

Van start met HTML en CSS

Web Frontend Basics

1	VAN START MET HTML EN CSS	3
1.1	Je eerste webpagina maken	3
1.1.1	VS Code	3
	Stap 1	3
	Stap 2	3
	Stap 3	3
	Stap 4	3
	Stap 5	4
	Stap 6	4
1.1.2	HTML Code onder de loep	5
1.1.1.1	Structuur van onze pagina	5
1.1.3	Commentaar in onze code	7
1.1.4	Onze eerste elementen	8
1.1.1.2	Hoofdingen h1 ... h6	8
1.1.1.3	Alinea – P	8
1.1.1.4	Toepassing	9
1.1.1.5	Basiselementen span en div	11
1.2	Het CSS Box Model	12
1.2.1	Wat is het CSS Box Model?	12
1.1.1.6	Vakken in het CSS Box Model	13
1.2.2	Inline en Block elementen	13
1.3	Onze eerste Cascading Style Sheet	14
	Stap 1	14
	Stap 2	14
	Stap 3	14
	Stap 4	15
	Stap 5	15
1.3.1	CSS Selector op Element	16

2 VAN START MET HTML EN CSS

2.1 JE EERSTE WEBPAGINA MAKEN

2.1.1 VS CODE

STAP 1

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

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

STAP 2

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

Manier

1

Rechtsklik op de folder die je net aanmaakte en klik op **Open with Code**. Zoals hiernaast te zien is. Indien je deze optie niet hebt dan ben je bij de installatie van VS Code deze setting vergeten aanvinken. Niet erg, je kan dan ook gewoon met de tweede manier aan de slag.

Openen

Aan Snelle toegang vastmaken

Open in Visual Studio

Git GUI Here

Git Bash Here

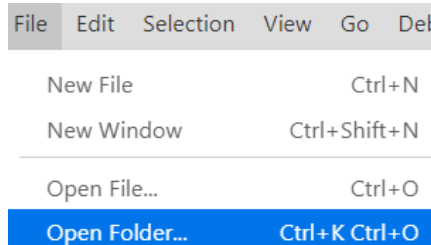
Open with Code

Scannen met Windows Defender...

Manier 2

Open VS Code en ga hier dan naar **File -> Open Folder ...**

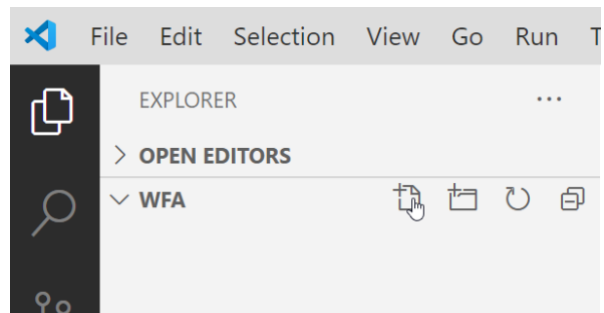
Kies hierbij dan de correcte folder waar je mee aan de slag wil gaan.



STAP 3

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

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



STAP 4

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

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

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

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

</body>
</html>
```

De startcode die voorzien wordt in VS Code

STAP 5

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

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

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

Updaten van de inhoud in het title element

In het body element voegen we vervolgens wat inhoud toe.

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

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

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

STAP 6

We kunnen het resultaat van onze uitwerking nu alvast even bekijken in onze browser. Druk hiervoor op **alt+b** om de pagina te openen in je standaard browser.
(Dit uiteraard op voorwaarde dat je de hiervoor nodige extensie geïnstalleerd hebt, zie Inleiding, Editoren, VS Code)

2.1.2 HTML CODE ONDER DE LOEP

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

2.1.2.1 STRUCTUUR VAN ONZE PAGINA

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

```
<!DOCTYPE html>
```

Declaratie van het type document

Vervolgens komen we het **html** element tegen. Het **html** element staat voor de 'root' (het element op het hoogste niveau) van een HTML-document. Daarom dat je in sommige referenties **root-element** kan terugvinden als ze het over dit element hebben. Alle andere elementen die je dus zal gebruiken, moeten binnen dit element hun plaats krijgen. Het **html** element heeft, net zoals alle andere elementen, de mogelijkheid om aangevuld te worden met **attributen**. Op onze eerste pagina hebben we reeds een language attribuut staan (*lang*) waar we verduidelijking kunnen geven aan de taal die hoofdzakelijk in het document gebruikt wordt. We kunnen dit dus gerust aanpassen naar **nl**, wat voor Nederlands staat.

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

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

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

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

Het head element

Het head element bevat 4 andere elementen waarvan er drie **meta** elementen zijn. Meta elementen komen later nog uitgebreider aan bod, maar we staan alvast toch even stil bij het eerste **meta** element. Het **meta** element met attribuut **charset** stelt ons in staat om de browser duidelijk te maken welke characterset er in het onderstaande document gebruikt wordt. Hier ga je standaard steeds UTF-8 invullen/tegenkomen.

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

Meta element met attribuut charset ingesteld op waarde UTF-8

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

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

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

Het title element op onze eerste pagina



Enkele richtlijnen en tips voor het samenstellen van goede titels:

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

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

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

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

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

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

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

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

2.1.3 COMMENTAAR IN ONZE CODE

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

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

Commentaar in onze code dient omringd worden door bovenstaande syntax

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

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

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

Onze eerste pagina, aangevuld met commentaar



Opgelet

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

2.1.4 ONZE EERSTE ELEMENTEN

2.1.4.1 HOOFDINGEN H1 ... H6

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

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

De verschillende niveaus van hoofdingen

Hoofding niveau 1

Hoofding niveau 2

Hoofding niveau 3

Hoofding niveau 4

Hoofding niveau 5

Hoofding niveau 6

Voorbeeld van de output in een browser



Enkele richtlijnen voor het gebruik van hoofdingen:

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

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

2.1.4.2 ALINEA – P

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

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

2.1.4.3 TOEPASSING

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

Leonardo da Vinci

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

Kunst

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

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

Wetenschap en techniek

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

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

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

Brontekst, Leonardo da Vinci

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

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

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

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

Voorbeeld weergave van de tekst in een browser

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

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

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

Toevoegen van h1 en h2 elementen

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

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

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

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

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

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

Leonardo da Vinci

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

Kunst

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

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

Wetenschap en techniek

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

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

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

Voorbeeld weergave van de aanpassingen in een browser.

2.1.4.4 BASISELEMENTEN SPAN EN DIV

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

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

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

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

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

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

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

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

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

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

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

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



Oefening

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



[Oefening 01 a via Github](#)

2.2 HET CSS BOX MODEL

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

2.2.1 WAT IS HET CSS BOX MODEL?

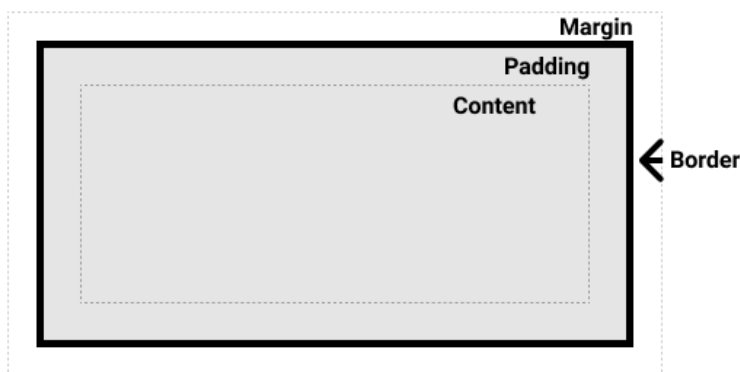
Alles in CSS heeft eigenlijk een **'box'** om zich heen. Deze **'box'** bestaat uit een viertal *vakken*, waar het begrijpen van hoe deze vakken werken essentieel is om lay-outs met CSS te kunnen maken of items uit te lijnen met elkaar. In dit stukje zullen we het **CSS Box model** even onder de loep nemen zodat je van start kan met de standaard en later ook de meer complexere lay-outtaken. We bekijken even de werking en de terminologie die hiermee gepaard gaat.

2.2.1.1 VAKKEN IN HET CSS BOX MODEL

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

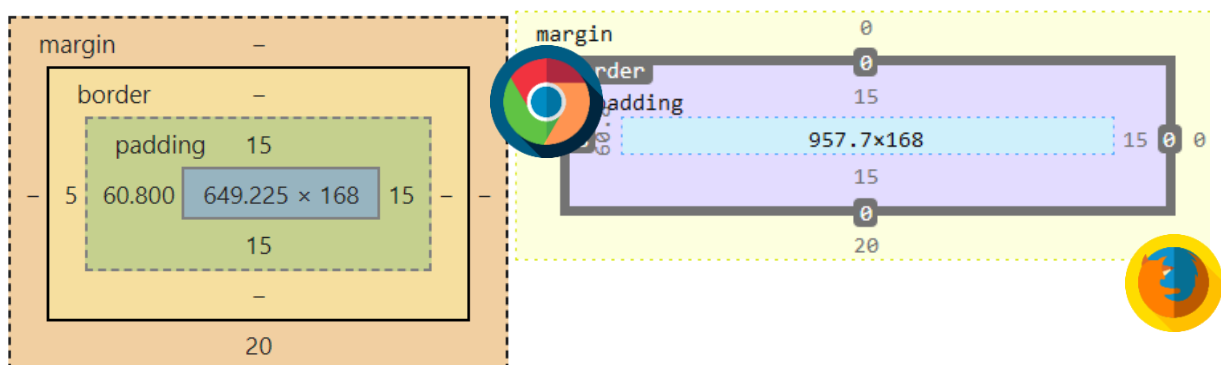
- **Content** (*inhoud*): het gebied waar de inhoud van het element wordt weergegeven. Aanpasbaar met eigenschappen zoals **width** en **height**.
- **Padding** (*opvulling*): de opvulling die zich bevindt rondom de inhoud als witruimte. Aanpassing kunnen hierop gebeuren door de property **padding**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

Het onderstaande diagram toont deze lagen:



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

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



2.2.2 INLINE EN BLOCK ELEMENTEN

Er bestaan hoofdzakelijk twee weergavetypes voor elementen: **Block** en **Inline**. **Block elementen** (zoals **p** en **div**) zullen steeds de volledig beschikbare ruimte innemen en geen elementen naast zich tolereren. Het is dus **'onmogelijk'** om twee **p** of **div** elementen **naast** elkaar te plaatsen. **Inline elementen** blijven netjes in dezelfde regel staan en nemen enkel de plaats in die ze nodig hebben op basis van hun inhoud. Inline elementen die we reeds gezien hebben zijn de **cite**, **mark**, **a** en **img**.

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

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

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

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

2.3 ONZE EERSTE CASCADING STYLE SHEET

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

STAP 1

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

STAP 2

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

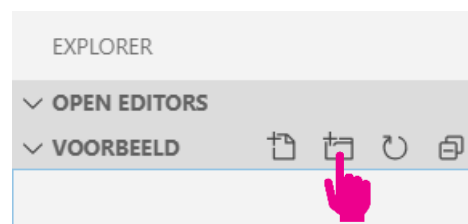
```
<h1>Introductie van CSS</h1>
<p>CSS wordt gebruikt voor de opmaak en het stijlen van onze pagina.</p>
<h2>CSS is tof!</h2>
<p>CSS is <span>super tof</span> om te gebruiken.</p>
```

Basis html invulling voor je pagina

STAP 3

Vervolgens maken we een nieuwe folder aan waar we onze Cascading Stylesheet in onder zullen brengen. Voeg via de explorer een **nieuwe folder** toe.

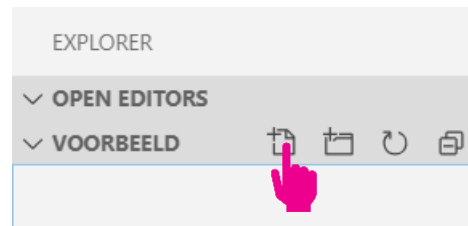
Dit doe je door op het icoontje **New Folder** te klikken naast de naam van je folder. Geef je nieuwe folder de naam **css**



STAP 4

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

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



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

STAP 5

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

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

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

Even verduidelijken wat we hier net aan onze code toevoegden.

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

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

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

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

2.3.1 CSS SELECTOR OP ELEMENT

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

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

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

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

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

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

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

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

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

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

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

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

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

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


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

Alle zijden van de margin instellen op 50 pixels breed.

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

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

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

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

```
p {
  ...
}

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

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

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

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

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

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

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

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

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


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

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



Oefening

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

 [Oefening 01 b via Github](#)

