input type="text" id="voornaam" name="voornaam">
  <label for="naam">Familienaam</label>
  <input type="text" id="naam" name="naam">
</fieldset>

```

Persoonlijke gegevens

Voornaam
 Familienaam

Toegankelijkheid verhogen

Ook mensen met een beperking surfen op het internet. Om ervoor te zorgen dat ons formulier ook toegankelijk is voor hen, houden we ons best aan enkele regels en afspraken:

- Zorg voor een correcte tab-volgorde in het formulier. Via de tabtoets kunnen bezoekers eenvoudig navigeren door een formulier. Standaard is de tab-volgorde van links naar rechts en van boven naar onder. Pas indien nodig de tab-volgorde aan door het attribuut 'tabindex' toe te voegen aan de labels en invoerelementen.
- Koppel ieder label aan het bijhorende invoerelement door gebruik te maken van het for-attribuut bij het label en het id-attribuut bij het invoerelement.
- Voeg aan de invoerelementen zonder label het attribuut 'title' toe. Dit attribuut zorgt ervoor dat het invoerelement een tooltip krijgt. De tekst in de tooltip is de waarde van het ingestelde attribuut.
- Plaats bij datumvelden ook de reguliere expressie (bvb. dd/mm/jjjj) mee in het label.
- Groepeer samenhangende invoerelementen in een fieldset met een legend.
- Voorzie een submit-knop met een duidelijke tekstboodschap op de knop.

Extra

Meta-tags en meta-gegevens

Meta-tags zijn elementen die specifieke informatie bevatten over de website en/of de webpagina. Meta-tags staan altijd in het head-element van een webpagina.

Meta-tags zijn optioneel, maar aangeraden om toe te voegen. De informatie in de meta-tags kan ervoor zorgen dat de website geoptimaliseerd wordt voor diverse browsers, zoekmachines, sociale media platformen en andere systemen.

Meta-tag met attribuut charset

Via de meta-tag met het attribuut charset kunnen we aangeven dat onze website gebruik maakt van de UTF-8 karakterset. Iemand die surft naar onze website met een andere standaard karakterset, krijgt hierdoor de website toch te zien in de door ons opgegeven karakterset.

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

Meta-tag met attributen name en content

Meta-tags hebben vaak de attributen 'name' en 'content':

- Name: het type waarde (auteur, applicatienaam, omschrijving, sleutelwoorden, ...).
- Content: de eigenlijke waarde van het aangegeven type waarde.

```
<meta name="author" content="Kimberly Willems">

<meta name="application-name" content="Web essentials website">

<meta name="description" content="Een website voor het opleidingsonderdeel
Web Essentials. Deze website bevat diverse eenvoudige voorbeelden over het
gebruik van HTML en CSS-code.">

<meta name="keywords" content="HTML, CSS, voorbeeld, eenvoudig,
opleidingsonderdeel, web essentials, meta-tags, positionering, tabellen,
formulieren, flexbox, bootstrap, bootstrap grid">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="robots" content="noindex">
```

Meta-tags via Open Graph protocol

Meta-tags spelen ook een belangrijke rol wanneer we website wensen te delen via sociale media. Zo kunnen we via meta-tags vastleggen welke titel, omschrijving en afbeelding getoond moet worden bij het delen van de webpagina.

Hiervoor maken we best gebruik van het Open Graph Protocol, dit protocol zorgt ervoor dat alle meta-gegevens op een open standaard manier gelezen kunnen worden en dat er extra gegevens toegevoegd kunnen worden bovenop de standaard meta-gegevens.

```
<meta property="og:title" content="Introductie in meta-tags">

<meta property="og:type" content="article" />

<meta property="og:description" content="Op deze webpagina vinden jullie
verschillende voorbeelden van meta-tags volgens het open graph protocol.">

<meta property="og:image" content="http://www.webessentials.be/thumb.jpg">
```



```
<meta property="og:url" content="http://www.webessentials.be/ogp.html">

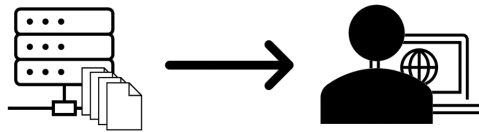
<meta property="og:site_name" content="Web essentials website">

<meta property="og:updated_time" content="1570213141">
```

Website online plaatsen

Webhosting

Een website bestaat uit één of meerdere bestanden. Deze bestanden kunnen we op de harde schijf van een server plaatsen. Een server is een krachtige computer die continu (24/24 en 7/7) in verbinding staat met het internet en die de bestanden raadpleegbaar maakt.



Wanneer we webhosting aankopen, huren we, voor een bepaalde tijd, een stukje ruimte op een server. Bij de aankoop van webhosting moeten we letten op:

- De betrouwbaarheid van de hostingprovider
- De benodigde hoeveelheid webruimte
- De hardware-specificaties van de server
- De uptime-garantie
- De mogelijkheden wat betreft beveiliging
- De mogelijkheden wat betreft (automatische) backups
- De toegelaten hoeveelheid dataverkeer
- De bereikbaarheid van de klantenservice
- De ondersteuning van de recentste versies van PHP en MySQL
- ...

Hostingproviders

Een hostingprovider is een bedrijf dat webdiensten of faciliteiten voor het hosten van een website aanbiedt. Deze bedrijven hebben vaak grote datacenters met krachtige servers.

Voorbeelden van enkele populaire hostingproviders:

- Neostada (<https://www.neostada.nl/>)
- Vimexx (<https://www.vimexx.nl/>)
- One.com (<https://www.one.com/nl/>)
- Combell (<https://www.combell.com/nl/>)
- Versio (<https://www.versio.nl/>)
- GoDaddy (<https://be.godaddy.com/>)
- ...

Domeinnaam

Een domeinnaam is een unieke naam die je voor jezelf, je bedrijf of organisatie vastlegt. De domeinnaam geeft je een identiteit op het internet en is een gemakkelijke manier om door andere gevonden te worden.

Een domeinnaam bestaat uit minimaal twee delen, gescheiden door een punt:

- een naam;
- een suffix of Top Level Domain (TLD).

De naam
WWW . VOORBEELD . BE
TLD

FTP

FTP staat voor File 'Transfer Protocol'. FTP is een netwerkprotocol dat de uitwisseling van bestanden tussen computers vergemakkelijkt door een aantal handelingen te standaardiseren. FTP maakt het mogelijk bestanden te verzenden (uploaden) en te ontvangen (downloaden) via het internet.

We kunnen een verbinding met een FTP-server maken via een webbrowser of een FTP-programma, ook wel FTP-client genoemd. De twee meest bekende FTP-clients zijn:

- FileZilla
- Cyberduck

