Cascading Style Sheets (CSS3)

Versie 1.0, mei-2019

4865

Inhoudsopgave

Inhoudsopgave

Inhoudsopgave

Inhoudsopgave

1 Inleiding

1.1 Inleiding

In deze cursus zullen we werken met Cascading Style Sheets (CSS) om de vormgeving van webpagina's te verzorgen. Door de HTML structuur van webpagina's te scheiden van de CSS opmaak kunnen websites beter onderhouden worden.

In dit hoofdstuk zullen we een introductie geven tot CSS.

- De toepassing van CSS code kunnen benoemen
- Het kunnen toepassen van CSS code op webpagina's

1.1.1 Legenda

Leerdoelen voor een hoofdstuk. Hiervan ga je straks de vruchten plukken! Wat kan je allemaal na dit hoofdstuk? Deze leerdoelen matchen met de leerdoelen van het HTML 5 deel van het Microsoft web development examen.

Dit is een code voorbeeld, een blueprint. Deze mag je uitvoeren. Kijk of je het hele voorbeeld begrijpt.

Zelf aan de slag! Geef kleur aan je eigen oefening! Dit is waar het om draait in deze cursus, praktische ervaring opdoen.

Let op!
Dit is belangrijk en kan je eventueel veel tijd en zorgen besparen in de praktijk!

Expert tip. Wil je de expert worden in dit vak gebied? Neem deze tips dan ter harte.

Denk hier eens over na. Dit is vaak een stukje tekst of voorbeeld ter overweging.

Performance indicator. Dit onderdeel beschrijft impact op de performance van je webpagina.

1.2 Introductie tot CSS

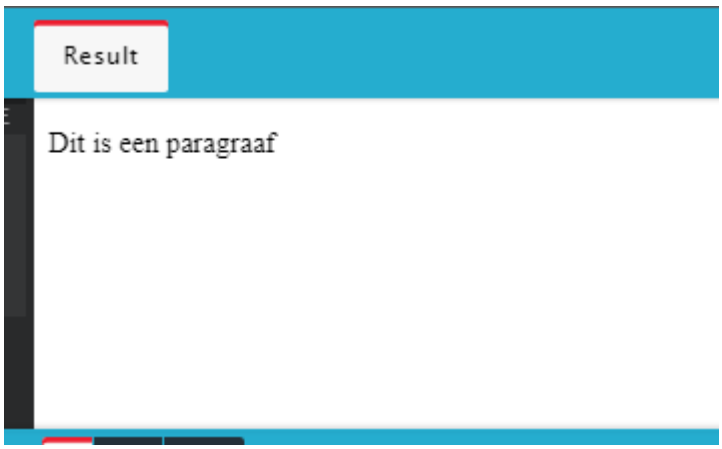
Webpagina's worden opgebouwd als een boomstructuur van HTML elementen. De meeste elementen hebben een begintag, een eindtag, en een inhoud. De begintag bevat de naam van het element, omsloten door de punthaken < en >. De eindtag ziet er hetzelfde uit, alleen staat er na de eerste punthaak een forward slash /.

1 Inleiding

Onderstaand voorbeeld is een eenvoudig HTML document:

```
<html>
  <head>
    <title>Basisdocument</title>
  </head>
  <body>
    <p>Dit is een paragraaf</p>
  </body>
</html>
```

Resultaat (codepen):



We zien hier bijvoorbeeld dat het element title bestaat uit de begintag "<title>", de eindtag "</title>", en de inhoud "Basisdocument". Ook is duidelijk te zien dat het html-element het hele document met alle subelementen bevat.

In HTML kunnen we onderscheid maken tussen structuur elementen en opmaak elementen. Deze laatste groep elementen is bedoeld om de vormgeving te beïnvloeden.

In het volgende voorbeeld hebben we tekst in de paragraaf van kleur voorzien. Verder is een tabel toegevoegd waarbij we alle kopjes in het groen, en de inhoud cursief op het scherm willen zien. Bekijk de code in onderstaande codepen:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Basisdocument</title>
  </head>
  <body>
    <p><font color="green">Dit is een paragraaf</font></p>
    <table border="1">
    <tr>
      <th><font color="green">Kolomkop 1</font></th>
      <th><font color="green">Kolomkop 2</font></th>
      <th><font color="green">Kolomkop 3</font></th>
    </tr>
    <tr>
      <td><i>Tekstregel 1</i></td>
```

```
      <td><i>Tekstregel 1</i></td>
      <td><i>Tekstregel 1</i></td>
    </tr>
    <tr>
      <td><i>Tekstregel 2</i></td>
      <td><i>Tekstregel 2</i></td>
      <td><i>Tekstregel 2</i></td>
    </tr>
    </table>
  </body>
</html>
```

Resultaat:

De opmaak elementen die we in dit voorbeeld hebben toegepast zijn <font> en . De tag <font> regelt de opmaak van het lettertype, de tags maken de tekst cursief. Zoals u kunt zien moeten we hier dus voor elke tekst in een cel die we cursief willen zien de codes en typen.

Er kleven enkele belangrijke nadelen aan deze manier van vormgeven:

- Er is veel herhaling van dezelfde tags nodig (arbeidsintensief);
- Bij wijziging van opmaak (bijvoorbeeld alle kolomkoppen rood maken), moet dat voor alle betreffende elementen opnieuw gebeuren;
- Toevoegen van veel van dergelijke opmaak elementen ontneemt het zicht op de structuur van het document.

Deze nadelen wegen des te zwaarder bij het onderhoud van meerdere pagina's van een website. Het is immers goed gebruik om een consistente vormgeving na te streven voor alle pagina's binnen een site. Stelt u zich even voor dat alle cellen van tabellen voortaan vet gedrukt in plaats van cursief moeten worden weergegeven. Nog afgezien van het feit dat deze wijziging veel tijd gaat kosten, sluipen er makkelijk fouten in, zoals het overslaan van sommige cellen, of typefouten bij het aanmaken van de opmaak tags.

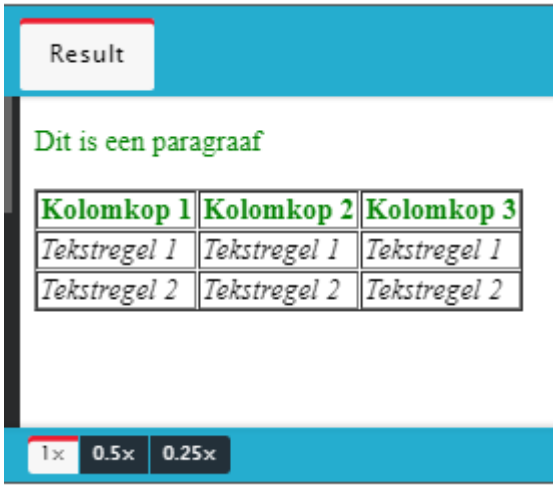
Hoeveel handiger zou het zijn als we met één opdracht alle cellen konden selecteren! We kunnen dan vervolgens "cursief gedrukt" vervangen door "vet gedrukt, en de HTML structuur verder ongemoeid laten.

1 Inleiding

Dit is precies waar CSS voor is bedoeld. Met CSS kunnen we vormgeving toepassen op specifieke delen van de structuur van een HTML document. We zullen nu bovenstaand voorbeeld vervangen door een versie waarin CSS wordt gebruikt voor de opmaak.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Basisdocument</title>
    <style type="text/css">
      p  { color: green; }
      th { color: green; }
      td { font-style: italic; }
    </style>
  </head>
  <body>
    <p>Dit is een paragraaf</p>
    <table border="1">
    <tr>
      <th>Kolomkop 1</th>
      <th>Kolomkop 2</th>
      <th>Kolomkop 3</th>
    </tr>
    <tr>
      <td>Tekstregel 1</td>
      <td>Tekstregel 1</td>
      <td>Tekstregel 1</td>
    </tr>
    <tr>
      <td>Tekstregel 2</td>
      <td>Tekstregel 2</td>
      <td>Tekstregel 2</td>
    </tr>
    </table>
  </body>
</html>
```

Resultaat:



Hierin vallen twee zaken op:

- er is een style element toegevoegd binnen het head element;
- de opmaak elementen <i> en <font> zijn verdwenen: alleen de structuur elementen zijn nog over.

Binnen het style element staat de CSS code. In dit geval wordt voor de elementen p, th en td de opmaak geregeld met behulp van CSS. Vooruitlopend op het volgende hoofdstuk bekijken we hier vast een regel van:

```
th { color: green; }
```

Links van de accolades herkennen we de element naam th. Het selecteert alle th elementen binnen het document. Voor de dubbele punt staat een eigenschap (color), daarachter staat de waarde voor die eigenschap (green).

Bekijk het vorige voorbeeld nogmaals.

Vervang nu de waarde 'green' door 'blue' en bekijk het resultaat.

Merk op dat we alleen de CSS code hoefde aan te passen om de opmaak te regelen: de HTML code blijft verder ongewijzigd.

1.3 Samenvatting

HTML kent structuur en opmaak elementen. Het gebruik van opmaak elementen maakt het bouwen en onderhouden van websites omslachtig en foutgevoelig. Met behulp van CSS kunnen we de structuur en de opmaak gescheiden houden.

De HTML code kan dan beperkt worden tot de structuur elementen, de opmaak regelen we met CSS code.

1 Inleiding

2 Opbouw van den CSS document

2.1 Inleiding

Dit hoofdstuk gaat nader in op de structuur van een CSS stylesheet. Eerst zullen selectors worden besproken. Hiermee geven we aan welk deel van het document we willen aanpassen. Verder behandelen we op welke manieren we CSS op een webpagina kunnen toepassen. Bij het gebruik van CSS is het belangrijk onderscheid te maken tussen enkele verschillende soorten HTML structuur elementen. Bij het bespreken van deze verschillen zal ook het Document Object Model (DOM) aan bod komen. Tot slot bespreken we in dit hoofdstuk wat overerving binnen CSS inhoudt.

- Het kunnen opzetten van een eenvoudig CSS document en de verschillende style vormen kunnen benoemen
- Het verschil tussen de verschillende selectoren kunnen benoemen
- Verschillende soorten HTML structuur elementen kunnen onderscheiden en benoemen
- CSS code op verschillende manieren kunnen koppelen aan webpagina's
- Het verband tussen de DOM structuur en CSS overerving kunnen benoemen

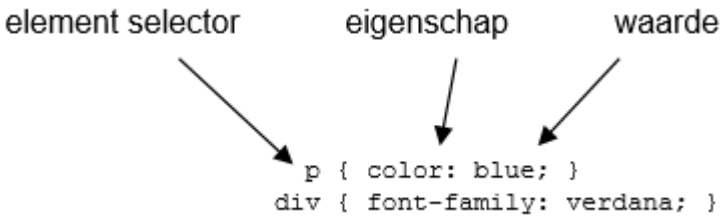
2.2 Selectors

In het voorgaande hoofdstuk hebben we gezien hoe we met één regel CSS code de opmaak van alle elementen van een bepaald type (zoals p, th of td) kunnen aanpassen. De eerste stap bij het toepassen van CSS code is het selecteren van precies dat deel van het document, waarvan we een opmaak eigenschap willen wijzigen.

```
<html>
  <head>
    <title>Oefening met een eenvoudige stylesheet</title>
    <style>
      p { color: blue; }
      div { font-family: verdana; }
    </style>
  </head>
  <body>
    <p>Dit is een paragraaf</p>
    <div>Dit is een div</div>
  </body>
</html>
```

2 Opbouw van den CSS document

In dit bestand ziet u weer een stuk code staan tussen <style> tags. Dit is de CSS code die dit bestand gebruikt. We zien hier dat er twee verwijzingen naar elementen zijn, namelijk p en div. Een dergelijke verwijzing heet ook wel de element selector. Naast de selector ziet u tussen accolades een eigenschap en een waarde staan. Dit is de eigenschap van het element die we gaan veranderen met de stijlregel. In bovenstaand voorbeeld worden dus twee zaken aangepast. Ten eerste wordt de tekstkleur van de paragrafen (p) blue. Daarnaast wordt het lettertype van alle div's verdana. We zien hieronder een grafische toelichting op element selectors en eigenschappen met waarden.



Veel eigenschappen binnen CSS werden tot voor kort per browser onder een andere naam geïmplementeerd. Er werd, om deze eigenschappen te kunnen gebruiken, een browser prefix voor de eigenschap naam gezet.

Een overzicht van de voornaamste prefixes die gebruikt kunnen worden per browser:

- -webkit- (Chrome, Safari, nieuwe versies van Opera, praktisch alle iOS browsers)
- -moz- (Firefox)
- -o- (Oude, pre-WebKit, versies van Opera)
- -ms- (Internet Explorer en Microsoft Edge)

Tegenwoordig zijn deze prefixes bijna niet meer nodig, maar het is alsnog aan te raden deze altijd te vermelden, om er zeker van te zijn dat je code in verschillende browsers werkt.

Een goed overzicht van eigenschappen die mogelijk browser-prefixes nodig hebben vindt u hier op de webpagina van Peter Beverloo, developer bij Google:

http://peter.sh/experiments/vendor-prefixed-css-property-overview/

Sla deze pagina op in uw favorieten! In deze cursus zullen wij verder (alvast) geen prefixes meer gebruiken. Dus waar u in andere browsers problemen ervaart; sla dan deze webpagina erop na.

2 Opbouw van den CSS document

In het volgende voorbeeld zullen we een eigenschap (kleur) wijzigen voor verschillende delen van een document. Dit bestand bevat code met enkele koppen (h1 en h2) en paragrafen (p). De laatste kop (h2) en paragrafen (p) zijn onderdeel van een div element.

```
<html>
  <head>
    <title>Selectors</title>
    <style>
      h1  {color: blue}
    </style>
  </head>
  <body>
    <h1>Selectors</h1>
    <h2>Paragrafen</h1>
    <p>Dit is de eerste paragraaf binnen deze pagina. <br/>
    Met behulp van de selector p kunnen alle paragrafen
worden aangewezen,
    om er een bepaalde opmaak aan toe te kennen.</p>
   <p>Dit is de tweede paragraaf binnen deze pagina. <br/>
    Met behulp van de selector p kunnen alle paragrafen
worden aangewezen,
    om er een bepaalde opmaak aan toe te kennen.</p>
    <div>
    <h2>Het div element</h2>
     <p>Deze derde paragraaf is onderdeel van een div
element.<br/>
     Het div element is bedoeld om delen van een pagina
structureel bij elkaar te houden.
     </p>
     <p>
     De tweede kop staat samen met de derde en vierde
paragraaf bij elkaar in &eacute;&eacute;n div element.<br/>
     </p>
    </div>
  </body>
</html>
```

2 Opbouw van den CSS document

In de browser kunnen we zien dat de kop (h1) blauw is weergegeven, als gevolg van de CSS regel h1 {color: blue}. In de volgende oefening zullen we het style element aanvullen, om het effect van selectors nader te bestuderen.

. Voeg nu achtereenvolgens de volgende CSS regels toe aan het style element in het bovenstaande codeblok, en bekijk het resultaat:
p {color: grey}
Het p element heeft nu alle paragrafen geselecteerd: de tekst wordt grijs weergegeven. Voeg nu toe:
div {color: purple}
Merkwaardig genoeg wordt nu alleen de tweede subkop (h2) in paars weergegeven, ook al bevat het div element ook p elementen! We hadden eerder echter de p elementen al aangepast. De regel voor het div element geldt voor alle subelementen waar nog geen opmaak aan was toegekend.

Haal nu de regel p {color: grey} weg.

Nu zien we dat de tekst van alle subelementen van het div element (h2 en p) paars worden weergegeven.

Uit voorgaand voorbeeld blijkt dat we met behulp van selectors snel en eenvoudig delen van het HTML document kunnen selecteren. Ook hebben we gezien dat bij het toepassen van overlappende CSS eigenschappen voorrangsregels worden gehanteerd. We komen hier later in deze cursus uitvoeriger op terug.

2 Opbouw van den CSS document

In de komende paragrafen zullen we verschillende soorten selectors bespreken.

2.2.1 Type selector

Een type selector is een verwijzing binnen CSS naar een type HTML element. De naam van de selector komt overeen met de naam van het element. Als we bijvoorbeeld de selector div{} in CSS opnemen, kunnen we daarmee de stijl van alle DIV elementen in de HTML pagina veranderen. Voor elk type element is een selector beschikbaar, zoals p{}, span{}, input{}, enzovoorts.

Hier volgen enkele voorbeelden met een een type selector:

```
p {
color:              green;
font-size:          80%;
}
```

De tekstkleur binnen een paragraaf wordt hierdoor groen en krijgt als grootte 80 procent van de standaard tekstgrootte binnen het document.

We zien dat tussen de accolades alle te wijzigen eigenschappen genoteerd worden, met daarachter een dubbele punt en de nieuwe waarde voor die eigenschap. Als er nog een eigenschap volgt, moet de waarde daarvoor worden afgesloten met een puntkomma. In dit voorbeeld zien we ook achter de laatste waarde een puntkomma: deze is optioneel.

```
input {
  border-style:   solid;
  border-width:   1px;
  border-color:   black;
}
```

In dit voorbeeld zien we dat we soms meer eigenschappen moeten aanpassen om een effect te verkrijgen. In dit geval krijgt een invoerveld met als tag <input> een zwarte rand: de border-style is solid (geen diepte of onderbreking), met als breedte één pixel en de kleur zwart. Meer over deze eigenschappen volgt in een volgend hoofdstuk.

Het komt vaak voor dat we een zelfde stijl aan meerdere elementen willen toekennen. Dit kan als volgt. Door onderstaande code wordt de tekst binnen de p en binnen de th elementen groen.

```
<style>
      p  { color:       green; }
      th { color:       green; }
      td { font-style:  italic; }

</style>
```

Maar het kán ook eenvoudiger. In het volgende voorbeeld ziet u hoe we dit dus logischer (en korter) noteren.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p, th  { color:         green; }
      td { font-style:  italic; }
    </style>
  </head>
  <body>
    <p>Dit is een paragraaf</p>
    <table>
      <tr>
        <th>Kolomkop 1</th>
        <th>Kolomkop 2</th>
      </tr>
      <tr>
        <td>Cel kolom 1</td>
        <td>Cel kolom 2</td>
      </tr>
    </table>
  </body>
```

In dit geval lijkt het verschil klein, maar vooral als complexere stijlen worden toegepast op meerdere elementen kan deze notatie de onderhoudbaarheid vergroten. Let wel: dit hoeft niet het geval te zijn! Gebruik deze notatie alleen als wijzigingen van de opmaak steeds op dezelfde groep elementtypen zullen worden toegepast. Neem bijvoorbeeld de eerder genoemde stijl voor input. We breiden deze stijl nu uit naar meerdere formulier-elementen.

```
input, select, textarea {
  border-style:   solid;
  border-width:   1px;
  border-color:   black;
}
```

Mochten we later besluiten dat formulier-elementen een bredere rand moeten krijgen, dan hoeven we dat maar op één plek te regelen:

```
input, select, textarea {
  border-style:   solid;
  border-width:   2px;
  border-color:   black;  }
```

2.2.2 class selector

Met behulp van een klasse kunnen stijlen aan groepen HTML elementen worden toegekend, buiten de eerder behandelde type selectors om. Een class selector heeft een zelf gekozen naam. Hieraan kunnen weer eigenschappen en waarden worden toegekend. In de HTML code kunnen we vervolgens met behulp van het attribuut class aangeven welke elementen voor deze stijl in aanmerking komen. De syntax voor de class selector is als volgt in HTML:

```
<elementnaam class="klassenaam">
```

De syntax voor de verwijzing ernaar in CSS is:

```
[element].klassenaam { }
```

De vierkante haken [en] geven aan dat het opgeven van een elementnaam niet verplicht is. U dient deze vierkante haken dus zelf niet over te nemen. Zonder elementnaam wordt elk element van de betreffende klasse geselecteerd. Enkele voorbeelden van class selectors zijn: .kop{}, div.klein{} of input.login{}.

Een voorbeeld zien we in onderstaande HTML code:

```
<html>
  <head>
    <title>Voorbeeld selective class selectors</title>
    <style>
    div.klein {
      color:            green;
      font-size:  80%; }
    </style>
  </head>
  <body>
  <div class="klein">Deze div heeft als klassenaam
'klein'.</div>
  <div>Deze div daarentegen heeft geen klassenaam en zal
dan ook niet door de
beschreven stijlregel beïnvloed worden.
  </div>
  </body>
</html>
```

We zien hier dat de klasse "klein" als attribuut is toegekend aan het eerste div element. Alleen dat element zal dus worden geselecteerd met de selector div.klein. Er mogen meerdere elementen gebruikt worden met eenzelfde klassenaam.

Een class naam mag nooit beginnen met een cijfer. Dit zal onder andere de browser Firefox niet accepteren.

Houd er in de naamgeving van klassen wel rekening mee dat wensen met betrekking tot opmaak kunnen veranderen. Wat bijvoorbeeld als uw opdrachtgever in al zijn wijsheid besluit dat de eerste div toch in grote letters moet worden weergegeven? U kunt dan ongestraft de font-size van div.klein wijzigen in 120%, maar dat maakt de code er niet leesbaarder op. In dit geval zouden we er voor kunnen kiezen om een extra klasse div.groot te maken, maar dan moeten we ook de HTML code weer aanpassen.

Vaak is het verstandiger om met de naamgeving van klassen aan te sluiten op de inhoud of de structuur van het document, in plaats van op de op vormgeving van dat moment. Klassenamen als 'koptekst', 'opvallend', of 'inleiding' zullen op den duur de lading beter dekken dan namen als 'groot', 'vet' of 'blauw'.

Het is ook mogelijk om meerdere klassen aan een element toe te kennen! We kunnen dan in het attribuut class de verschillende klassen achter elkaar noemen, door spaties gescheiden.

De mogelijkheid om meerdere klassen aan één element toe te kennen heeft als voordeel dat we niet teveel verschillende soorten eigenschappen (zoals kleur en vorm) in één klasse hoeven onder te brengen. Dat maakt de code homogener, leesbaarder en beter onderhoudbaar.

Maak in het vorige voorbeeld een extra klasse 'opvallend' aan:

```
.opvallend {
        text-decoration:      underline;
        font-style:           italic;
        font-weight:          bold
      }
```

Pas daarna het attribuut class van de het eerste div element aan:

```
<div class="klein opvallend">Deze div heeft de klassen
'klein' en 'opvallend'.</div>
```

Bekijk nu het resultaat en probeer te begrijpen wat er gebeurd.

De mogelijkheid om meerdere klassen aan één element toe te kennen heeft als voordeel dat we niet teveel verschillende soorten eigenschappen (zoals kleur en vorm) in één klasse hoeven onder te brengen. Dat maakt de code homogener, leesbaarder en beter onderhoudbaar.

2.2.3 ID selector

In voorgaande paragraaf hebben we gezien hoe we met een class selector naar meedere elementen van een bepaalde klasse kunnen verwijzen. Een id selector lijkt daarop: een element krijgt als attribuut een id toegewezen, waar we in CSS naar kunnen verwijzen. Een bepaald id is echter maar voor één element van toepassing.

2 Opbouw van den CSS document

De syntax voor een id selector in HTML is als volgt:
`<elementnaam id="idnaam">`

De syntax voor de verwijzing ernaar in CSS is:

```
#idnaam { }
```

Enkele voorbeelden zijn: #menu, #bodem, #kop enzovoorts.

```
<html>
  <head>
    <title>Voorbeeld id selector</title>
    <style>
    #menu {
      color:            purple;
      font-size:  120%; }
    </style>
  </head>
  <body>
  <div id="menu">Deze div heeft als id 'menu'.</div>
  <div>Deze div daarin tegen heeft alleen een klassenaam en
zal dan ook niet door de
beschreven stijlregel beïnvloed worden.
  </div>
  </body>
</html>
```

We zien in de HTML code dat een div element een attribuut id heeft gekregen, met als waarde menu. In de CSS code verwijzen we naar dat element via de selector #menu. Een element kan altijd maar één id hebben, en een id selector mag niet aan meerdere elementen op een pagina worden toegekend. Wanneer men zich hier niet aan houdt, wordt de laatste toekenning van het id aangenomen.

In HTML zijn de attribuut waarden van class en id *altijd* hoofdletter gevoelig.

2.2.4 Descendant selectors

Vaak zijn webpagina's opgebouwd uit blokken HTML-elementen, die elk hun eigen stijl hebben. Bijvoorbeeld: boven een menu, daaronder "basis" tekst, een kolom met nieuws-items rechts, en onderaan de copyright informatie. In alle blokken wordt van dezelfde soort elementen gebruik gemaakt, zoals links en list-items. Om deze van een andere stijl te voorzien zou gebruik kunnen worden gemaakt van classes.

Bijvoorbeeld:

```
<!-- code voor menu -->
<a class="menu" href="url1.html">eerste menu link</a>
<a class="menu" href="url2.html">tweede menu link</a>
<a class="menu" href="url3.html">derde menu link</a>

<!-- code voor nieuwsitems -->
<a class="nieuws" href="url4.html">eerste nieuws link</a>
<a class="nieuws" href="url5.html">tweede nieuws link</a>
<a class="nieuws" href="url6.html">derde nieuws link</a>
```

Merk op dat we hier weer HTML-code aanmaken die we eerder probeerden te vermijden. Overal worden weer (class-)attributen toegevoegd om de stijl aan te passen. Om dit te voorkomen kunnen we gebruik maken van descendant selectors. We kunnen blokken elementen samenvoegen binnen een div met een id, bijvoorbeeld:

```
<div id="menu">
<a href="url1.html">eerste menu link</a>
<a href="url2.html">tweede menu link</a>
<a href="url3.html">derde menu link</a>
</div>
```

In de stijlsheet kunnen we hiernaar verwijzen met een selector, die verwijst naar elementen binnen de div met het betreffende id, bijvoorbeeld:

```
<style>
  #menu a   {color: red}
  #nieuws a {color: green}
<style>
```

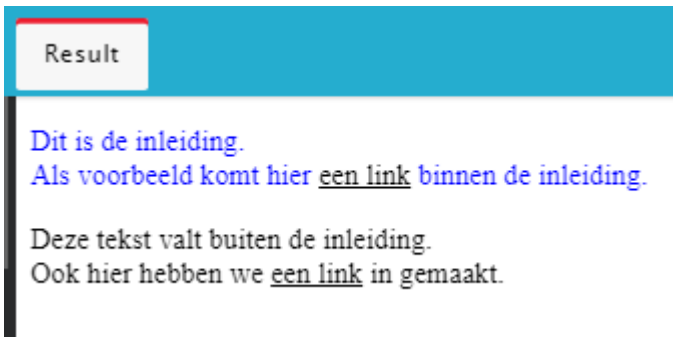
We zien hier direct achter de id-naam een type-selector staan. Een descendant selector als #menu a{} is te lezen als "alle a elementen binnen het element met id menu".

Descendant selectors kunnen ook uit meer selectors zijn opgebouwd, en zijn niet gebonden aan id selectors. Zo kunnen we bijvoorbeeld ook naar links verwijzen die deel uitmaken van list items in een div:

```
div li a { color : red }
```

Hoewel een li element een ul of ol element als directe parent heeft, hoeven we die niet op te nemen in de CSS code. Deze code is dus te lezen als: alle a elementen die deel uitmaken van een li element, ergens in een div element.

```
<html>
  <head>
    <title>Voorbeeld descendant selector</title>
    <style>
    a { color: black }
    #inleiding { color: blue }
    </style>
  </head>
  <body>
    <div id="inleiding">
        <p>Dit is de inleiding. <br/>
         Als voorbeeld komt hier  <a
href="descendant_selector.html">een link</a> binnen de
inleiding.</p>
    </div>
    <div>
       <p>Deze tekst valt buiten de inleiding.<br/>
       Ook hier hebben we <a
href="descendant_selector.html">een link</a> in
gemaakt.</p>
   </div>
  </body>
</html>
```



Hierin zijn twee regels opgenomen: links moeten in het zwart worden weergegeven, tekst binnen het element met id "inleiding" moeten in het blauw worden getoond. Als gevolg van de eerste regel wordt ook de link binnen de inleiding zwart weergegeven. De eerste regel heeft voor links dus voorrang op de tweede regel.

Als dat niet de bedoeling is kunnen we een uitzondering op de eerste regel maken door een descendant selector toe te voegen.

Dit gaan we in onze volgende opdracht doen.

Open het vorige voorbeeld (zie bovenstaand).

Zorg ervoor dat ook links binnen het element met id 'inleiding' blauw worden weergegeven. Voeg daartoe de volgende descendant selector toe en bekijk het resultaat:

```
#inleiding a { color: blue }
```

Nu hebben we ervoor gezorgd dat ook links binnen de inleiding blauw worden weergegeven.

Als deze kleur voor alle soorten elementen binnen inleiding moet gelden dan kunnen we ook gebruik maken van de universele selector: *.
Het sterretje * staat hier voor de naam van elk elementtype.
Haal de laatste CSS regel weer weg, en wijzig #inleiding {color: blue} in:

```
#inleiding * { color: blue }
```

Ook nu blijven links binnen de inleiding blauw gekleurd.

Er is nog een aantal selectors beschikbaar die we niet hebben besproken. De belangrijkste hiervan staan in de onderstaande tabel beschreven. Deze selectors zijn niet compatibel met Internet Explorer 7.0.

Selector type	**Syntax**	**Omschrijving**
Adjacent sibling	element1 + element2	element2 volgt direct na element1
Child	element1 > element2	element2 is een child element van element1
Simple attribute	element[attribuutnaam]	het element beschikt over het aangegeven attribuut
Exact attribute value	element[attribuutnaam="waarde"]	element dat beschikt over het attribuut attribuutnaam met de opgegeven waarde

Met pseudo classes kunnen we een stijl toekennen aan elementen die in een bepaalde toestand verkeren, zoals links die wel of niet bezocht zijn, of elementen die focus krijgen. Pseudo classes zijn te verdelen in twee groepen, namelijk: de link gerelateerde klassen en de dynamische pseudo klassen.

Link pseudo classes
Deze klassen worden gebruikt om naar de status te verwijzen die op een link van toepassing is (zoals: een bezochte link). Om te kunnen verwijzen naar een link die bezocht is gebruiken we tussen het element en de pseudo klasse een dubbele punt.
De syntax hiervoor is:

syntax: a:sleutelwoord { eigenschap: waarde; }
sleutelwoord: active | link | visited
default: geen

:active Verwijst naar een momenteel actieve link.
:link Verwijst naar een onbezochte link. De a tag moet een href attribuut hebben
:visited Verwijst naar een bezochte link.

We kunnen bijvoorbeeld een link paars kleuren wanneer deze bezocht is met behulp van de volgende code: a:visited {color: purple; }. De tweede link pseudo klasse is :link. Hiermee kunnen we direct verwijzen naar een link, en niet zozeer naar een a tag (waar ook weleens geen href attribuut aan gekoppeld kan zijn). Er kan hiermee dus de stijl ingesteld worden van een link die nog niet bezocht is. We kunnen ook link pseudo classes gebruiken op links waarnaar verwezen wordt met een id of klasse. Bekijk het volgende voorbeeld:

```
<a href="http://www.vijfhart.nl">
<a href="#" id="legelink">
```

Als we vervolgens naar deze links willen verwijzen met een link pseudo class, dan kan dat door de volgende code in een stylesheet te gebruiken:

```
a:link {color: white; }
a:visited {color: red; }
```

2.2.4.1 Dynamic pseudo classes

Deze klassen zijn vanaf CSS2.1 geïmplementeerd en hebben als doel om elementen op een website dynamisch te laten reageren op gebruikersacties. Deze klassen zijn ook op links van toepassing. We kunnen bijvoorbeeld de randkleur van een tekstveld veranderen wanneer een gebruiker hierin zijn cursor heeft staan. Onderstaand volgt een dergelijk voorbeeld. Het stukje code in de webpagina is als volgt:

```
<input type="text" name="txtNaam">
```

In de stylesheet kan dan staan:

```
input {
  border-style:   solid;
  border-width:   1px;
  border-color:   black; }

input:focus {
  border-color:   orange; }
```

De eerste regel zorgt ervoor dat het invoerveld standaard een zwarte rand heeft. De tweede regel gaat in werking zodra dit veld de focus krijgt, door de cursor in dat veld te plaatsen. De rand om het invoerveld wordt dan oranje. Wij voeren dit uit in onderstaande voorbeeld:

```
<html>
<head>
<style>
input {
  border-style:   solid;
  border-width:   1px;
  border-color:   black;
}

input:focus {
  border-color:   orange;
  border-width: 2px;
}
</style>
</head>
```

```
<body>
  Voer uw naam in: <input type="text" name="txtNaam">
</body>
</HTML>
```

We zien dat de rand om het veld verandert zwart naar oranje zodra we in het veld klikken. Let er wel op; sommige browsers geven ook / of een eigen effect.

Hier volgt een overzicht van alle mogelijke dynamic pseudo classes. Omdat dit er teveel zijn om zo uit ons hoofd te leren, zullen wij een aantal belangrijke gaan toelichten in een voorbeeld.

Let op! Sommige selectors beginnen met TWEE maal een dubbele punt (::). Deze zullen wij ook beide moeten gebruiken indien aangegeven. Anders zal de selector niet herkend worden.

- :active — Selecteert de gekozen actieve link.
- ::after — Plaatst content achter de inhoud van het gekozen element.
- ::before — Plaats content voor de inhoud van het gekozen element.
- :checked — Selecteert ieder aangevinkt input element.
- :disabled — Selecteert ieder gekozen input element dat het ‘disabled’ attribuut heeft.
- :empty — Selecteert ieder gekozen element dat geen enkele child nodes onder zich heeft.
- :enabled — Selecteert ieder input element dat niet disabled is.
- :first-child — Selecteert ieder gekozen element dat zelf het eerste element van zijn parent is.
- ::first-letter — Selecteert de eerste letter van ieder gekozen element.
- ::first-line — Selecteert de eerste regel van ieder gekozen element.
- :first-of-type — Selecteert ieder gekozen element dat zelf het eerste element van dit type binnen zijn parent is.
- :focus — Selecteert het gekozen element dat momenteel focus heeft.
- :hover — Selecteert het gekozen element waar met de muis overheen bewogen wordt.
- :in-rage — Selecteert ieder gekozen input element die qua waarden binnen de opgegeven range zit.
- :invalid — Selecteert ieder gekozen input element die een invalide invoer heeft ontvangen.

- :lang(language) Selecteert ieder gekozen element, waarvan het ‘lang’ attribuut de afgekorte land notatie ingevuld heeft staan.
- :last-child Selecteert ieder gekozen element die zelf de laatste child zijn binnen zijn parent.
- :last-of-type Selecteert ieder gekozen element dat zelf het laatste element van dit type binnen zijn parent is.
- :link Selecteert alle onbezochte links
- :not(selector) Selecteert ieder gekozen element dat niet van het type is dat tussen haakjes erachter staatl
- :nth-child(n) Selecteert ieder gekozen element dat zelf het laatste element van dit type binnen zijn parent is.
- :nth-last-child(n) Selecteert ieder gekozen element dat zelf het n-de (gekozen) element is binnen de parent. Begint bij de laatste child te tellen.
- :nth-last-of-type(n) Selecteert ieder gekozen element dat zelf het laatste element is van het n-de (gekozen) type binnen de parent.
- :nth-of-type(n) Selecteert ieder gekozen element dat het n-de (gekozen) element is van zijn parent.
- :only-of-type Selecteert ieder gekozen element dat de enige is zijn zijn type binnen zijn parent.
- :only-child Selecteert ieder gekozen element dat de enige child is van zijn parent.
- :optional Selecteert ieder gekozen element dat geen ‘required’ attribuut heeft ingesteld.
- :out-of-range Selecteert ieder gekozen input element die qua waarden buiten de opgegeven range zitten.
- :read-only Selecteert ieder gekozen input element dat een ‘read-only’ attribuut heeft ingesteld.
- :read-write Selecteert ieder gekozen input element dat géén ‘read-only’ attribuut heeft ingesteld.
- :required Selecteert ieder gekozen input element dat een ‘required’ attribuut heeft ingesteld.
- :root Selecteert het root element van deze pagina.
- ::selection Selecteert het deel van een element dat geselecteerd is de gebruiker.
- :target Selecteert het actieve gekozen element (waarna er op een link geklikt is met een anker met deze naam erin)
- :valid Selecteert ieder gekozen input element dat een valide invoer heeft ontvangen.
- :visited Selecteert alle bezochte links

Wanneer we de :hover classe gebruiken moet deze na een eventuele :link en/of :visited classe zijn vermeld in de stylesheet. Wordt de :hover classe ervoor genoemd, dan zal deze geen effect meer hebben. Voor :active geldt verder dat deze na een eventuele :hover moet worden geplaatst.
Bekijk de volgende voorbeelden om een beter beeld te krijgen van wat deze selectors doen:

```
<html>
<head>
<style>
/* focus wordt vaak op invoervelden toegepast */
input:focus {
```

```
  background-color: orange;
}

/* hover die een afbeelding bron veranderd */
#plaatje {
  width: 150px;
  height: 200px;
  background-image: url("https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bw.jpg");
}

#plaatje:hover {
  background-image: url("https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/kleur.jpg");
}

/* De first-letter selector wordt vaak gebruikt om
capitalen mee aan te duiden.  */
#hoofdletter:first-letter {
  font-size: 3em;
}


/* Let op dat we hier geen nummer, maar 'even' en 'odd'
gebruiken */
tr:nth-child(even) {
  background-color: lightgray;
  color: white;
}

tr:nth-child(odd) {
  background-color: gray;
  color: white;
}
</style>
</head>

<body>
  <p>Klik in het onderstaande veld om de 'focus' selector
te triggeren:</p>
  <input type="datetime" placeholder="Klik hier"/></br>
<p>Beweeg uw muis over deze afbeelding van een uniek
Nederlands boek:</p>
  <div id="plaatje"></div>
  <p id="hoofdletter">Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Aenean commodo ligula eget
dolor. Aenean massa. Cum sociis natoque penatibus et magnis
dis parturient montes, nascetur ridiculus mus. Donec quam
felis, ultricies nec, pellentesque eu, pretium quis, sem.
Nulla consequat massa quis enim. Donec pede justo,
fringilla vel, aliquet nec, vulputate eget, arcu. In enim
justo, rhoncus ut, imperdiet a, venenatis vitae, justo.
Nullam dictum felis eu pede mollis pretium. Integer
```

```
tincidunt. Cras dapibus. Vivamus elementum semper nisi.
Aenean vulputate eleifend tellus. Aenean leo ligula,
porttitor eu, consequat vitae, eleifend ac, enim. Aliquam
lorem ante, dapibus in, viverra quis, feugiat a, tellus.
Phasellus viverra nulla ut metus varius laoreet. Quisque
rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue.
Curabitur ullamcorper ultricies nisi. Nam eget dui.</p>
  <table>
    <tr>
      <th>Naam voetbalclub</th>
      <th>Jaar van oprichting</th>
    </tr>
    <tr>
      <td>Manchester United</td>
      <td>1878</td>
    </tr>
    <tr>
      <td>Real Madrid</td>
      <td>1902</td>
    </tr>
    <tr>
      <td>FC Barcelona</td>
      <td>1899</td>
    </tr>
    <tr>
      <td>Chelsea</td>
      <td>1905</td>
    </tr>
  </table>
</body>
</HTML>
```

2.3 Opname van stijlen in HTML met CSS

We kunnen op vier verschillende manieren CSS code opnemen in een web pagina. De code kan daarbij rechtstreeks in de webpagina geplaatst worden of het kan in een ander bestand gezet worden waarnaar verwezen kan worden.

De verschillende manieren voor het opnemen van CSS code staan hier weergegeven:

Stylesheet vorm	**Toelichting**	**Voorbeeld**
Embedded	Wordt gebruikt binnen de pagina waarin deze CSS code is geplaatst.	`<style type="text/css"> p { color: green; } </style>`
Inline	Is alleen van toepassing op het element waar het style attribuut is beschreven.	`<p style="color: green">`
Linked	De CSS code staat in een apart bestand waarnaar verwezen kan worden.	`<link rel="stylesheet" type="text/css" href="naam.css" media="all">`

Imported	Zelfde mogelijkheden als 'linked', maar met een kleine uitbreiding erop.	`<@import naam.css>`

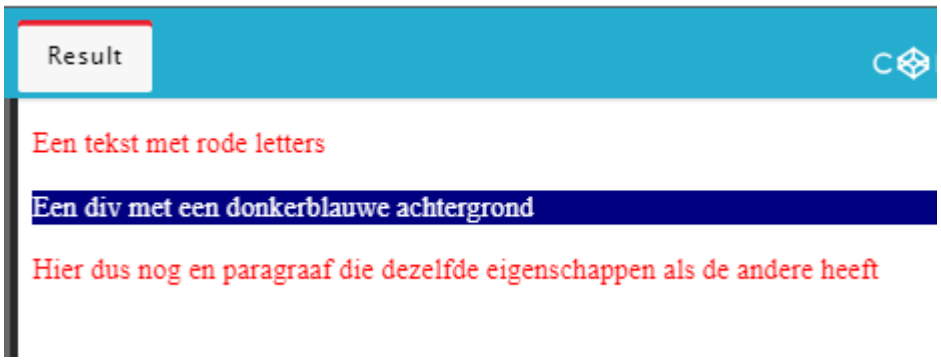
De methode die we tot dusver hebben gebruikt is de embedded style methode. Hierbij staat de CSS code bovenaan een pagina. Dit is een veelgebruikte methode wanneer er maar één of twee pagina's gebruikt worden.
Alle methoden om CSS code op te nemen worden in de volgende subparagrafen toegelicht.

2.3.1 Embedded

Tot nu toe hebben we met embedded styles gewerkt. Hierbij staan alle CSS regels in een style element binnen het head element van de pagina.

We zien hier dat de stylesheet die gedefinieerd binnen het head element, van toepassing is op het gehele document. De tekst in de twee paragrafen wordt rood gekleurd en daarnaast wordt elke div blauw van kleur met daarin witte tekst. Een voordeel van een embedded style is dat we de CSS code en de HTML structuur van een pagina in een oogopslag kunnen overzien. Echter, wanneer we deze opmaak op meerdere pagina's willen toepassen, dan zullen we dezelfde CSS code naar andere pagina's moeten copiëren. In dergelijke gevallen kunnen we beter een linked of imported stylesheet gebruiken. Dit zal verderop worden toegelicht.

```
<html>
  <head>
      <title>Embed CSS voorbeeld</title>
    <style type="text/css">
    p {
      color:            red; }
    div {
      background-color: navy;
      color:                  white; }
    </style>
  </head>
  <body>
    <p>Een tekst met rode letters</p>
    <div>Een div met een donkerblauwe achtergrond</div>
    <p>Hier dus nog en paragraaf die dezelfde eigenschappen
als de andere heeft</p>
  </body>
</html>
```



2 Opbouw van den CSS document

Tot nu toe hebben we met embedded styles gewerkt. Hierbij staan alle CSS regels in een style element binnen het head element van de pagina.

We zien hier dat de stylesheet die gedefinieerd binnen het head element, van toepassing is op het gehele document. De tekst in de twee paragrafen wordt rood gekleurd en daarnaast wordt elke div blauw van kleur met daarin witte tekst. Een voordeel van een embedded style is dat we de CSS code en de HTML structuur van een pagina in een oogopslag kunnen overzien. Echter, wanneer we deze opmaak op meerdere pagina's willen toepassen, dan zullen we dezelfde CSS code naar andere pagina's moeten copiëren. In dergelijke gevallen kunnen we beter een *linked* of *imported* stylesheet gebruiken. Dit zal verderop worden toegelicht.

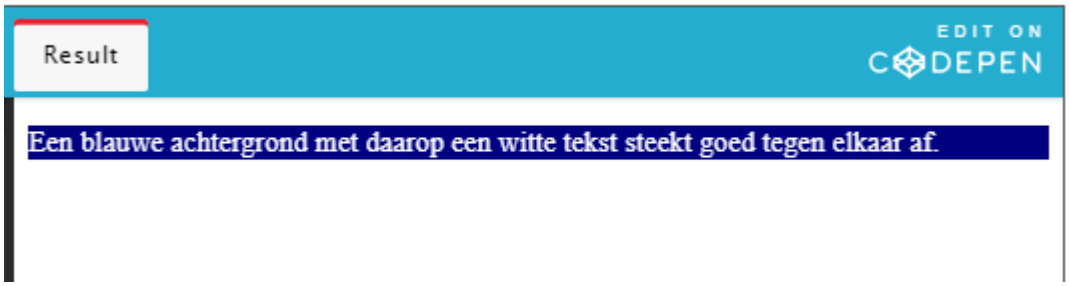
2.3.2 Inline

We kunnen het HTML style attribuut gebruiken om een inline style te definiëren. Dit attribuut kunnen we aan afzonderlijke elementen binnen de pagina toekennen om hier een afwijkende opmaak aan mee te geven.

Wij adviseren om alleen in uitzonderingssituaties gebruik van te maken. De code wordt er namelijk minder overzichtelijk door en begint meer te lijken op stijlen met HTML tags.

Hieronder staat een voorbeeld:

```
<html>
  <head>
      <title>Inline CSS voorbeeld</title>
  </head>
  <body>
    <p style="background-color: navy; color: white;">Een
blauwe achtergrond met
    daarop een witte tekst steekt goed tegen elkaar af.</p>
  </body>
</html>
```



Inline stijlen zijn alleen van toepassing op het element waar ze bij gedefinieerd zijn. Ze kunnen met uitzondering van enkele html elementen, op elk element worden toegepast. De uitzonderingen zijn: doctype, html, head en title.

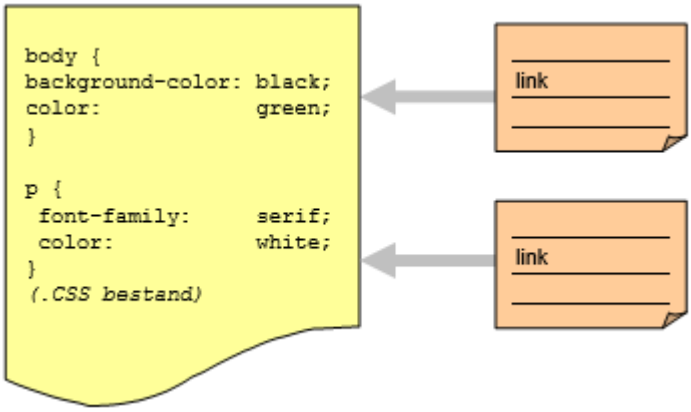
inline style syntax

```
<selector style="eigenschap: waarde">
```

* de eigenschap: waarde combinatie kan meerdere keren genoemd worden binnen het style attribuut.

2.3.3 Linked

Met linked stylesheets verwijzen we naar externe bestanden. Hierdoor kunnen alle stylesheets centraal opgeslagen worden en neemt het weinig code in beslag in de html pagina. Dit zorgt weer voor performance verbetering en betere onderhoudbaarheid. Onderstaand staat een schematisch overzicht van documenten die m.b.v. link gekoppeld zijn aan een pagina.



Bij een linked stylesheet kunnen verschillende attributen worden meegegeven, namelijk: rel, type, href en media. Het rel attribuut geeft aan wat de relatie (relation) is tussen het aan te roepen document en de webpagina. In het geval van stylesheets wordt hiervoor de notatie: rel="stylesheet" gebruikt. Het attribuut type is nauw verwant aan rel. Er wordt hierin namelijk aangegeven welk type data het aan te roepen bestand bevat. De notatie voor CSS is als volgt: type="text/css". Beide attributen dienen opgegeven te worden.

Het attribuut media geeft aan voor welk type media deze betreffende stylesheet van toepassing is.

Men kan hierbij kiezen uit:

all (default)	Om te gebruiken voor alle typen media
aural	Voor het gebruik van scherm lezers, spraak synchronistatie en andere geluidstechnische benaderingen van de pagina
braille	Wanneer we het document met een Braille apparaat willen benaderen
embossed	Wanneer we het document willen printen met een Braille print apparaat
handheld	Gebruiken voor PDA's, web telefoons en andere zeer kleine Internet apparaten
print	Gebruiken wanneer we een print voorbeeld willen zien van een webpagina
projection	Gebruiken bij een projectiescherm en bijvoorbeeld presentaties
screen	Gebruiken op normale computerschermen
tty	Gebruiken wanneer we een document willen opleveren in een vaste afstand omgeving voor bijvoorbeeld gebruik met teletype printers
tv	Gebruiken wanneer we de pagina presenteren op een televisie

De syntax voor het opnemen van linked stylesheets ziet er als volgt uit:

linked style syntax

```
<link rel="stylesheet" type="text/css" href="bestandsnaam.css">
```

Let op: deze opdracht wordt wat pittiger. We zullen hiervoor een aantal bestanden op uw eigen computer gaan aanmaken.

Maak op uw lokale computer een HTML bestand
Voeg in de HTML file minimaal de HTML, HEAD en BODY tags toe
Voeg een P element toe en typ er wat tekst in
Maak een CSS bestand waarin wij de tekst in alle P elementen een grijze kleur geven
Zorg ervoor dat wij in de HEAD van de HTML file naar het CSS bestand verwijzen wat wij ook hebben gemaakt
Maak tot slot een andere HTML file, met ook een P element erin en laat deze ook naar de CSS file verwijzen
Test uiteraard het resultaat.

2.3.4 Imported

Net zoals de linked stijl kan men bij de imported style gebruik maken van media typen. Het verschil tussen de twee methoden is de notatiewijze. Import wordt genoteerd in de style tag op een pagina.

imported style syntax

```
 <style type="text/css">

         @import url(bestandsnaam) mediatype;

</style>
```

We kunnen met de URL waarde een pad en bestandsnaam opgeven waarmee we naar een extern bestand verwijzen. De haken () om de bestandsnaam zijn verplicht.

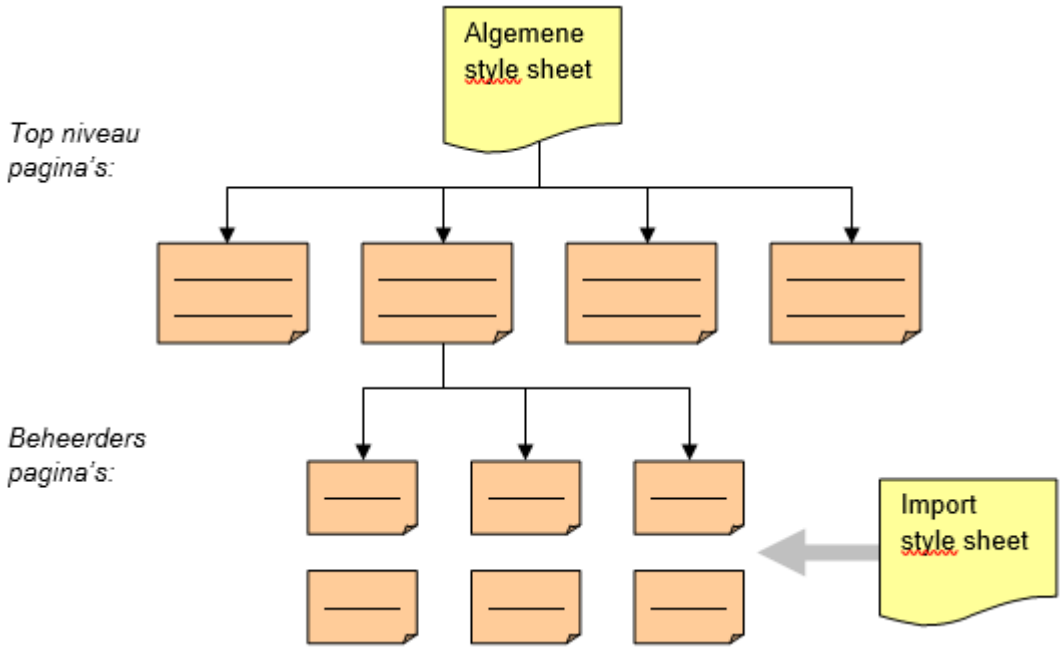
Het @import commando moet altijd het eerste commando zijn binnen de style tags óf binnen een CSS bestand.

Het is mogelijk om meerdere imports toe te passen op een pagina. De regel met de tekst @import kan dan tweemaal genoemd worden, met bij de tweede maal een andere bestandsnaam. Externe stylesheets mogen geen document lay-out bevatten zoals: <style> of
. Het voordeel van @import is dat we @import zelf wel in een externe stylesheet mogen gebruiken. We kunnen dan bijvoorbeeld de volgende code tegenkomen in een externe stylesheet:

```
<html>
      <head>
            <style>
               @import url(print.css) print;    /* Deze file
bestaat nu niet */
                  body {
                  background-color:    gray;
```

```
                color:              white; }

                p {
                font-size:          10pt;
                color :                     black; }
        </style>
  </head>
  <body>
        Invulling
        <p>Invulling 2</p>
</body>
<html>
```

Op deze manier kunnen we stylesheets modulair opbouwen. Onderstaand is schematisch weergegeven hoe een site structuur eruit kan zien.



n dit geval wordt voor de beheerders pagina's naast de algemene stylesheet een toegevoegde stylesheet gebruikt. Er is echter ook een groot nádeel aan het gebruik van @import stylesheets: lage performance! Het gebruik van een linked stylesheet met behulp van de link tag, maar ook embedded en inline styles zijn sneller te gebruiken.

Omdat @import bestanden niet parallel gedownload kunnen worden, zal de @import altijd wachten totdat de voorgaande file gedownload is. Het gebruik van meerdere <link> elementen gaat sneller, omdat deze files wél synchroon te downloaden zijn.

Wij raden aan alleen @import te gebruiken indien je specifiek wilt voortbouwen op één of meerdere stijlen uit een voorgaande stylesheet.

Wanneer meerdere stijl definities worden gebruikt, zal er één algehele virtuele stylesheet aangemaakt worden. Dit is bijvoorbeeld het geval wanneer een externe (linked/imported) en

interne (embedded) stylesheet zijn gedefinieerd en er vervolgens ook inline styles worden toegepast. De inline style heeft de hoogste prioriteit, gevolgd door de embedded en dan de external style. De inline style zal een gelijke stijl die gedefinieerd is in bijvoorbeeld een externe stylesheet, overschrijven.

2.4 Voorrangsregels

We zijn een paar keer tegengekomen dat CSS regels elkaar kunnen overlappen, of dat ze met elkaar in conflict zijn. Welke stijl in dergelijke gevallen wordt toegepast, wordt bepaald aan de hand van enkele voorrangsregels.
In het algemeen geldt: de meest specifieke regel wint. Hoe specifek een regel is kunnen we als volgt achterhalen. Voor elk van de volgende niveaus tellen we het aantal voorkomens van de betreffende selector in de regel. Uiteindelijk plakken we die cijfers aan elkaar.

a. het style="..." attribuut binnen een HTML element heeft de hoogste prioriteit
b. tel het aantal voorkomens van een id selector
c. tel het aantal voorkomens van class selectors en attribuut selectors (element[attribuut] of element[attribuut="..."])
d. tel de element en pseudo-element selectors

Stel bijvoorbeeld dat we de volgende HTML code hebben:

```
<html>
      <head>
            <style>
                              #vb1 {
              /* probeer mij! */
            }

            body div p {
              /* probeer mij! */
            }

            div.voorbeelden {
              /* probeer mij! */
            }
        </style>
  </head>
  <body>
      <div class="voorbeelden">
            <p id="vb1">Voorbeeld</p>
      </div>
  </body>
<html>
```

Welke van de volgende regels zal dan de hoogste prioriteit hebben?

```
#vb1{color: black;}

body div p{color: green;}

div.voorbeelden {color: red;}
```

In het bovenstaande voorbeeld zal de eerste regel het winnen:

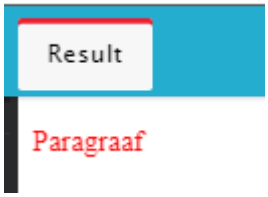
```
#vb1{color: black;}            /* a:0 b:1 c:0 d:0 -> 0,1,0,0 -> score
100 */
body div p{color: green;}      /* a:0 b:0 c:0 d:3 -> 0,0,0,3 -> score
3   */
div.voorbeelden {color: red;} /* a:0 b:0 c:1 d:1 -> 0,0,1,1 -> score
11  */
```

Het id staat net zo dicht bij de tekst als het p element maar heeft een hogere prioriteit. Wanneer de style voor het id vb1 wordt weggelaten wordt de tekst groen. De div.voorbeelden heeft wel een hogere prioriteit maar die doet er niet toe omdat de <p> dichterbij staat.

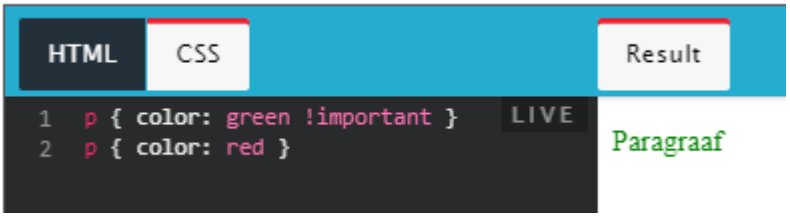
Wat nu als er twee conflicterende regels in een stylesheet staan met dezelfde specificiteit? De stylesheet wordt van boven naar beneden gelezen en toegepast, dus in dat geval wint de laatste regel.

Bijvoorbeeld:

```
<html>
      <head>
            <style>
            p { color: green }
p { color: red }
        </style>
  </head>
 <body>
  <p>Paragraaf</p>
</body>
<html>
```



In dit geval zal de tekst dus rood worden weergegeven. Om een regel toch een hogere prioriteit te geven in geval van gelijke specificiteit kan dit worden aangegeven met !important, op de volgende manier:



In dit geval zal de tekst dus in groen worden weergegeven, ondanks de tweede regel met gelijke specificiteit.

Het komt ook regelmatig voor dat dezelfde selectors vaker voorkomen in een stylesheet, maar met verschillende stijlen. In dat geval zijn de regels niet met elkaar in conflict. Nieuwe stijlregels worden gewoon aan de oude toegevoegd.

Bijvoorbeeld:

```
p { color: green }
p { font-weight:bold }
```

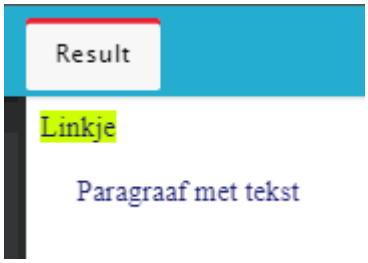
Nu wordt de tekst binnen een p element groen en vet gedrukt weergegeven.

De mogelijkheid om op deze manier nieuwe stijlen toe te voegen is erg handig. Zoals we verderop in deze cursus zullen zien kunnen we veel verschillende soorten stijlen toekennen aan één element, zoals stijlen met betrekking tot kleur, grootte, vorm en positie. Stylesheets kunnen daardoor al snel slecht onderhoudbaar worden, zoals het volgende voorbeeld laat zien:

```
a {
    text-align: left;
    color: MidnightBlue;
    text-decoration: none;
    background-color: #EFD7EA;
    .....
    .....
}
p {
    text-indent: 20px;
    text-align: justify;
    color: MidnightBlue;
    .....
    .....
}
```

Stel nu dat we later besluiten om de tekstkleur toch te veranderen, dan zouden we in dit geval alle selectors moeten nalopen op de style color. Er is in dit geval veel voor te zeggen om elementen met gelijksoortige stijlen samen in een regel vast te leggen, en de overige regels daaronder toe te voegen, zoals in onderstaand voorbeeld:

```
<html>
      <head>
            <style>
            a, p { color: MidnightBlue; }
            a { background-color: #CCFF00; }
            a {
                  text-decoration: none;
            }
            p {
                  text-indent: 20px;
            }
        </style>
  </head>
 <body>
  <a href="#">Linkje</a>
<p>Paragraaf met tekst</p>
</body>
<html>
```



Als we later een andere tekstkleur willen, dan hoeven we dat in dit geval maar op één plek te aan te passen.

2.4.1 Commentaar in Stylesheet

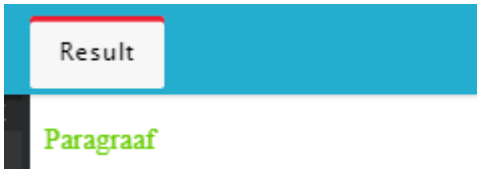
Commentaar wordt in CSS code voor verscheidene doeleinden gebruikt. Ten eerste voor ontwikkelaars en vormgevers die nadat de code geschreven is, iets moeten aanpassen of de code overgedragen krijgen. Daarnaast wordt commentaar ook regelmatig gebruikt als herinnering voor de schrijver van de code.

Er zijn twee manieren om commentaar op te nemen in CSS (net als in HTML). Bekijk de onderstaande tabel voor toelichting.

Commentaar notatie	Toelichting
/ * hier staat code */	Deze combinatie van een slash en ster (en vise versa), wordt gebruikt om commentaar over meerdere regels in te voegen.
// hier staat code	De dubbele slash wordt gebruikt om commentaar per regel in te voegen.

In de onderstaande code treft u beide twee commentaarvormen aan.

```
<html>
      <head>
            <style>
            /* Onderstaand zien wij hoe we commentaar
kunnen toevoegen: */

p {
       color:     #66CC00;    // kleur medium groen
       font-size: 80%; }

      #alinea_voorbeeld {
       color:     green;
       font-size: 80%;  /* lettertype is nu 8pt
                     kleur medium groen */
}
        </style>
  </head>
 <body>
<p>Paragraaf</p>
</body>
<html>
```



In de praktijk worden de kleurcodes vaak op deze manier toegelicht. Meestal wordt de /* */ combinatie gebruikt wordt omdat deze het meest veelzijdig is.

2.5 Elementsoorten

We kunnen HTML elementen opsplitsen in vier mogelijke groepen. Een element is replaced of non-replaced en het kan een block element of een inline element zijn.

Hoe een element in de browser wordt weergegeven hangt niet alleen af van de stylesheet, maar ook van de groep waar het element toe behoort.

Replaced HTML elementen zijn elementen die verwijzen naar grafische componenten, zoals plaatjes, knoppen of invoervelden. Runtime wordt een replaced element vervangen door de grafische component. Dergelijke elementen hebben geen tekst tussen de tags. Sommige elementen kunnen echter wel een tekstwaarde bevatten, zoals het input element van het type text.

Bij replaced elementen wordt in een HTML document de inhoud vervangen door iets anders. De eigenlijke beschrijving van een replaced element bevat dus geen inhoud, alleen een verwijzing. Een replaced element heeft alleen een openingstag (met eventueel aan het einde een /). Een input tag is bijvoorbeeld een replaced element, omdat deze kan zorgen voor onder andere een text input, checkbox of radio button.
Ook een image (img) is een replaced element. Er is geen </img> tag nodig om het element af te sluiten, de volgende code voldoet:

```
<img src=”bestandsnaam.jpg” alt=”plaatje” />
```

De meeste HTML elementen zijn van het type non-replaced, zoals , en <p>. Onderstaand volgt een aantal voorbeelden van elementen en de bijbehorende typen.

Replaced: img, input, hr, br

Non-replaced: div, li, p, table, form, pre, h1..h6, span, a, strong, em

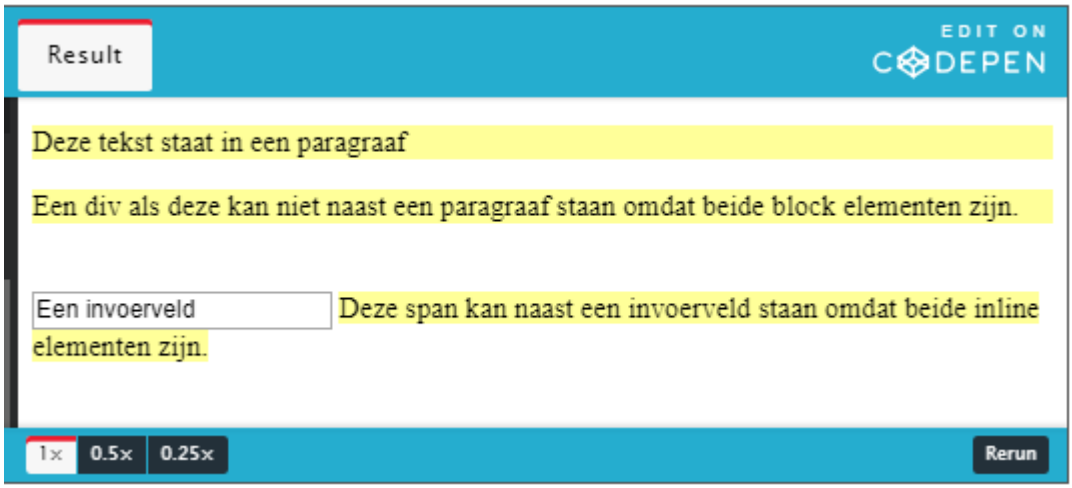
Replaced elementen hebben intrinsieke dimensies, zoals breedte en hoogte: we hoeven deze dimensies daarom niet te bepalen met een stylesheet, hoewel we ze in veel gevallen wel kunnen aanpassen.

Block elementen beginnen met een nieuwe regel en nemen de hele beschikbare breedte in gebruik. Meerdere block elementen achter elkaar worden in de pagina als blokken onder elkaar geplaatst.

Block elementen creëren een blok om zichzelf heen waarmee het omliggende element (de zogeheten ‘parent’) wordt opgevuld. Meer over parent en child elementen leest u verderop in dit hoofdstuk. Hieronder volgt een aantal voorbeelden van elementen en de bijbehorende typen.
Block: div, li, p, hr, table, form, pre, h1..h6

Inline: br, span, a, strong, input, em, img

```
<html>
  <head>
    <title>Block en inline elementen</title>
    <style type="text/css">
       p,div,span {
         background-color:   rgb(255,255,153);
            }

    </style>
  </head>

  <body>

     <p>Deze tekst staat in een paragraaf</p>
     <div>Een div als deze kan niet naast een paragraaf
staan omdat beide block elementen zijn.</div>

     <br><br>

     <input type="text" name="txtVeld" value="Een
invoerveld">
       <span> Deze span kan naast een invoerveld staan
omdat beide inline elementen zijn.</span>

  </body>
</html>
```



U kunt hier zien dat block elementen zoals div en p onder elkaar worden geplaatst op de pagina. De inline elementen input en span worden naast elkaar geplaatst.
Inline elementen nemen de ruimte in van de inhoud (enige uitzondering hierop is het input element). Block elementen nemen de gehele breedte in van het omliggende element (de parent. Daarom kan er naast een block element ook geen ander element staan. Er is daar geen ruimte voor.

Deze eigenschappen zijn met behulp van CSS aan te passen, waardoor bijvoorbeeld wel twee block elementen naast elkaar geplaatst kunnen worden. We komen daar verderop in deze cursus op terug.

en DIV mag andere block elementen bevatten. Een P mag geen andere block elementen (ook geen andere P) bevatten.

Afhankelijk van het soort element – inline of block, replaced of non-replaced – kunnen toegekende CSS stijlen anders uitpakken. We gaan daar in deze cursus niet dieper op in. Wel is het van belang te weten dat inline replaced elementen zich gedragen als een enkel karakter van een tekst: de hoogte van de gehele regel wordt aangepast aan de hoogte van het replaced element.

In HTML kunnen inline elementen geplaatst zijn in block elementen maar niet andersom. Echter, in CSS kunnen we het type van een element (block of inline) aanpassen met het display attribuut. Onderstaand staat een voorbeeld.

```
div { display:          inline; }
span { display:   block;}
```

Dit zorgt ervoor dat de span een block element wordt en een onzichtbaar block kader om zich heen krijgt, terwijl de div een inline element wordt, dus deel uitmaakt van een regel. We kunnen een dergelijk effect echter niet creëren met het verplaatsen van HTML tags. We kunnen dus niet zeggen:
`<span>begin span <div>Dit is de div in een span, kan dat?</div> eind span</span>` Dit mag dus wel, maar dit is geen goede HTML code en geeft ook niet het gewenste resultaat. We zien dan dat het onderstaande het resultaat hiervan is:

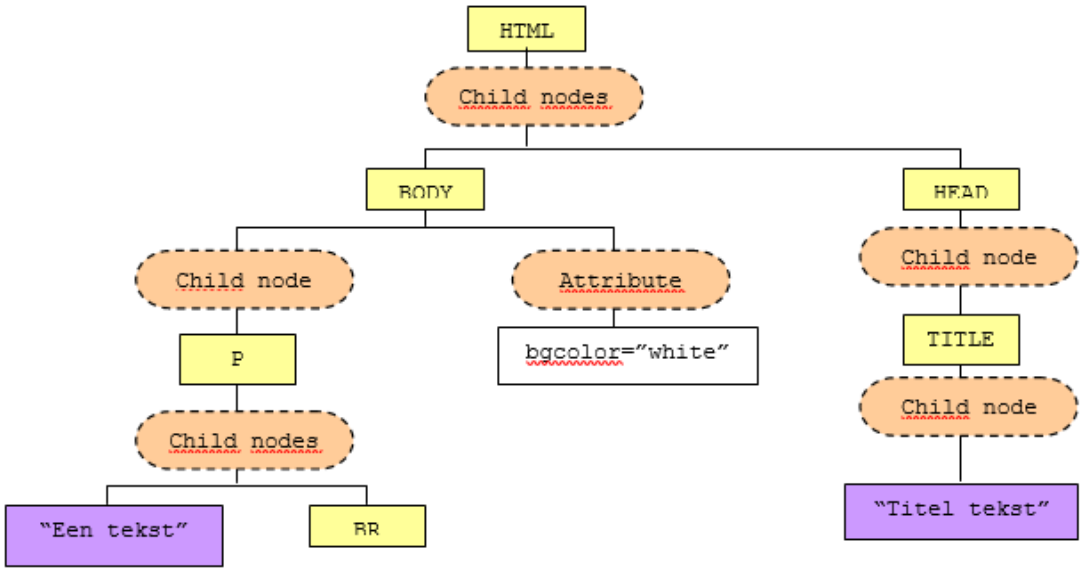
begin span
Dit is de div in een span, kan dat?
eind span

De span vult één regel. Vervolgens staat de div buiten de span weergegeven (omdat deze zelf een nieuwe regel en regelovergang nodig heeft. Dan zien we op de regel daarna het restant van de span weer op een aparte regel. Dit is en blijft dezelfde span. Hij is dus niet opgesplitst.

2.6 Overerving van parent naar child

In een HTML document is achter de schermen een boom structuur aanwezig. Deze wordt ook wel de DOM tree genoemd. DOM staat hier voor Document Object Model. Het Document Object Model is een taalonafhankelijke standaard voor het weergeven van de structuur van onder andere HTML en XML documenten. Het DOM model (niveau 1) wordt bijvoorbeeld door JavaScript gebruikt om een document dynamisch te kunnen analyseren en wijzigen. In onderstaand diagram is weergegeven hoe een HTML structuur eruit ziet volgens het Document object model.

2 Opbouw van den CSS document



De met oranje weergegeven velden (de HTML tags), zijn zogeheten nodes. Wanneer een node onder een andere node hangt, dan is dit een subelement van de daarboven gelegen node. De paars weergegeven velden (links en rechts) zijn text nodes. Het middelste witte veld is een attribuut van de *body node*, die een element is van de *html* node (dus ook wel: het *document* object).

CSS stijlen die van toepassing zijn op parent elementen, zijn in de meeste gevallen automatisch ook van toepassing op de onderliggende child elementen (child nodes). Wanneer we dit voor een eigenschap expliciet willen instellen gebruiken we de waarde *inherit*. Hiermee geven we van een element aan dat de waarde voor de betreffende eigenschap overgenomen moet worden van het parent element.

We zien hier een formulier met achtergrondkleur, twee invoervelden en een knop, dit bestand gaan wij vervolgens qua CSS code aanpassen:

```
<html>
      <head>
            <style>
            body    {
              margin:0;
              padding:0;
            }

            div     {
              color:#AA7733;
              background-color:#FFEECC;
              width:300px;
              margin:10px;
              padding:10px;
              height:auto;
            }

            input   {
```

```
            display:block;
            margin:10px;
          }

          #submit {
            display:inherit;
            margin-left:100px;
            margin-bottom:-10px;
          }

          label   {
            display:block;
            float:left;
            width:100px;
          }
        </style>
  </head>
 <body>
   <div>
   <form>
    <label for="naam">Naam: </label><input type="text"
id="naam" name="naam"></input>
    <label for="mail">Email adres: </label><input
type="text" id="mail" name="mail"></input>
    <input type="submit" id="submit" value="Verstuur"/>

   </form>
  </div>
  </body>
<html>
```

Voeg nu de volgende code toe en bekijk het resultaat:
input { color:inherit; background-color:inherit; }

2 Opbouw van den CSS document

2.7 Samenvatting

Een CSS regel is opgebouwd uit een selector en een declaratie. Een declaratie is vervolgens op te splitsen in een eigenschap en een waarde. Een selector verwijst naar een element in de HTML code en een declaratie zegt wat we aan dat element willen wijzigen. Een inline stylesheet kan alleen worden toegepast op een HTML tag. Een externe stylesheet kan toepassing hebben op zowel HTML tags, classes en id's. Binnen een HTML document is een structuur aanwezig. Elementen die binnen andere elementen zijn geplaatst noemen we nodes of child elementen. Logischerwijs zijn dan de elementen die juist andere elementen onder zich hebben de parent elementen.

3 Lettertypen en tekstopmaak

3.1 Inleiding

In de komende paragraaf zullen we verschillende tekstuele CSS eigenschappen bespreken die we kunnen toepassen op elementen. We behandelen eerst hoe we teksten kunnen uitlijnen en aankleden. Vervolgens bespreken we hoe we lettertypen geheel naar wens kunnen wijzigen. En tot slot gaan we kijken naar het gebruiken van web fonts.

- Kunnen toevoegen en aanpassen van lettertypen binnen een HTML document
- Webfont's kunnen toepassen op elementen
- Gebruik kunnen maken van icon fonts

3.2 Lettertypen

Tekst is nog altijd de meest gebruikte methode om informatie over te brengen op het internet. Vandaar dat het van belang is dat dit goed gebeurt. Naast inhoudelijke criteria, zoals een correcte spelling en heldere formulering, is speciaal voor internet pagina's de vormgeving van belang. Meer dan bij papieren informatie geldt voor web pagina's immers dat het moeilijk is de aandacht van de lezer vast te houden. Een duidelijke vormgeving helpt daarbij. We bespreken in deze paragraaf de tekst eigenschappen font-family, font-size, font-weight en font-style.

3.2.1 Font family

Eén van de eerste keuzes die we bij de vormgeving van tekst maken is het lettertype. Voor papieren teksten kunnen we een groot aantal verschillende lettertypen gebruiken. Voor webpagina's zullen we ons moeten beperken: de gebruikte lettertypen moeten al op de PC van de gebruiker geïnstalleerd zijn, en we weten immers van te voren niet welke lettertypen dat zijn.
Het is daarom verstandig om eventuele alternatieve lettertypen (of: fonts) op te geven, naast het lettertype van onze voorkeur.

Fonts worden ingedeeld in vijf groepen

Soort font	Voorbeeld
Serif font:	Georgia
Sans-serif font:	Arial
Monospace font:	Courier
Cursive font:	Comic Sans
Fantasy font:	□□□□□

Wanneer we een website bekijken met een fantasy font erin gedefinieerd, dan wordt dit font alleen correct weergegeven indien dat font ook op de client computer geïnstalleerd is.

Anders wordt een standaard lettertype weergegeven zoals ‘Times new roman’. Hieronder zullen de vijf fontsoorten nader worden toegelicht.

Serif fonts
Dit zijn de fonts die een schreef hebben en een verschillende letterbreedte. Serifs (schreven) zijn kleine decoraties die bijvoorbeeld op de volgende letters zitten: l, T, R en I. Het zijn de kleine uitstekende puntjes in dit geval, die een letter sierlijker maken. De letterbreedten verschillen bij Serif fonts per letter. Zo neemt de letter i binnen een tekst minder ruimte in dan een m. Voorbeelden van serif fonts zijn: Georgia en Times.

Sans-serif fonts
Ook deze fonts hebben een verschillende letterbreedte, maar ze zijn schreefloos. Ze bevatten dus geen kleine decoraties aan de letters. Voorbeelden van sans-serif fonts zijn: Arial, Geneva, Helvetica en Verdana.

Monospace fonts
Monospace fonts hebben een vaste letterbreedte. Elk karakter neemt exact evenveel ruimte in beslag. Dit soort fonts kan serifs bevatten. Voorbeelden van monospace fonts zijn: Andale Mono, Courier en Courier New.

Cursive fonts
Dit type font is vaak in overeenstemming met het menselijk handschrift. De ronde vormen en decoraties zijn regelmatig extremer dan bij een serif lettertype. Voorbeelden van cursive fonts zijn: Author en Comic Sans.

Fantasy fonts
Deze lettertypen zijn niet onder te brengen in de andere groepen. Hieronder vallen bijvoorbeeld, zoals de naam ook zegt, lettertypen die te maken hebben met fantasie (verhalen, films, enzovoorts). Enkele voorbeelden van deze fonts zijn: Klingon, Western en Woodblock.

In CSS gebruiken we de eigenschap font-family om een lettertype aan te geven. Ook de eventuele alternatieve lettertypen kunnen we hierbij aangeven; gescheiden door komma’s. CSS zal proberen het eerst beschreven font te gebruiken. Mocht dat niet lukken dan gaat CSS naar het tweede beschreven font, enzovoorts. De syntax voor het instellen van lettertype is als volgt:

font-family syntax

syntax: selector { font-family: sleutelwoord; }

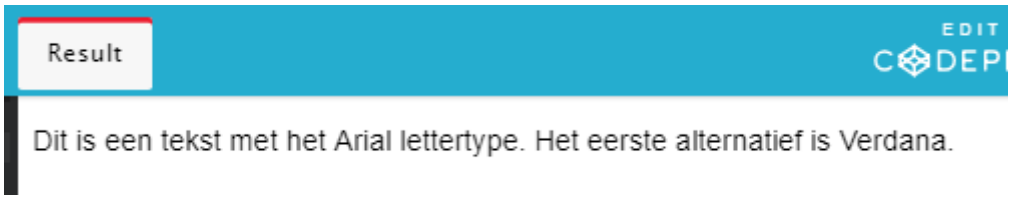
sleutelwoord: [serif | sans-serif] | <fontnaam> | inherit

Let op: 'Inherit' staat voor overerven. Dit sleutelwoord zullen we nog vaker tegenkomen in CSS en deze houdt in dat de waarde van de eigenschap wordt overgenomen van het parent element.

Stel, we willen een arial lettertype instellen, met als alternatief verdana. Maar wat te doen als een gebruiker deze fonts beide niet op zijn computer heeft staan? We kunnen dan als laatste alternatief een groepnaam met fonts opgeven. Bijvoorbeeld:

3 Lettertypen en tekstopmaak

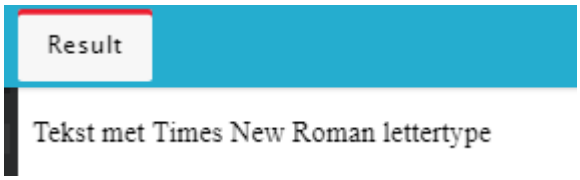
```
<html>
      <head>
    <style>
      p {
  font-family:    arial, verdana, sans-serif;
}
      </style>
  </head>
 <body>
    <p>Dit is een tekst met het Arial lettertype. Het
eerste alternatief is Verdana.</p>
  </body>
<html>
```



Dit stelt dus in dat paragrafen standaard het lettertype arial moeten gebruiken. Is dat niet aanwezig op de computer van de gebruiker, dan zal Verdana gebruikt worden. Indien dat font ook niet aanwezig is, zal één van de aanwezige sans-serif fonts gebruikt worden.

Een groepnaam zoals sans-serif opgeven is altijd veiliger dan het opgeven van individuele lettertypen. Wanneer we een lettertype willen opgeven dat uit meerdere woorden bestaat zoals Times New Roman, dan zullen we die woorden gezamenlijk tussen apostrofs (één kommaatje in de lucht, niet te verwarren met het aanhalingsteken ") moeten zetten. Anders zal CSS het niet goed oppikken.

```
<html>
      <head>
    <style>
      p {
  font-family:    'times new roman', serif;
}
      </style>
  </head>
 <body>
   <body>
      <p>Tekst met Times New Roman lettertype</p>
  </body>
<html>
```



3.2.2 Font size

Font-size is de eigenschap die we instellen wanneer wij de afmeting van teksten willen aanpassen. Voor het instellen van deze eigenschap kunnen we een onderscheid maken tussen beschrijvende en niet beschrijvende meetvormen en tussen absolute en relatieve meetvormen.

Beschrijvende meetvormen
Wanneer we de font-size willen instellen zonder een standaard maat te gebruiken, kunnen we beschrijvende meetvormen gebruiken. Beschrijvende meeteenheden geven een meeteenheid aan in woorden. Deze zijn alleen van toepassing op lettertypen. Daarnaast is de werkelijke afmeting op het scherm afhankelijk van de CSS en browser versie.
font-size syntax (beschrijvend)

syntax: selector { font-size: sleutelwoord; }

sleutelwoord: <grootte>| inherit

default: medium

Op de plaats van *sleutelwoord* komt de beschrijvende waarde van de grootte. Dit kan bijvoorbeeld*: x-small, medium, smaller* of *larger* zijn. De waarden x-small en medium vallen onder de absolute meetvormen. Er wordt hiervoor namelijk niet gekeken naar bijvoorbeeld de font-size van de parent. Omdat hierbij het berekenen van de juiste afmetingen ten opzichte van elkaar lastig is, worden de absolute meetvormen zelden gebruikt. Relatieve beschrijvende maten zijn praktischer en komen we daarom vaker tegen.

De waarden *smaller* en *larger* zijn relatieve meeteenheden. Voor het stellen van deze maten wordt uitgegaan van de grootte van de tekst in het parent element. In de tabel hieronder ziet u welke absolute en relatieve maten u kunt gebruiken in CSS.

Absolute beschrijvende meetvormen:	xx-small, x-small, small, medium, large, x-large en xx-large
Relatieve beschrijvende meetvormen:	smaller of larger

```
<html>
      <head>
    <style>
            span {
                  font-size:  smaller;
                  }
      </style>
  </head>
 <body>
    <p>Dit is het parent paragraaf block element.
    <span>En dit is een child inline span element, met een
kleinere lettergrootte</span>
    </p>
  </body>
<html>
```

3 Lettertypen en tekstopmaak

Zoals u kunt zien in de browser, is het resultaat dat de tekst in het span element, één maat kleiner is dan het parent p element.

Niet beschrijvende meetvormen
Deze meetvormen hebben een numerieke waarde en zijn niet in woorden te beschrijven.
De syntax hiervoor is:

font-size syntax (niet beschrijvend)

syntax: selector { font-size: sleutelwoord; }

sleutelwoord: <lengte> | <percentage>| inherit

default: medium

Er zijn verschillende niet beschrijvende meeteenheden te gebruiken voor fonts, maar ook voor andere CSS eigenschappen die later in de training aan bod komen.

Deze niet beschrijvende eenheden zijn op te delen in absolute en relatieve maten. Deze zijn in de onderstaande tabellen weergegeven en toegelicht.

Absolute meetvormen	**Omschrijving**
Centimeters (cm)	Een standaard. Eén centimeter is gelijk aan 0.39 inch of aan tien millimeter.
Inches (in)	Een standaard Engelse meeteenheid. Eén inch is gelijk aan 2.54 centimeter.
Millimeters (mm)	Een standaard. Tien millimeter is gelijk aan één centimeter.
Pica's (pc)	Een standaard printmaat. Eén pica is gelijk aan twaalf punten.
Punten (pt)	Een standaard printmaat. Eén punt is gelijk aan 1/72e inch. Punten zijn niet erg betrouwbaar voor het web omdat het aantal punten kan variëren per lettertype.
Pixels (px)	Een standaard schermmaat. Eén pixel staat gelijk aan één echte pixel (stipje) op het scherm. Px is het beste om te gebruiken binnen CSS. Alleen wanneer er gewerkt wordt in verschillende schermresoluties, is deze meetvorm meer relatief.

Relatieve meetvormen	**Omschrijving**
Ems (em)	Een standaard printmaat. *Ems* is relatief ten opzichte van de breedte van een hoofdletter M karakter van de overgeërfde fontgrootte.

x-height (ex)	Een standaard printmaat. *x-height* is relatief ten opzichte van de hoogte van een klein x karakter van de overgeërfde fontgrootte.
Percentage (%)	Een procentueel relatieve waarde ten opzichte van de overgeërfde fontgrootte.
View-width (vw)	Percentage van de breedte van de viewport (zichtbare deel webpagina).

Om een beeld te geven van grootte van de bovenstaande absolute meetvormen, hebben we een bestand aangemaakt met voorbeelden. We hebben ervoor gekozen de meest gebruikte vormen te tonen. Overigens hoeven wij niet altijd met hele waarden te rekenen.

```
<html>
      <head>
    <style>
            table {
  cellspacing: 2px;
}

/* alle cellen met een id ingesteld lijnen wij uit */
td[id] {
  text-align: center;
}

#punten {
  font-size: 8pt;
}

#picas {
  font-size: 8pc;
}

#inches {
  font-size: 8in;
}

#centimeters {
  font-size: 8cm;
}

#millimeters {
  font-size: 8mm;
}

#pixels {
  font-size: 8px;
}

#beschrijvend {
  font-size: medium;
}

#ems {
```

```
  font-size: 8em;
}

#xheight {
  font-size: 8ex;
}

#percentage {
  font-size: 80%;
}

#viewwidth {
  font-size: 8vw;
}
      </style>
  </head>
 <body>
    <table>
      <tr>
        <th>Meeteenheid</th>
        <th>Voorbeeld</th>
      </tr>
      <tr>
        <td>Punten (pt)</td>
        <td id="punten">8 punten</td>
      </tr>
      <tr>
        <td>Pixels</td>
        <td id="pixels">8 pixels</td>
      </tr>
      <tr>
        <td>Beschrijvend</td>
        <td id="beschrijvend">Medium</td>
      </tr>
      <tr>
        <td>Ems printmaat (em)</td>
        <td id="ems">8 em</td>
      </tr>
      <tr>
        <td>Percentage (%)</td>
        <td id="percentage">80%</td>
      </tr>
      <tr>
        <td>View width (vw)</td>
        <td id="viewwidth">8 vw</td>
      </tr>
    </table>
  </body>
<html>
```

Resultaat:

3 Lettertypen en tekstopmaak

Meeteenheid	Voorbeeld
Punten (pt)	8 punten
Pixels	8 pixels
Beschrijvend	Medium
Ems printmaat (em)	8 em
Percentage (%)	80%
View width (vw)	8 vw

3.2.3 font-weight

De eigenschap font-weight geeft aan of tekst vet gedrukt is en in welke mate.
font-weight syntax

syntax: selector { font-weight: sleutelwoord; }
sleutelwoord: <numeriek: 100–900> | light | normal | bold | bolder| inherit
default: normal

Men kan als sleutelwoord kiezen tussen een numerieke of non-numerieke notatievorm. De numerieke notatie loopt van 100 t/m 900 en gaat in stappen van 100. Voor de non-numerieke notatie gebruiken we: *normal*, *bold*, *bolder* en *lighter*.

```
<html>
      <head>
    <style>
            p {

            }
      </style>
  </head>
 <body>
  <p>Paragraaf met tekst
    <span>Ook zit deze spantekst erin</span>
  </p>
</body>
<html>
```

. Bekijk de bovenstaande code en zorg ervoor dat de tekst in de paragraaf bold wordt weergegeven.

Valt u iets op nadat u dat heeft gedaan? Het span element is nu ook bold weergegeven. Om dit effect voor de span terug te draaien, moeten we een lagere waarde voor de span opgeven. Geef nu de span een numerieke weight waarde van 200 en bekijk het resultaat.

3.2.4 Font-style

Met font-style kunnen we een tekst normaal, cursief of schuingedrukt weergeven.

font-style syntax

syntax:	selector { font-style: sleutelword; }
sleutelwoord:	normal | italic | oblique| inherit
default:	normal

Hierbij spreekt het voor zich dat normal de standaard instelling voor het sleutelwoord is. Met de tekst italic worden de tekens niet alleen schuin gedrukt, ze worden in geval van serif lettertypen ook extra decoratief. Dit in tegenstelling tot de oblique weergave die er simpelweg voor zorgt dat de tekst schuin gedrukt wordt weergegeven. .

In veel lettertypen is er geen verschil tussen oblique en italic font styles. Het komt ook voor dat oudere browsers de ondersteuning niet goed bieden.

3.2.5 Web fonts

De meeste browsers zijn in staat gebruik te maken van de CSS eigenschap font-face. Deze maakt het mogelijk om met lettertypen te werken die niet op de pc van de gebruiker geïnstalleerd staan. In plaats daarvan wordt het font-bestand op de server gezet.

Met behulp van de selector @font-face geven we in de stylesheet aan hoe dit lettertype gedefinieerd is. De apenstaart (@) notatie wordt binnen CSS vaker gebruikt om een stuk code te maken waar wij ergens anders in onze CSS code naartoe kunnen verwijzen.

Helaas verwachten verschillende browsers verschillende soorten lettertypen:

rowser	**Ondersteunde fonts**
Internet Explorer 4+	Embedded Open Type (eot)
Firefox 3.5+	TrueType, OpenType, Web OpenType Format (ttf, otf, woff) De laatste (woff) wordt sinds 3.6 ondersteund.
Chrome 4.0+	TrueType, OpenType, Web OpenType Format (ttf, otf, woff) De laatste (woff) wordt sinds versie 6 ondersteund.
Opera 10+	TrueType, OpenType, SVG (ttv, otf, svg)
Webkit/Safari 3.1+	TrueType, OpenType, SVG (ttf, otf, svg)

Ontwikkelaars hebben een methode gevonden om aan al deze browsers tegemoet te komen. We moeten dan wel aan alle versies van een font zien te komen. Op het internet zijn zeer veel gratis lettertypen te vinden. Er zijn ook sites, zoals Font Squirrel (http://www.fontsquirrel.com/), waarmee van een fontbestand de overige soorten lettertypen gegenereerd kunnen worden. Hierbij wordt ook de benodigde CSS code gegenereerd om van deze lettertypen gebruik te kunnen maken.

```
<html>
      <head>
    <style>
            @font-face {font-family: 'Hobbiton Brush';
            src: url('https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/hobbitonbrushhand.ttf');
             src: local('#'), url('https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/hobbitonbrushhand.ttf')
format('truetype')
                font-weight:normal;
                font-style:normal;
            }

            p {
              font-family: 'Hobbiton Brush', serif;
              font-size:   25pt;
            }
      </style>
  </head>
 <body>
  <p>Paragraaf tekst in de stijl van Lord of the Rings</p>

</body>
<html>
```

NB wanneer u geen bijzonder letterype zie, voer dit voorbeeld dan uit in codepen.

In de browser ziet u een tekst staan in een alternatief lettertype: 'Hobbiton Brush'.
We gebruiken de @font-face selector om het nieuwe type te declareren. We geven hiermee aan dat de font-family 'Hobbiton Brush' gebruik maakt van de genoemde bestanden en dat het een type is met een normale font-weight en font-style. In dit geval zijn de lettertypen op een andere server geplaatst. In de praktijk komen deze lettertypen vaak op dezelfde server te staan (in een map 'fonts' bijvoorbeeld).

IE9 en Firefox verwachten dat de fonts op hetzelfde domein staan als de website. Verwijzingen naar een ander domein zijn dus pas aan te raden vanaf IE10 en nieuwere FireFox versies.

We gaan hier niet dieper in op de betekenis van de rest van de code: de gegenereerde code en uitleg daarover kunt u vinden op de eerder genoemde website.

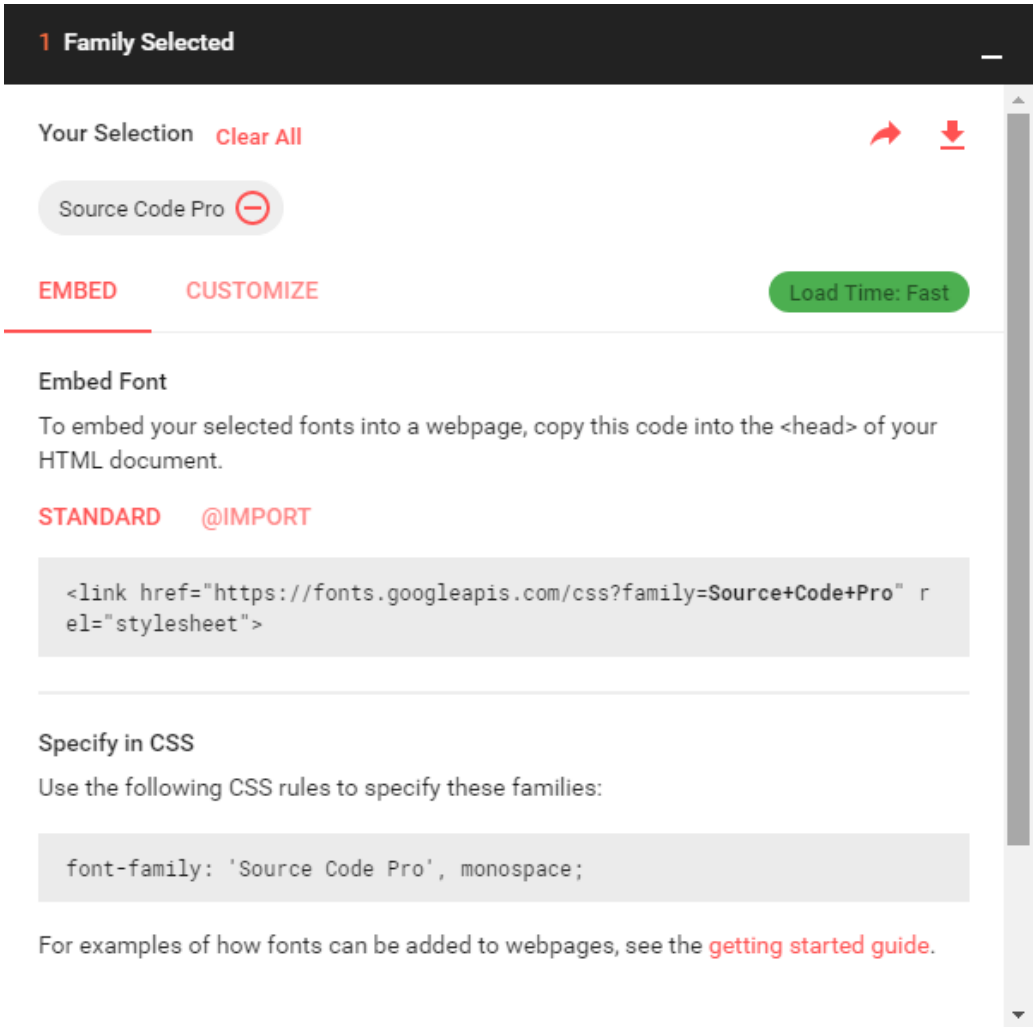
Verderop in de stylesheet kunnen we naar dit lettertype verwijzen. In het voorbeeld hierboven in de stijl voor het P-element.
Zorg er wel voor dat u alternatieve lettertypen opgeeft voor oudere browsers die @font-face niet ondersteunen. In dit geval is de groep serif als alternatief gebruikt.
Indien u niet uw eigen lettertypen wilt hosten op uw website, maar u toch graag bijzondere lettertypen wilt gebruiken, dan is de Google Font service wellicht iets voor u. Google heeft op haar servers een groot aantal lettertypen staan die wij kunnen gebruiken.

3 Lettertypen en tekstopmaak

Hoe gebruiken wij een lettertype van Google Fonts? Ga naar: https://fonts.google.com/, kies daar een font uit de (gigantische) lijst door op het (+) teken bij het lettertype (of de lettertypen indien u er meer wenst te gebruiken) zodat ze in een soort winkelwagen terecht komen.

Klik vervolgens onderin beeld op deze balk om de zogenaamde lettertypen winkelwagen te openen:

Nu is er zichtbaar hoe wij het lettertype kunnen toevoegen aan onze pagina:



We zullen zoals bovenstaand te zien is moeten verwijzen naar een externe stylesheet op de Google website midels:
`<link href="https://fonts.googleapis.com/css?family=Source+Code+Pro" rel="stylesheet">`
En vervolgens kunnen wij het lettertype toepassen op een element middels de bekende font-family eigenschap:
`font-family: 'Source Code Pro', monospace;`
N.b.: 'monospace' is hier de alternatieve lettertype categorie indien het font niet meer beschikbaar zou zijn.

3.2.6 icons

Indien we een eenvoudig pictogram of icoon willen weergeven, kunnen we in veel gevallen ook gebruik maken van een font in plaats van een afbeelding. Hoe gaat dat in zijn werk? Eerst kiezen we een lettertype uit via (bijvoorbeeld) We Love Icon Fonts

1. Kies hier een lettertype uit waar het gewenste icoon in zit
2. Klik op ‘(hartje) Add’ en vervolgens onderin beeld op: ‘Use X’, waarij ‘X’ staat voor het aantal lettertypen dat wij hebben geselecteerd.
3. Er verschijnt nu een blok met CSS code. De import regel dienen we in onze CSS file te plaatsen om de lettertype(n) te kunnen gebruiken. Het toepassen van deze lettertypen gaat echter iets anders dan dat we gewend zijn.

4. Geef de elementen waarbinnen we (direct) de iconen willen gebruiken een class-attribuut met als waarde: ‘naam_lettertype-gekozen_icoon’. Hiernaar kunnen wij vervolgens vanuit onze CSS code verwijzen met de code:

```
[class*="naam_lettertype-"]:before {
        font-family: 'naam_lettertype, sans-serif;
}
```

Let op:
Het gebruik van icon fonts in SVG formaat, wordt niet meer ondersteund in Google Chrome sinds versie 38. Hierdoor zal in Chrome een vierkantje getoond worden in plaats van het juiste icoon.

```
<html>
      <head>
    <style>
            @import url(https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/entypo.css);

                  /* entypo */
                  [class*="entypo-"]:before {
                    font-family: 'entypo', sans-serif;
                  }
      </style>
  </head>
 <body>
    <h3>Icon Fonts</h3>
    <div>
      <p>Deel icoon:<span class="entypo-export"></span></p>
      <p>Winkelwagen icoon:<span class="entypo-
basket"></span></p>
      <p>Bijlage icoon:<span class="entypo-
attach"></span></p>
      <p>Duim omhoog icoon:<span class="entypo-thumbs-
up"></span></p>
    </div>
  </body>
<html>
```

Icon Fonts

Deel icoon:

Winkelwagen icoon:

Bijlage icoon:

Duim omhoog icoon:

Voor meer voorbeelden, bekijk de http://weloveiconfonts.com/ webpagina.

3 Lettertypen en tekstopmaak

3.3 Tekst eigenschappen

De tekstindeling van een pagina betreft de uitlijning en plaatsing van tekens en hele teksten. Het plaatsen van blokken met tekst (zoals div's en span's), wordt in het laatste hoofdstuk besproken. In deze paragraaf zullen we de volgende tekst eigenschappen bespreken: text-indent, text-align, vertical-align, letter-spacing, word-spacing, line-height en text-decoration.

3.3.1 Inspringen (text-indent en margin)

Het inspringen van tekst kan op twee manieren. De text-indent eigenschap is de meest gebruikte methode. Hiermee springt de eerste regel tekst van een block element in. Willen we hetzelfde effect creëren voor een inline element, dan kan dit alleen door het instellen van de padding of margin. Deze begrippen komen in een later hoofdstuk uitgebreid aan bod. We kunnen de text-indent waarden op verschillende manieren aangeven.

text-indent syntax

syntax:	selector { text-indent: sleutelwoord; }
sleutelwoord:	<lengte> | <percentage>| inherit
default:	0

```
<html>
      <head>
    <style>
                    div { width:   300px; }
       #p1 { text-indent: 50px; }
       #p2 { margin-left:50px; }
       #s1 { text-indent: 50px; }
       #s2 { margin-left: 50px; }
      </style>
  </head>
  <body>
    <div>
      <p id="p1">Deze paragraaf staat in een div van 300
pixels breed. De tekst is ingesprongen met 50 pixels.</p>
      <p id="p2">Deze paragraaf staat in een div van 300
pixels breed. De tekst heeft een linker marge van 50
pixels.</p>

      <span id="s1">Dit span element staat in een div van
300 pixels breed. De eigenschap text-indent heeft op dit
inline element geen effect.</span>
      <br/><br/>
      <span id="s2">Dit span element staat in een div van
300 pixels breed. De tekst heeft een linker marge van 50
pixels.</span>
     </div>
  </body>
</html>
```

Deze paragraaf staat in een div van 300 pixels breed. De tekst is ingesprongen met 50 pixels.

Deze paragraaf staat in een div van 300 pixels breed. De tekst heeft een linker marge van 50 pixels.

Dit span element staat in een div van 300 pixels breed. De eigenschap text-indent heeft op dit inline element geen effect.

Dit span element staat in een div van 300 pixels breed. De tekst heeft een linker marge van 50 pixels.

We zien dat de eerste regel is ingesprongen van het blok-element p met behulp van text-indent. Hetzelfde effect krijgen we bij het inline element span met margin-left. De eigenschap text-indent heeft op inline elementen geen effect.

3.3.2 text-align

Wanneer we tekst horizontaal willen uitlijnen binnen een pagina, kunnen we gebruik maken van de eigenschap *text-align*. Deze eigenschap is alleen beschikbaar voor block elementen en moet niet verward worden met het HTML element. Dat element centreert namelijk niet alleen tekst, maar ook plaatjes.

text-align syntax

syntax:	selector { text-align : *sleutelwoord*; }
sleutelwoord:	left | center | right | justify| inherit
default:	browser afhankelijk

Dus we kunnen de tekst links, rechts, gecentreerd of uitgevuld willen weergeven.
Een toelichting op de sleutelwoorden:

text-align: left	text-align: center	text-align: right	text-align: justify
Zoals u ziet is deze tekst links uitgelijnd.	Deze tekst wordt gecentreerd weergegeven.	Deze tekst wordt aan de rechter kant uitgelijnd.	Bij justify worden regels opgevuld met extra ruimte tussen wooorden, behalve bij de laatste regel.

```
<html>
      <head>
    <style>
            div {
              width:  500px;
              text-align: right;
            }
      </style>
  </head>
  <body>
  <div>Elke week gaan wij, Hetty en Mieke, met elkaar de
strijd aan om de meeste stappen te zetten in een werkweek.
De fitbit zussenstrijd begint op maandagnacht 0:01 en
eindigt op vrijdagnacht 0:00. Omdat wij beiden erg leuk
vinden en fanatiek aan het stappen zijn, willen wij jullie
graag meenemen. Check ons <a
href="http://www.thuisbijdezussen.nl">blog</a>. :)
</body>
</html>
```

Elke week gaan wij, Hetty en Mieke, met elkaar de strijd aan om de meeste stappen te zetten in een werkweek. De fitbit zussenstrijd begint op maandagnacht 0:01 en eindigt op vrijdagnacht 0:00. Omdat wij beiden erg leuk vinden en fanatiek aan het stappen zijn, willen wij jullie graag meenemen. Check ons blog. :)

Open bovenstaand voorbeeld. We zien dat de paragraaf tekst rechts is uitgelijnd. Pas de code aan zodat de tekst in de div uitgevuld wordt weergegeven.

3.3.3 vertical-align

Voor het verticaal uitlijnen van tekst gebruiken we in CSS de *vertical-align* eigenschap. Deze eigenschap is van toepassing op inline elementen zoals <img>, , <b> en <i>. De syntax voor de vertical align eigenschap is als volgt:

vertical-align syntax

syntax:	selector { vertical-align: *sleutelwoord*; }
sleutelwoord:	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <lengte> | <percentage> | inherit
default:	baseline

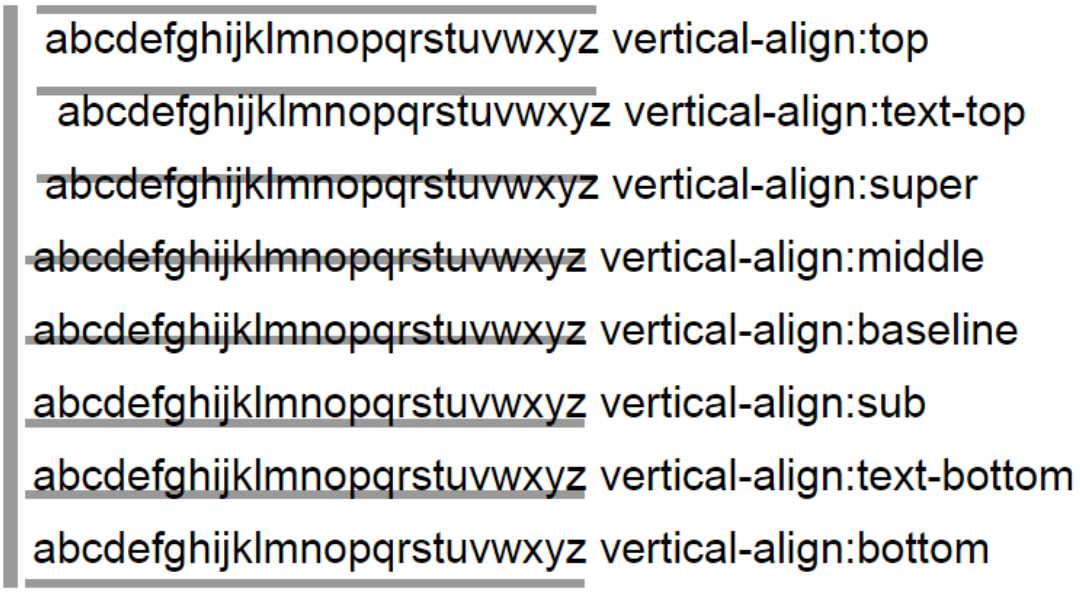
We kunnen voor het *sleutelwoord* acht waarden opgeven. Deze waarden geven aan op welke hoogte het element komt te staan, ten opzichte van de rest van de regel.

Stel dat een plaatje via css een vertical-align waarde meekrijgt. De volgende tabel geeft aan hoe het plaatje dan in de tekst wordt geplaatst.

vertical-align	plaatsing
top	De bovenkant van het plaatje wordt wordt bovenaan in de regel geplaatst. Dat kan de bovenkant van de tekst zijn, maar als er een groter plaatje in staat, dan wordt de hoogte van de regel door dit plaatje bepaald.
text-top	De bovenkant van het plaatje wordt bovenaan in de tekst geplaatst. De hoogste letter van dit lettertype bepaalt de hoogte van de tekst.
super	Superscript: de onderkant van het plaatje wordt op de hoogte van de letters zonder stok geplaatst (zoals a, e, en m)
middle	Het midden van het plaatje wordt uitgelijnd met het midden van de regel.
baseline (default)	De onderkant van het plaatje wordt uitgelijnd met de onderkant van de letters zonder stok (zoals a, e, en m).
sub	Subscript: de onderkant van het plaatje wordt geplaatst tussen de baseline en de onderkant van letters met stok naar beneden, zoals g, p en j.
text-bottom	De onderkant van het plaatje wordt geplaatst ter hoogte van de onderkant van de letters met stok naar beneden, zoals g, p en j.
bottom	De onderkant van het plaatje wordt op de onderkant van de regel geplaatst. Dat kan de onderkant van de tekst zijn, maar onder invloed van grotere inline elementen, zoals plaatjes, kan dat ook lager zijn.

```
<html>
      <head>
    <style>
              #sb{ vertical-align:sub; }
#tb{ vertical-align:text-bottom;  }
#bt{ vertical-align:bottom;  }
#bl{ vertical-align:baseline;  }
#md{ vertical-align:middle;  }
#tp{ vertical-align:top;  }
#tt{ vertical-align:text-top;  }
#sp{ vertical-align:super; }

body {
  font-family:sans-serif; font-size:30px;
}

img{
     width:385px;
     height:6px;
     margin-left:-5px;
     margin-right:-380px;
     margin-top:0;
     margin-bottom:0;
}

img.blok {height:50px;width:10px; margin:0px; margin-right:10px;
vertical-align:middle; }
      </style>
```

```
  </head>
   <body>
    <div>
      <img class="blok"
      src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">
      <img id="tp" src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">abcdefghijklmnopqrstuvwxyz
vertical-align:top<br>
      <img class="blok"
      src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">
      <img id="tt" src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">
      abcdefghijklmnopqrstuvwxyz vertical-align:text-top<br>
      <img class="blok"
      src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">
      <img id="sp" src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">abcdefghijklmnopqrstuvwxyz
vertical-align:super<br>
      <img class="blok"
      src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png"><img id="md"
src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">abcdefghijklmnopqrstuvwxyz
vertical-align:middle<br>
      <img class="blok" src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png"><img id="bl"
src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">abcdefghijklmnopqrstuvwxyz
vertical-align:baseline<br>
      <img class="blok" src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png"><img id="sb"
src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">abcdefghijklmnopqrstuvwxyz
vertical-align:sub<br>
      <img class="blok" src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png"><img id="tb"
src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">abcdefghijklmnopqrstuvwxyz
vertical-align:text-bottom<br>
      <img class="blok" src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png"><img id="bt"
src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/lijn.png">abcdefghijklmnopqrstuvwxyz
vertical-align:bottom<br>
    </div>
  </body>
</html>
```



Behalve de genoemde waarden kunnen we ook een percentage of een lengte instellen. Met een positief percentage wordt de baseline zoveel procent van de line-height (zie de volgende paragraaf) hoger geplaatst. Met een negatief percentage wordt de baseline verlaagd. Op dezelfde manier kunnen we ook lengte-eenheden gebruiken (px, em, pt, etcetera) om de baseline te verplaatsen.

Van de genoemde vertical-align waarden zijn top, middle en bottom ook op tabel cellen van toepassing. We kunnen daarmee de tekst in een cel bovenaan, in het midden of onderaan uitlijnen.

Het onderstaande voorbeeld verklaart het één en ander nader:

`vertical-align: top`	Deze tekst wordt bovenaan uitgelijnd in het blok.
`vertical-align: center`	Deze tekst staat in het midden van het blok.
`vertical-align: bottom`	Deze tekst wordt onderaan uitgelijnd in het blok.

3.3.4 Text-spacing

Onder de term text spacing vallen verschillende CSS eigenschappen waarmee we de ruimte tussen tekens, woorden en regels kunnen instellen. We kunnen onderscheid maken tussen *letter-spacing*, *word-spacin* en *line-height*. Deze eigenschappen worden onderstaand beschreven.

Letter spacing

Met de letter-spacing eigenschap kunnen we de afstand tussen individuele letters aangeven. Dit kan handig zijn wanneer we bijvoorbeeld een koptekst meer willen laten opvallen. De syntax voor letter-spacing is als volgt:

letter-spacing syntax

syntax:	selector { letter-spacing: *sleutelwoord*; }
sleutelwoord:	<lengte> | normal | inherit
default:	normal

Waarde is hier de numerieke afstand die men wilt tussen de tekens. Het sleutelwoord is hier de meeteenheid. We kunnen als waarde ook een negatief getal opgeven zodat de tekens juist dichter bij elkaar komen te staan. Normal is de default waarde. Dat staat bij letter-spacing voor de waarde 0.

Word spacing

De eigenschap word-spacing doet in principe hetzelfde als letter-spacing, alleen dan toegepast op woorden. We kunnen ook hier een negatieve waarde opgeven om ervoor te zorgen dat de woorden juist dichter bij elkaar komen te staan. Dit kan simpelweg door een minteken voor de waarde te zetten. Beide waarden voor de word-spacing worden in het volgende voorbeeld getoond:

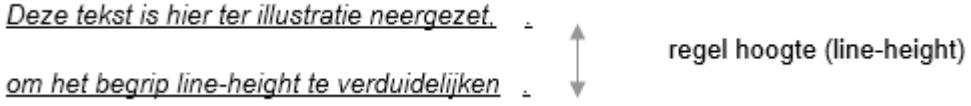
```
<html>
      <head>
    <style>
      #normaal { word-spacing: 10px; }
        #negatief { word-spacing: -3px; }
      </style>
  </head>
  <body>
    <p id="normaal">Word spacing, ofwel de ruimte tussen
woorden is nu ruimer ingesteld.</p>
    <p id="negatief">En dit is word-spacing met een
negatieve waarde ingesteld.</p>
  </body>
</html>
```

Word spacing, ofwel de ruimte tussen woorden is nu ruimer ingesteld.

En dit is word-spacing met een negatieve waarde ingesteld.

3.3.5 line-height

Met line-height geven we aan hoeveel ruimte we tussen regels willen zien. Hoe de regelhoogte precies wordt gemeten kunt u zien in het volgende plaatje:



We kunnen dit aangeven met de volgende syntax:

line-height syntax

syntax:	selector { line-height: *sleutelwoord;*}
sleutelwoord:	<lengte> | <percentage> | <multiplier> | normal | inherit
default:	normal

Naast deze notatievorm hebben we voor het instellen van *line-height* ook de mogelijkheid de meeteenheid (van de lengte of het percentage) weg te laten. Het getal wat dan nog resteert is een multiplier. Dit houdt in dat de regelhoogte twee maal de normale hoogte wordt als we het volgende opgeven:

selector { line-height: *2;*}

De normale hoogte is de hoogte die is ingesteld in het gebruikte lettertype.

3.3.6 text-decoration

Met de text-decoration eigenschappen kunnen we een lijn trekken over, onder of door een tekst. Dit kan gedaan worden middels de volgende syntax:

text-decoration syntax

syntax: selector { text-decoration : *sleutelwoord:*}
sleutelwoord: none | underline | overline | line-through | blink | inherit
default: none

Met het *blink* sleutelwoord kunnen we een tekst laten knipperen op een interval dat vast is ingesteld door de browser. Dit wordt zelden meer gebruikt. Een toepassing die nog wel veel wordt gebruikt gaan we in onderstaand voorbeeld toepassen.

```
<html>
      <head>
    <style>
            a { text-decoration:  none; }
          a:hover { text-decoration:  underline; }
      </style>
  </head>
  <body>
    <a href="#">Beweeg met uw muis over deze link</a>
  </body>
</html>
```

Beweeg met uw muis over deze link

We zien hier in de code dat we voor de standaardweergave van een link, de text-decoration eigenschap op none hebben gezet. De tekst blijft dus gewoon, zonder decoratie. Daarnaast hebben we middels de :hover pseudo klasse, ingesteld dat de tekst onderstreept wordt indien men er met de muis overheen beweegt.

3.4 Samenvatting

Met verschillende font eigenschappen kunnen we teksten een stijl geven, zodat ze goed passen binnen de gewenste webpagina's. Binnen lettertypen kunnen we een onderscheid maken tussen: serif, sans-serif, monospace, cursive en fantasy fonts. Met behulp van @font-face kunnen modernere browsers ook andere lettertypen gebruiken.. Om waarden aan te geven voor grootte kunnen we gebruik maken van beschrijvende en niet beschrijvende meetvormen. Daarbinnen kunnen we deze meetvormen onderscheiden in absolute en relatieve meetvormen.

4 Kleurgebruik, afbeeldingen en animaties

4.1 Inleiding

In dit hoofdstuk gaan wij afbeeldingen toevoegen aan onze content en wij gaan leren hoe wij deze kunnen positioneren. Wij gaan kijken hoe kleuren in CSS worden toegepast op teksten en elementen. Ook gaan wij schaduw-, transparantie- en gradienteffecten toepassen en leren wij zowel 2D als 3D transformaties te maken. Daarnaast komt het opmaken van HTML list items met CSS aan bod. Tot slot kijken wij naar het nut van het CSS content attribuut.

- Kunnen toevoegen en aanpassen van kleuren en afbeeldingen binnen een HTML document
- Achtergrondplaatjes kunnen positioneren
- Kunnen maken van schaduw-, transparantie-, en gradienteffecten
- List-items kunnen vormgeven als menu's
- 2D en 3D transformaties kunnen maken
- Keyframes kunnen toepassen
- Het CSS content attribuut kunnen verklaren

4.2 display

Hoe HTML elementen binnen een tekst worden weergegeven hangt af van de *display* eigenschap. We hebben het al gehad over inline en block elemenenten. List items hebben ook een eigen herkenbare weergave (opsomming onder elkaar), evenals cellen van tabellen bijvoorbeeld.

Met de display eigenschap kunnen we deze weergave aanpassen, of zelfs helemaal onzichtbaar maken. Een veel gebruikte toepassing is het opbouwen van een horizontaal menu. Zo'n menu wordt meestal gemaakt van list items, waarbij elk item een link bevat naar een andere pagina. Standaard worden de links dan onder elkaar geplaatst. Met behulp van display:inline kunnen we ze naast elkaar weergeven.
Ook kunnen we block elementen inline weergeven of andersom.

De syntax voor deze eigenschap, met de meest voorkomende waarden is:

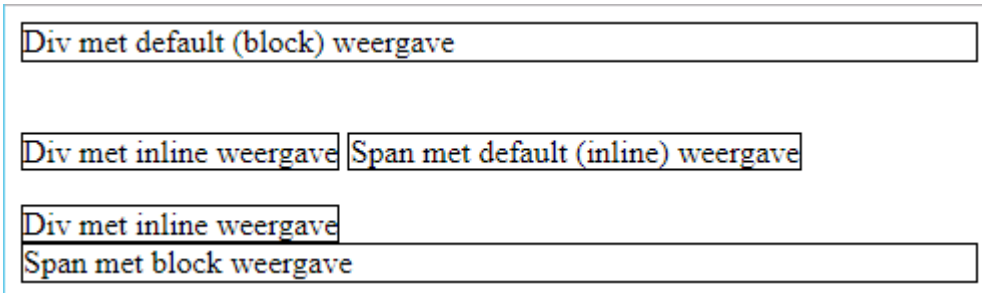
display syntax

syntax:	selector { display: *sleutelwoord;*}
sleutelwoord:	block | inline | inline-block | list-item | flex | flow | grid | none
default:	inline

Er zijn met de komst van CSS 3 een groot aantal andere mogelijke waarden bij gekomen. Omdat dit waarden zijn die vooral op tabellen focussen en in de praktijk niet veel ingezet worden, zullen wij hier niet verder op in gaan.

Een toelichting op de flex, flow en grid sleutelwoorden komt in het hoofdstuk 'positionering' uitgebreid aan bod. De overige mogelijkheden kunt u over lezen op de MDN website https://developer.mozilla.org/en-US/docs/Web/CSS/display.

```
<html>
      <head>
    <style>
            div, span {
              border: 1px solid black;
            }

            .inline {
              display: inline;
            }

            .block {
              display: block;
            }

            .table-cell {
              display: table-cell;
            }
      </style>
  </head>
  <body>
    <div>Div met default (block) weergave</div>
    <br/><br/>
    <div class="inline">Div met inline weergave</div>
    <span>Span met default (inline) weergave</span>
    <br/><br/>
    <div class="inline">Div met inline weergave</div>
    <span class="block">Span met block weergave</span>
  </body>
</html>
```



We gaan nu vervolgens samen een horizontaal menu aanmaken.

```
<html>
      <head>
    <style>

      </style>
  </head>
  <body>
    <ul id="navigatie">
    <li><a href="test.html">Home</a></li>
    <li><a href="producten.html">Producten</a></li>
    <li><a href="referenties.html">Referenties</a></li>
    <li><a href="forum.html">Forum</a></li>
    <li><a href="contact.html">Contact</a></li>
    </ul>
  </body>
</html>
```

- Home
- Producten
- Referenties
- Forum
- Contact

In bovenstaande file wordt een aantal links onder elkaar als een lijst weergegeven. Voeg tussen <STYLE></STYLE de volgende CSS code toe, en bekijk na elke stap het resultaat:

```
#navigatie { padding:0; }                    /* niet
inspringen */
#navigatie li {display:inline; padding:2px; } /* menu items
naast elkaar zetten, met enige afstand */
#navigatie li a {text-decoration:none; }      /* links niet
onderstrepen */
#navigatie li a:hover {color:orange}          /* actieve
link andere kleur geven */
```

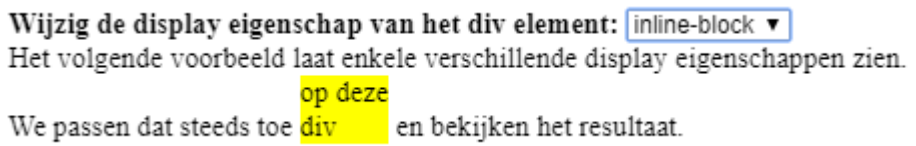
Eindresultaat:

Home Producten Referenties Forum Contact

In deze opdracht is #navigatie het ID van een unordered list. We maken we gebruik van padding, waarmee we aangeven hoeveel ruimte er moet komen tussen het element en de rand van de omliggend box. Standaard geldt voor een lijst dat deze ingesprongen wordt weergegeven: die inspringing heffen we op door de padding op 0 te zetten. We komen hier volgend hoofdstuk uitgebreider op terug.

```
<html>
      <head>
    <style>
      label { font-weight:bold; }
      #voorbeeld { background-color:yellow; }
      </style>
      <script>
      function display(){
  document.getElementById('voorbeeld').style.display =
               document.getElementById('keuze').value;
}
      </script>
  </head>
  <body>
    <form>
      <label for="keuze">Wijzig de display eigenschap van
het div element: </label>
        <select id="keuze" onchange="display()">
        <option value="block">block</option>
        <option value="inline">inline</option>
        <option value="inline-block">inline-block</option>
          <option value="table-cell">table-cell</option>
        <option value="none">none</option>
      </select>
    </form>

    Het volgende voorbeeld laat enkele verschillende
display eigenschappen zien.
    We passen dat steeds toe <div id="voorbeeld">op
deze<br>div</div> en bekijken het resultaat.

  </body>
</html>
```

Open het bovenstaande bestand in de browser. Hierin kunnen we met behulp van een pull down menu de display eigenschap van een div wijzigen. De beschikbare waarden zijn block, inline, inline-block en none. Probeer deze waarden uit en bekijk het resultaat.



Er zijn naast de sleutelwoorden die hier zijn genoemd nog meer display mogelijkheden. Deze mogelijkheden hebben onder meer betrekking op tabellen, maar zullen verder niet worden besproken in deze training.

4 Kleurgebruik, afbeeldingen en animaties

4.3 Kleurgebruik

We weten nu al het één en ander over hoe een CSS document is opgebouwd, welke HTML elementen er zijn en wat we met teksten kunnen doen in CSS. In deze paragraaf zullen we toelichten waar en hoe we kleuren kunnen toepassen. De CSS eigenschappen die ons in staat stellen iets met kleur te doen zullen we hieronder bespreken.

4.3.1 Background-color

Een veel gebruikte CSS eigenschap is background-color. Dit stelt, zoals het woord al zegt, een achtergrond kleur in voor HTML elementen.

De syntax voor deze eigenschap is:

background-color syntax

syntax:	selector { background-color: *sleutelwoord*;}
sleutelwoord:	<kleur> | transparent | inherit
default:	transparent

4.3.2 Border-color

Met de border-color eigenschap kunnen we de randen een kleur geven van een HTML element. Denk hierbij bijvoorbeeld aan de randen van een tabel, een div of een invoerveld.

De syntax voor de border-color eigenschap is als volgt:

border-color syntax

syntax:	selector { border-color: *sleutelwoord;*}
sleutelwoord:	<kleur> | transparent | inherit transparent | inherit
default:	geen

4.3.3 Color

Tot slot kunnen we met de eigenschap *color*, tekst een kleur geven.
De syntax hiervoor is:

color syntax

syntax:	selector { color: *sleutelwoord;*}
sleutelwoord:	<kleur> | inherit
default:	browser afhankelijk

De sleutelwoorden (de kleur) voor de genoemde eigenschappen kunnen we invullen met rgb waarden (numeriek of procentueel), met hexadecimale waarden of met kleurnamen. De verschillende beschikbare kleurnamen staan in de onderstaande tabel weergegeven. Daarbij staan de kleurcodes die voor die kleuren beschikbaar zijn. Zoals u kunt raden, zijn er veel meer mogelijke kleuren te maken, we gaan in de onderstaande tabel echter uit van de beschikbare kleurnamen.

We zien hier alle kleurnamen, de bijbehorende RGB waarden en hexadecimale waarden:

Kleur	Percentage	Numeriek	Hexadecimaal	Kort hex
Red	rgb(100%,0%,0%)	rgb(255,0,0)	#FF0000	#F00
Orange	rgb(100%,40%,0%)	rgb(255,102,0)	#FF6600	#F60
Yellow	rgb(100%,100%,0%)	rgb(255,255,0)	#FFFF00	#FF0
Green	rgb(0%,50%,0%)	rgb(0,128,0)	#008000	

4 Kleurgebruik, afbeeldingen en animaties

Blue	rgb(0%,0%,100%)	rgb(0,0,255)	#0000FF	#00F
Aqua	rgb(0%,100%,100%)	rgb(0,255,255)	#00FFFF	#0FF
Black	rgb(0%,0%,0%)	rgb(0,0,0)	#000000	F000
Fuchsia	rgb(100%,0%,100%)	rgb(255,0,255)	#FF00FF	#F0F
Gray	rgb(50%,50%,50%)	rgb(128,128,128)	#808080	
Lime	rgb(0%,100%,0%)	rgb(0,255,0)	#00FF00	#0F0
Maroon	rgb(50%,0%,0%)	rgb(128,0,0)	#800000	
Navy	rgb(0%,0%,50%)	rgb(0,0,128)	#000080	
Olive	rgb(50%,50%,0%)	rgb(128,128,0)	#808000	
Purple	rgb(50%,0%,50%)	rgb(128,0,128)	#800080	
Silver	rgb(75%,75%,75%)	rgb(192,192,192)	#C0C0C0	
Teal	rgb(0%,50%,50%)	rgb(0,128,128)	#008080	
White	rgb(100%,100%,100%)	rgb(255,255,255)	#FFFFFF	#FFF

Visueel in een hexagon weergegeven:

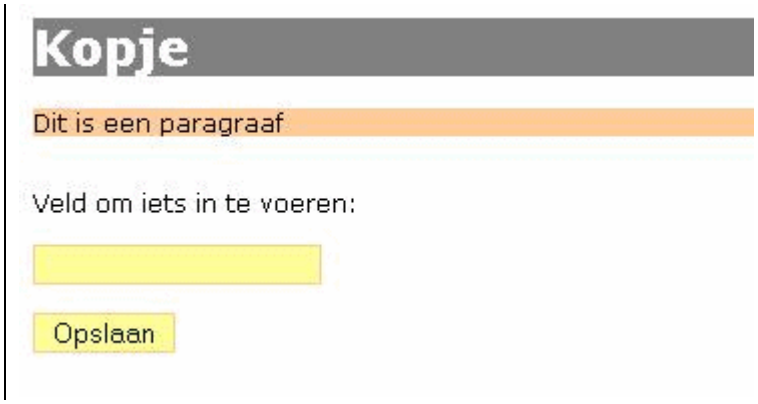
Bovenstaande kleuren met bijbehorende waarden zegt u wellicht weinig. Bekijk daarom ook de volgende webpagina van W3schools met een kleuren selectie tool: https://www.w3schools.com/colors/colors_picker.asp.

Binnen een kleurenpalet kunnen we nog de web-safe colors onderscheiden. Deze kleuren veranderen niet van kleur wanneer ze worden weergegeven op een computer die 256 kleuren ondersteunt. Wanneer we web-safe kleuren willen maken met RGB dan kunnen we de percentages noteren die deelbaar zijn door 20, of de numerieke notaties die deelbaar zijn door 51. Als we hexadecimale web-safe kleuren willen gebruiken, mogen we alleen paren maken van de tekens 0, 3, 6, 9, C en F. Dus bijvoorbeeld: 0033CC.

Zorg ervoor dat gebruikte kleuren passen bij de kleuren van plaatjes op de pagina. Op internet zijn generatoren te vinden waarmee kleurenschema's kunnen worden ontworpen.

```
<html>
      <head>
    <style>
      body {
        font-family:        verdana;
            font-size:        9pt;
            color:                    white;
        background-color: black;
      }

      input {
        border-color:   black;
        border-style: solid;
            border-width: 1px;
      }
      </style>
  </head>
    <body>
    <h1>Kopje</h1>
    <p>Dit is een paragraaf</p>
    <br>
    Veld om iets in te voeren:
    <br><br>
    <input type="text" name="txtInvoer">
    <br><br>
    <input type="button" name="btnOpslaan" value="Opslaan">
  </body>
</html>
```

Bekijk de bovenstaande code en zorg dat het resultaat er (ongeveer) als volgt uit komt te zien:



We zien dat de achtergrond, de randen en de lettertype kleuren van de div, het invoerveld en de pagina zijn aangepast. Zorg ervoor dat het bestand er hetzelfde uit komt te zien. De gebruikte RGB kleuren staan hier vermeld: rgb(128,128,128), rgb(255,204,153) en rgb(255,255,153).

4.4 Achtergrondplaatjes

In deze paragraaf bespreken we hoe achtergrondplaatjes verwerkt kunnen worden in webpagina's door gebruik te maken van de CSS eigenschap background-image. Behalve als achtergrond, kan een background-image ook op de voorgrond dienst doen, of als afbeelding in een kader van een element. Dit laatste gaan wij zien in het volgende hoofdstuk.
We kunnen een achtergrondplaatje instellen op ieder willekeurig block en inline element. Wat we ook vaak zien is dat er een achtergrondplaatje wordt ingesteld op de *body* tag.

background-image syntax

syntax:	selector { background-image: *sleutelwoord*; }
sleutelwoord:	url() | none | inherit
default	none

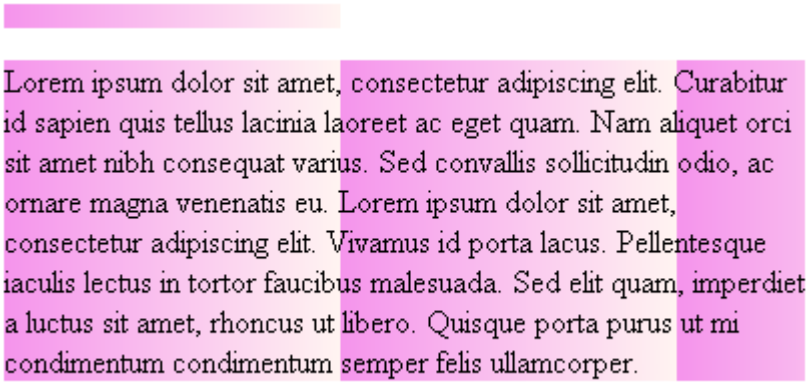
Bovenstaand zien we de syntax voor het instellen van een achtergrondplaatje. Deze eigenschap wordt niet automatisch overgeërfd. Dit zou er namelijk voor zorgen dat wanneer we een achtergrondfoto instellen op de body, we deze foto vervolgens als achtergrond in ieder element nogmaals zouden zien.

Om een plaatje toe te kennen als background-image gebruiken we dezelfde syntax als bij list-style-image: url(<bestandsnaam>).
Bijvoorbeeld:

```
div#plaatje { background-image: url(img/plaatje.jpg);}
```

```
<html>
      <head>
    <style>
            p {
        width:400px;
        /* background-image: */
      }
      </style>
  </head>
  <body>
      <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/fade.jpg"/>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Curabitur id sapien quis tellus lacinia laoreet ac
eget quam. Nam aliquet orci sit amet nibh consequat varius.
Sed convallis sollicitudin odio, ac ornare magna venenatis
eu. Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Vivamus id porta lacus. Pellentesque iaculis lectus
in tortor faucibus malesuada. Sed elit quam, imperdiet a
luctus sit amet, rhoncus ut libero. Quisque porta purus ut
mi condimentum condimentum semper felis ullamcorper. </p>
  </body>
</html>
```

De bovenstaande pagina bevat een plaatje en een tekst. Pas de code nu zo aan dat de paragraaf als achtergrondplaatje met als pad de url die ook als 'src' binnen het image element gebruikt is.
We zien dan het volgende resultaat:



De default waarde is repeat, dus zowel in horizontale als in verticale richting.

Pas de code in bovenstaande voorbeeld aan zodat het achtergrondplaatje alleen in verticale richting wordt herhaald en bekijk het resultaat

We zien nu dat alleen de linker kant wordt opgevuld met het plaatje. De rest van de paragraaf behoudt een witte achtergrond. We kunnen er een consistenter beeld van maken door de paragraaf een achtergrondkleur te geven, die overeenkomt met de kleur aan de rechterkant van het plaatje.

Geef het de paragraaf behalve het achtergrondplaatje ook een achtergrondkleur, met de hexadecimale waarde #FFF4F1.
Haal het losse plaatje bovenin de pagina weg en bekijk het resultaat.

4.4.1 background-attachment

We kunnen met deze eigenschap ervoor kiezen of we willen dat een achtergrondplaatje mee naar beneden gaat als we op een pagina naar beneden scrollen

background-attachment syntax

syntax:	selector { background-attachment: *sleutelwoord*; }
sleutelwoord:	scroll | fixed | inherit
default:	scroll

Met de code background-attachment: fixed; kunnen we ervoor zorgen dat de achtergrond zijn positie ten opzichte van het venster behoudt. Met het *scroll* sleutelwoord bereiken we het tegenover gestelde. Wanneer we naar beneden zouden scrollen, scrollt de image uit het beeld.

4.4.2 background-position

Wanneer we een achtergrondplaatje willen instellen, willen we natuurlijk ook kunnen bepalen waar hij komt te staan op het scherm. Hiervoor gebruiken we de *background-position* eigenschap

background-position syntax

syntax:	selector { background-position: *sleutelwoord*; }
sleutelwoord:	<percentage> | <lengte waarde> | left | center | right] | <percentage> | <lengte waarde> | top | center | bottom] | inherit
default:	0% 0%

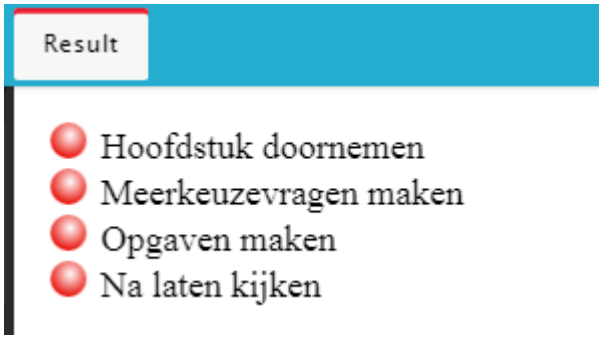
We zien in het bovenstaande schema dat twee waarden ingevuld moeten worden. Namelijk een waarde voor de horizontale positie en voor de verticale positie. We zien dat we zowel percentages als lengte waarden (bijvoorbeeld *px* of *cm*) en beschrijvende posities op kunnen geven als sleutelwoorden.

Wanneer we de positie met één percentage aangeven, dan geldt dat percentage alleen voor de horizontale positionering. De verticale positionering wordt automatisch op 50% gesteld.

Het gebruik van percentages werkt als volgt. Stel bijvoorbeeld dat we alleen de horizontale positie willen bepalen van een plaatje. Bij een positie van 0 % staat de linker kant van het plaatje tegen de linker kant van de box. Bij 50% staat het midden van het plaatje op het midden van de box. Bij 100% staat de rechterkant van het plaatje tegen de rechterkant van de box. Met andere woorden: de gebruikte grens van het plaatje verschuift mee met het gebruikte percentage.

We hebben het eerder gehad over de list-style-image. Daarmee kunnen we zelf een plaatje toekennen als markering van een list item. De verticale positie daarvan kunnen we echter niet bepalen.

```
<html>
      <head>
    <style>
      body {
        font-size: 20px;
      }

      ul {
        list-style-image: url("https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bullet.png");
      }
      </style>
  </head>
  <body>
   <ul>
     <li>Hoofdstuk doornemen</li>
     <li>Meerkeuzevragen maken</li>
     <li>Opgaven maken</li>
     <li>Na laten kijken</li>
   </ul>
  </body>
</html>
```



We gaan er nu voor zorgen dat de plaatjes verticaal gecentreerd worden weergegeven.

We geven eerst aan dat list-items geen markering meer krijgen. Daarnaast zorgen we ervoor dat list items niet automatisch worden ingesprongen. Haal de bestaande CSS code voor een list weg en voeg het volgende toe.

```
ul {  list-style-type:none;
      margin:0;
      padding:0;
}
```

Op het gebruik van margin en padding komen we in het volgende hoofdstuk terug.

Nu geven we list items een background-image.

```
      li {
           background-image:url("https://s3-us-west-
      2.amazonaws.com/s.cdpn.io/1201815/bullet.png");
           background-repeat:no-repeat; // niet herhalen
           background-position:0 50%;   // verticaal
      gecentreerd
      }
```

Bekijk het resultaat. We zien nu het plaatje wel verschijnen, maar door de tekst heen. Om dit op te lossen geven we aan dat de tekst enige afstand van de linker kant moet beginnen. Voeg daarom de volgende regel toe aan de code voor list items:

```
      padding-left: 35px;
```

Bekijk nu nogmaals het resultaat.
De bullets staan nu verticaal gecentreerd weergegeven.

4.5 List items en CSS

List items worden gebruikt om opsommingen weer te geven. We kennen een verschil tussen gesorteerde en ongesorteerde lijsten. Gesorteerde lijsten (ordered lists, met als tag: <ol>) zijn lijsten met een nummering als aanduiding. Deze nummering kan zowel numeriek als alfanumeriek zijn. Ongesorteerde lijsten (unordered lists, met als tag: <ul>), zijn lijsten zonder nummering. Deze kunnen per item zijn aangeduid met bijvoorbeeld ‘bullets’.

We kunnen een lijst niet in een paragraaf plaatsen, wel in een div.

In HTML zijn er weinig mogelijkheden om de weergave van lijsten aan te passen. De behandelde CSS tekst-stijlen zijn echter ook geschikt voor lijsten. Daarnaast kent CSS eigenschappen die specifiek op list items van toepassing zijn. We zullen de belangrijkste list eigenschappen bespreken en vervolgens gaan bekijken in een voorbeeld.

```
<html>
      <head>
    <style>
            #lijst1 {
              list-style-type:      upper-roman;
              list-style-position:  inside;
            }

            #lijst2 {
              list-style-type:      circle;
              list-style-position:  outside;
            }

            #lijst3 {
              list-style-image:     url("https://s3-us-
west-2.amazonaws.com/s.cdpn.io/1201815/5hart_icoon.jpg");
            }

            #lijst4 {
```

```
            list-style-type:      none;
          }
          #lijst4 li {
            display:           inline;
          }
      </style>
  </head>
  <body>
    <ol id="lijst1">
      <li>Nintendo</li>
      <li>Sony</li>
      <li>Microsoft</li>
    </ol>
    <ul id="lijst2">
      <li>Wii</li>
      <li>Playstation</li>
      <li>Xbox</li>
    </ul>
    <ul id="lijst3">
      <li>De</li>
      <li>drie</li>
      <li>spelcomputers</li>
    </ul>
    <br><br>
    <ul id="lijst4">
      <li>De</li>
      <li>horizontale</li>
      <li>display</li>
    </ul>
  </body>
</html>
```

I. Nintendo
II. Sony
III. Microsoft

- Wii
- Playstation
- Xbox

De
drie
spelcomputers

De horizontale display

We gaan nu per list eigenschap ook naar bovenstaand voorbeeld kijken, zodat we kunnen zien wat het effect is van verschillende instellingen.

4.5.1 List style type

Met deze eigenschap kunnen we de aanduiding van een lijst wijzigen. We kunnen bijvoorbeeld zeggen dat we Romeinse cijfers of decimalen gebruiken voor de nummering.

list-style-type syntax

syntax	selector { list-style-type: *sleutelwoord;*}
sleutelwoord:	circle | disc | square | decimal | decimal-leading-zero | armenian | georgian | lower-alpha| upper-alpha | lower-greek | lower-roman | upper-roman | lower-latin | upper-latin | none | inherit
default:	none

De mogelijke sleutelwoorden staan in de tabel hieronder weergegeven.

Sleutelwoord	**Effect**
circle	Gebruikt een open cirkel als aanduiding
disc	Gebruikt een dichte cirkel als aanduiding
square	Gebruikt een (dicht of open) vierkant als aanduiding
decimal	1,2,3,4,5,….
decimal-leading-zero	01,02,03,04,05,….
armenian	Traditionele Armenische nummering
georgian	Traditionele Georgische nummering
lower-alpha	a,b,c,d,e,….
upper-alpha	A,B,C,D,E,….
lower-greek	Traditionele griekse symbolen in kleine tekens
lower-roman	i,ii,iii,iv,v,….
upper-roman	I,II,III,IV,V,….
lower-latin	a,b,c,d,e,….
upper-latin	A,B,C,D,E,….
None	*Geen* aanduiding

List style image

Met deze eigenschap kunnen we in plaats van de standaard aanduiding, een plaatje plaatsen voor ieder list-item. Het werkt als volgt:

list-style-image syntax

syntax:	selector { list-style-image: url(*sleutelwoord*); }
sleutelwoord:	bestandsnaam | none | inherit
default:	geen

We hoeven dus alleen een bestandsnaam op te geven tussen de haakjes van het *url* sleutelwoord. In ons voorbeeld was dit dus:*5hart_icoon.jpg*. Er wordt dan voor ieder item in de list dit plaatje gebruikt. Een nadeel van het gebruik van list-style-image is dat we geen invloed kunnen uitoefenen op de verticale positie van het plaatje, en dat die positie per browser verschillend kan zijn. We hebben bij de behandeling van achtergrondplaatjes al gezien hoe we dit wel kunnen aanpassen.

4.5.2 List style position

We geven met de *list-style-position* eigenschap aan hoe we de markering van een list item positioneren ten opzichte van de inhoud van het item.

De syntax hiervoor is:

list-style-position syntax

syntax: selector { list-style-position: *sleutelwoord* }
sleutelwoord: inside | outside | inherit
default: outside

Hier kunnen we het woord inside of outside opgeven als sleutelwoord. Let op de horizontale positie van de tekst in het voorbeeld. We zien dat bij de eerste opsomming de teksten verder inspringen. Dit komt omdat we bij die list de list-item-position op inside hebben ingesteld. Dit betekent dat de markering deel uitmaakt van de inhoud van het item, waardoor de tekst verder verspringt. Bij outside wordt de markering links van de inhoud geplaatst, en verspringt de tekst niet.

4.6 Shaduw-, transparantie- en gradienteffecten

4.6.1 Box-shadow en text-shadow

Ook met CSS kunnen wij schaduw effecten aan onze elementen en aan onze teksten toevoegen. Hiervoor gebruiken wij respectievelijk de box-shadow en text-shadow eigenschap. Voor beiden geldt dat ze de volgende opbouw van de syntax gebruiken:

```
<hozitonale postitie> <vericale positie> <intensiteit / verspreiding> <kleur>
```

Bij de positie waarden (horizontaal of verticaal) mag ook een negatief getal komen te staan. Dan zal de schaduw respectievelijk links of boven het element uitkomen in plaats van normaliter rechts en onder het element.

Wij kunnen ook meerdere schaduweffecten toepassen in één keer, door achter de bovengenoemde syntax een komma (,) te plaatsen en dan wederom de bovenstaande syntax te gebruiken. Deze mogelijkheid wordt vrijwel nooit benut of toegepast.

Onderstaand ziet u een voorbeeld van toepassing van beide effecten. Pas vooral de eerste drie waarden van één van de twee eigenschappen aan om te kijken wat er gebeurt.

```
<html>
      <head>
    <style>
            input {
              /* <hozitonale postitie> <vericale positie>
<intensiteit / verspreiding> <kleur> */
              box-shadow: 6px 6px 4px gray;
            }

            button {
              background-color: green;
              color: white;
              box-shadow: 6px 6px 4px gray;
```

```
            }
      </style>
  </head>
  <body>
            <label for="mobiel_nr">Mobiele nummer:</label>
            <input type="text" id="mobiel_nr" />
            <button>Ok</button>
      </body>
</html>
```

4.6.2 Transparantie

Op alle elementen in CSS kunnen wij transparantie toepassen. Dat is tegenwoordig erg in trek bij bijvoorbeeld invoervelden en afbeeldingen. De hiervoor gebruikte term en eigenschap is opacity. Een opacity van waarde 1 betekent dat het element geheel niet transparant is. Een waarde van 0 betekent dat een element volledig transparant is. Het is dus vaak zoeken naar het mooiste of beste effect met getallen tussen 0.0 en 1.0.

Een voorbeeld met een bijna geheel transparante afbeelding:

```
<html>
      <head>
    <style>
            img {
              opacity:  0.25;
            }
            span {
              position: absolute;
              left: 2%;
              top: 10%;
            }
      </style>
  </head>
 <body>
    <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bitcoin.jpg"/>

    <span>Deze absoluut gepositioneerde tekst kan nu prima
over deze afbeelding heen.</span>
  </body>
</html>
```

Ook een voorbeeld met een semi-transparant invoerveld. Dit zorgt voor een mooi (en veel toegepast) effect:

```
<html>
      <head>
    <style>
            input {
              opacity:  0.7;
              color: black;

              /* mooiere opmaak: */
              border-top-left-radius: 10px;
              border-bottom-left-radius: 10px;
              padding-left: 12px;
            }

            span {
              position: absolute;

              left: 2%;
              top: 16%;

              color: #FFF;
              font-weight: bold;
            }

      </style>
  </head>
```

```
 <body>
    <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bitcoin.jpg"/>

    <span>Bedrag: <input type="number" placeholder="Vul
hier een bedrag in..."/></span>
  </body>
</html>
```

Merk op dat als je in het bovenstaande voorbeeld de opacity te hoog zet (te dicht bij de waarde '1'), het effect weinig zin meer heeft door de donkere achtergrond.

4.6.3 Gradient effecten

Een veel gebruikt effect in foto bewerkingsprogramma's is kleurverloop. Dit wordt ook wel een gradient genoemd.
Met een gelijksoortige CSS eigenschap kunnen wij een element een kleurverloop geven.

Bij een kleurverloop hebben wij altijd een begin-, eind- en eventueel tussenkleur. Wij geven op welk type gradient wij willen gebruiken (linear of radial) en kiezen een bijbehorend subtype om aan te geven welke kan wij de gradient op willen laten verlopen (v.l.n.r., diagonaal, 78graden, etcetera).

Er zijn vele mogelijkheden voor het instellen van deze kleurverlopen. Bekijk op de W3C website hier de mogelijkheden.

Een beknopt voorbeeld van in dit geval de linear gradient:

```
<html>
      <head>
    <style>
            div {
              height: 100px;
              width:  100px;
            }

            #div1 {
              background: linear-gradient( 45deg, blue, red
);
            }

            #div2 {
              background: linear-gradient( 0deg, blue,
               green 40%, red);
            }
      </style>
  </head>
  <body>
    <div id="div1">Inhoud</div>
    <br/>
    <div id="div2">Inhoud</div>
  </body>
</html>
```

4.7 Transformaties en animaties in 2D en 3D

4.7.1 Elementen in 2d transformeren

Met CSS kunnen wij elementen in twee dimensies transformeren. Dat houdt in dat wij een element kunnen:

- wijzigen qua breedte en lengte
- verplaatsen over de x- en y-as
- verdraaien
- draaien

Hiervoor gebruiken wij de eigenschap 'transform' met één van de volgende waarden:

translate(): verplaatsen van een element. Hierbij geven wij aan hoe ver het element verplaatst wordt op de x- en y-as ten op zichte van de huidige positie.

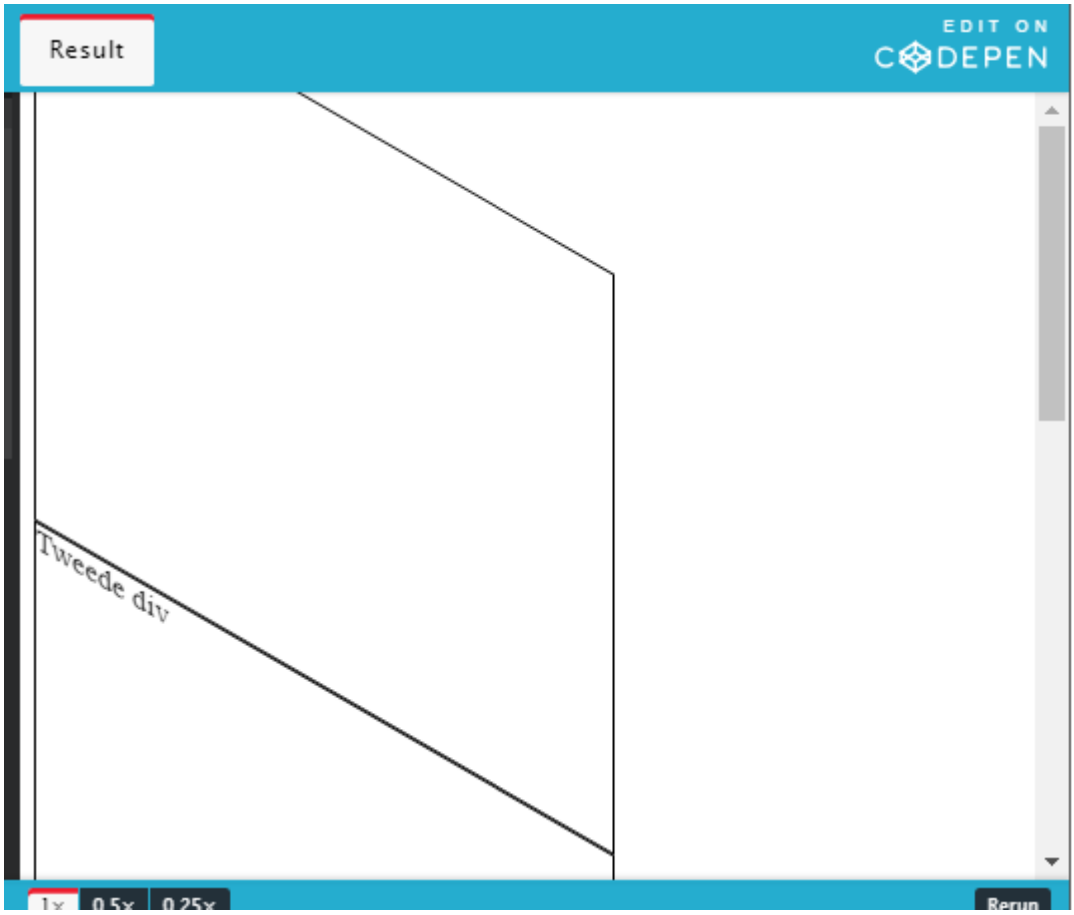
scale(): hiermee wijzigen wij de grootte van een element. De nieuwe breedte en lengte afmetingen geven wij op.

rotate(): het draaien van een element (ofwel roteren). Door ons opgegeven in aantal graden.

skew(): verdraaien van een element doen wij met 'skew'. Denk aan het veranderen van een rechthoek naar een wiebertje. Dat is het effect van verdraaien.

matrix(): dit is een numerieke notatie representatie van meerdere transform eigenschappen. zeer complex door mensen in te vullen. Dus het wordt aangeraden deze notatie eventueel te laten genereren.

```
<html>
      <head>
    <style>
            div {
            width:            20em;
            height:           20em;
            border:           solid 1px black;

            /* Hij zal altijd de laatste 'transform' optie
gebruiken */
            transform:  translate(2em, 3em);
            transform:  scale(0.5,4);
            transform:  rotate(-45deg);
            transform:  skew(10deg, 30deg);
            transform:  skewx(10deg);
            transform:  skewy(30deg);
      }
      </style>
  </head>
  <body>
            <main>
                  <div>Eerste div</div>
                  <div>Tweede div</div>
                  <div>Derde div</div>
            </main>
      </body>
</html>
```



4.7.2 Elementen in 3d transformeren

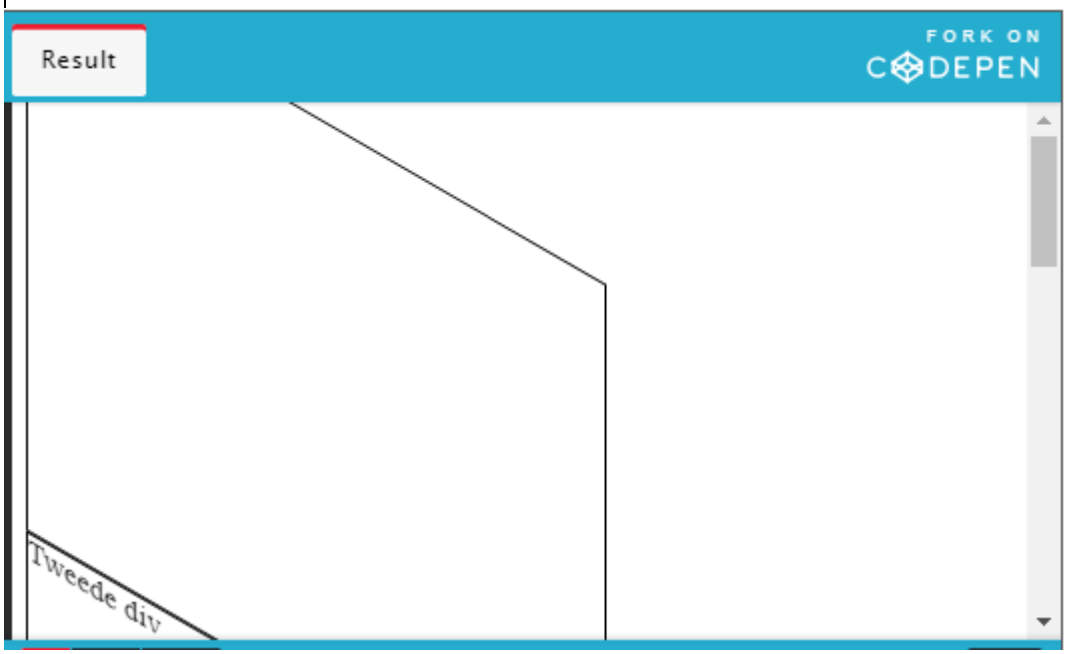
Ook om een element in drie dimensies te transformeren (dus elementen die ook een diepte of een zogeheten z-as hebben), gebruiken wij de 'transform' eigenschap. Ook de mogelijke waarden voor deze eigenschap komen grotendeels overeen. Bij veel waarden is er alleen een 'z-as' invulmogelijkheid bijgekomen.

De lijst met mogelijke 3d transformatie eigenschappen is nu:

- translate3D(x, y, z)
- scale3D (x, y, z)
- skew (x, y, z)
- skewx (x)
- skewy (y)
- skewz (z)

Bekijk voor een uitwerking van deze mogelijkheden het volgende voorbeeld:

```
<html>
      <head>
    <style>
            div {
            width:            20em;
            height:           20em;
            border:           solid 1px black;

            /* Hij zal altijd de -laatste- transform optie
gebruiken */
            transform:  translate3D(2em, 3em, 4em);
            transform:  scale3d(0.5,4,7);

            /* rotate3d = x,y,z,angle */
            transform:  rotate3d(-10deg, 10deg, 10deg, 5);
            transform:  skew(10deg, 30deg, 20deg);
            transform:  skewx(10deg);
            transform:  skewy(30deg);
            transform:  skewz(20deg);
      }
      </style>
  </head>
  <body>
            <main>
                  <div>Eerste div</div>
                  <div>Tweede div</div>
                  <div>Derde div</div>
            </main>
      </body>
</html>
```



Het effect van 3d transformaties is soms slecht zichtbaar op 2d elementen (zoals het vlak in het vorige voorbeeld). De volgende website toont op duidelijke wijze het effect: https://desandro.github.io/3dtransforms/examples/transforms-01-functions.html.

4.7.3 Elementen animeren

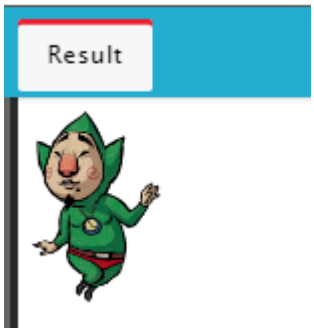
Bij het animeren met behulp van keyframes maken wij wederom gebruik van een apestaart (@) notatie. Dit betekent net als bij het gebruik van @font-face, dat we naar dit deel van de code kunnen verwijzen vanuit een ander stuk CSS code.

Het idee bij een animatie is dus om bij de animation eigenschap te verwijzen naar de bijbehorende keyframe(s). Binnen een @keyframes blok kunnen wij meerdere keyframes (sleutelpunten dus eigenlijk) declareren. Ieder keyframe is een zelf bedacht moment binnen de animatie.

In het volgende voorbeeld laten we een personage 'bewegen' van klein naar groot met behulp van twee keyframes.

Dit voorbeeld maakt gebruik van de verkorte 'animation' notatie. Het voorbeeld wat daarop volgt heeft iedere 'animation' eigenschap uitgewerkt.

```
<html>
      <head>
    <style>
            @keyframes lopen {
            0%  {
                  top:  0em;
                  width:      2em;
                  height:     3em;
            }
            100% {
                  top:  25em;
                  left: 25em;
                  width:      20em;
                  height:     30em;
            }
      }
      .karakter {
        animation-name: lopen;
        animation-duration: 5s;
            animation-timing-function: ease;
            animation-iteration-count: infinite;
      }
      </style>
  </head>
  <body>
            <main>
                  <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/tingle.png" width="50"
height="75" class="karakter" />
            </main>
      </body>
</html>
```



Als het goed is zie je het mannetje van klein naar groot veranderen.

Zorg ervoor dat de animatie van het karakter in het bovenstaande voorbeeld niet enkel de transformatie van klein naar groot betreft, maar ook weer andersom.

Tip: maak extra keyframes aan.

Mocht je het mannetje ook willen verschuiven dan zal de postion relative of absolute moeten worden ipv static, zie hoofdstuk 6

Overzicht van animation eigenschappen:

- animation: de verkorte notatie voor alle volgende punten tezamen
- animation-delay: na hoeveel seconden (s) of milliseconden (ms) dient de animatie te beginnen?
- animation-direction: begint de animatie bij 100% of bij 0%? Ofwel, gaat hij van begin naar eind of van het eind naar het begin?
- animation-duration: hoe lang gaat de animatie duren in seconden(s) of milliseconden (ms)?
- animation-fill-mode: eindigt de animatie aan het begin of aan het einde?
- animation-iteration-count: hoe vaak dienen we de animatie te herhalen?
- animation-name: wat is de naam van de animatie én dus ook het bijbehorende keyframes(!)
- animation-play-state: begint de animatie direct of staat hij standaard op pauze?
- animation-timing-function: in welke snelheid spelen wij de animatie af?
- default is: 'ease', ofwel: de animatie begint langzaam, versnelt dan en vertraagd weer naar het einde. In de praktijk zullen we eerder 'linear' gebruiken om (bijvoorbeeld) een personage vloeiender te laten bewegen.

Voor meer animation-timing-function waarden, kijk bijvoorbeeld op de w3c site: https://www.w3schools.com/cssref/css3_pr_animation-timing-function.asp.

4.8 Samenvatting

Hoe een element binnen een tekst wordt weergegeven hangt af van de display eigenschap. Een veelgebruikte toepassing daarvan is het horizontaal plaatsen van list items in een menu. Verder zijn in dit hoofdstuk kleuren en achtergrondplaatjes behandeld. Achtergrondplaatjes worden standaard herhaald weergegeven, zowel verticaal als horizontaal. Ook dit is weer met CSS aan te passen. Tot slot hebben we geleerd animaties te maken en objecten te transformeren.

5 Padding, margin en borders

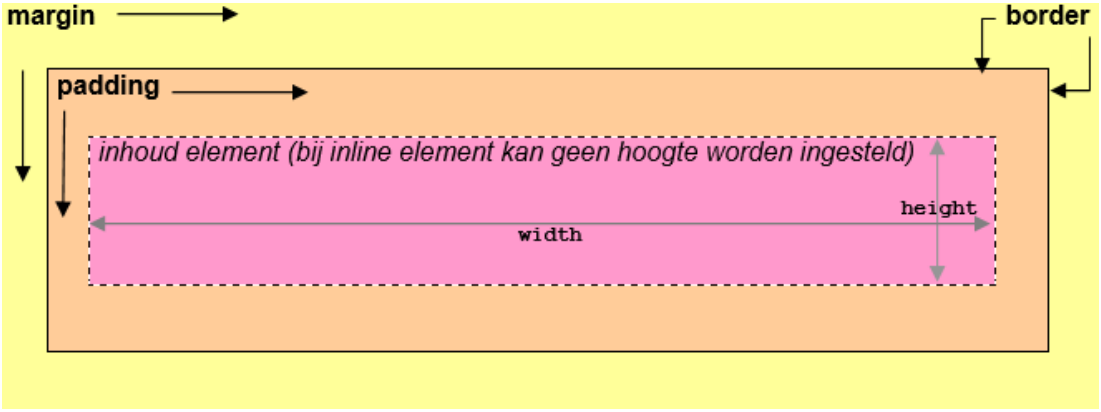
5.1 Inleiding

In dit hoofdstuk behandelen we hoe we de positie en grootte van elementen in een webpagina kunnen beïnvloeden. Daarbij komt ook de afstand tussen elementen onderling en de orde, en de ruimte en tussen de inhoud van een element en de buitenrand.

- Het boxmodel inclusief display eigenschappen kunnen beschrijven
- Margin, padding en borders kunnen instellen op elementen
- Het kunnen centreren van content en elementen

5.2 Ruimtelijke opbouw van elementen

Elementen zijn de basis van een document. Met behulp van CSS kunnen we verschillende ruimtelijke eigenschappen van een element beïnvloeden. Deze ruimtelijk eigenschappen worden in onderstaande figuur weergegeven.



We kunnen hier dus een *border* (kader), *margin* en *padding* onderscheiden. Wat deze begrippen in detail inhouden wordt in de volgende paragrafen duidelijk.

Volgens de CSS definitie horen we te weten dat elk element op het scherm een onzichtbaar kader om zich creëert.

- De *border* is het kader dat om het element heen loopt. De borderdikte kan ingesteld worden vanaf 0 pixels. De afstand tussen de border en de tekst van het element hangt af van de ingestelde padding.
- *Padding* is de ruimte tussen de inhoud van het element en het kader eromheen. Wanneer er geen kader (border) is ingesteld, zal deze nog altijd onzichtbaar aanwezig zijn met een breedte van 0 pixels.
- *Margin* is de ruimte tussen de border van het element en de border van aanliggende elementen.
- *Width* en *height* (bij block elementen) betreffen breedte en hoogte van de de inhoud van het element.

In de volgende paragrafen zullen we eerst standaard gedrag van de browser gaan bekijken en vervolgens dieper op borders, margin, padding en andere lay-out principes zoals *float* en *clear* ingaan.

5 Padding, margin en borders

Belangrijk is om te weten dat veel browsers bepaalde CSS eigenschappen al voor ons instellen. Denk aan een margin die al is ingesteld op paragrafen (dat wordt door de meeste browsers gedaan). Dit kan voor problemen zorgen indien wij er eigenlijk vanuit willen gaat dat wij met een 'schone lei' beginnen.

Om dit probleem (deels) tegen te gaan kunnen wij gebruik maken van CSS resets. Dit kan iets heel concreets zijn zoals: P { margin: 0px; }. Hierdoor wordt dat ons startpunt.

In de praktijk zijn er al vele CSS reset files ontwikkeld om met de bovengenoemde browser standaard CSS instellingen om te gaan. Eén van de meest gebruikte files hiervoor is Normalize.css. Meer hierover leest u in het hoofdstuk 'Frameworks'.

5.3 Borders

Een border is een lijn die om de inhoud en de padding van een element heen loopt. Een border kan ingesteld worden op de meeste HTML elementen. In de tabel hieronder kunt u zien op welke elementen een border kan worden ingesteld en of deze standaard al een breedte heeft.

Element	Border	Standaard border?	
A	Ja	Nee	
Br	Nee	Nee	
Div	Ja	Nee	
Img	Ja	Nee	
Input	Ja	Ja:	1px
P	Ja	Nee	
Select	Ja	Ja:	1px
Span	Ja	Nee	
Table	Ja	Ja:	1px
Textarea	Ja	Ja:	1px

Voor borders kunnen we drie individuele eigenschappen instellen: de vormgeving (style), de breedte (width) en de kleur (color). Hoe we dit precies kunnen realiseren ziet u hieronder.

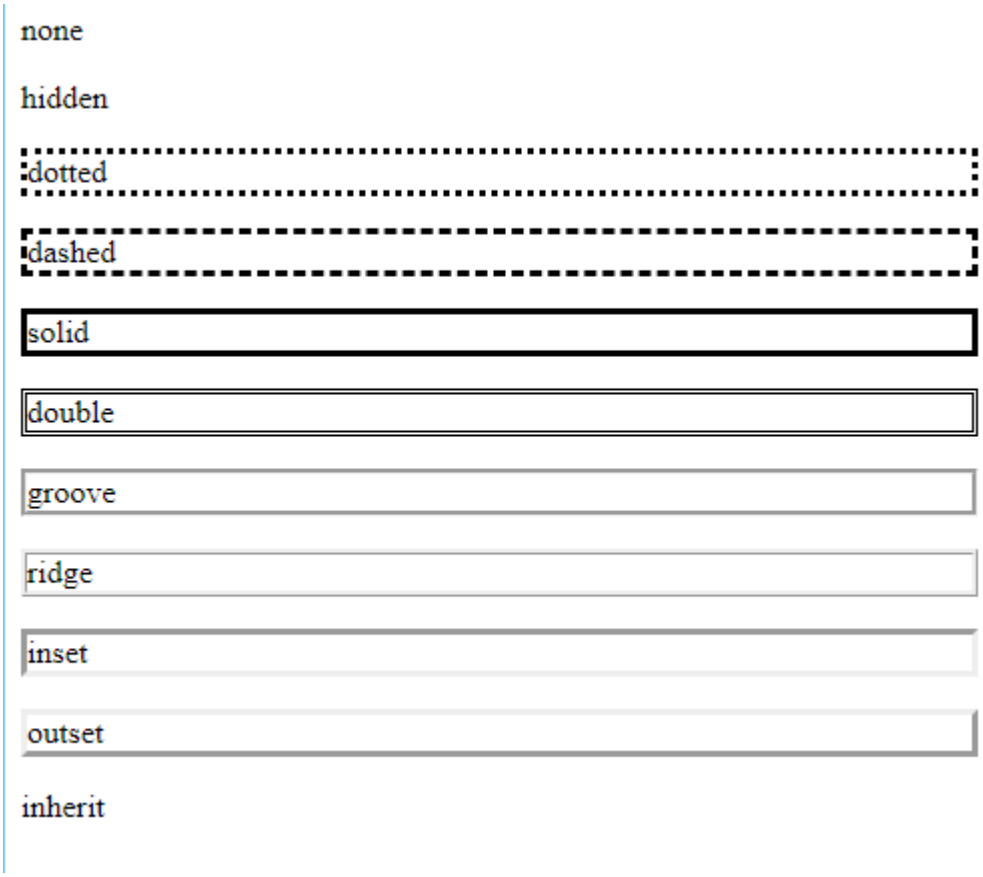
5.3.1 Border-style

Wanneer we een border willen instellen, is het handig als we dit naar eigen wens kunnen vormgeven. Daarvoor gebruiken we de border-style eigenschap. We kunnen hiermee bijvoorbeeld aangeven of het een doorlopend kader moet zijn, of een stippellijn.

border-style syntax
syntax: selector { border-style: sleutelwoord; }
sleutelwoord: none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit
default: geen, maar per zijde is het: ‘none’

Met elk sleutelwoord zien we een andere soort border weergegeven op een element. Hieronder staan de verschillende border-style voorbeelden vermeld:

```
<html>
      <head>
      <style>
    #none{  border-style: none;}
      #hidden{ border-style: hidden;      }
      #dotted{  border-style: dotted;     }
      #dashed{  border-style: dashed;     }
      #solid{  border-style: solid; }
      #double{  border-style: double;     }
      #groove{  border-style: groove;     }
      #ridge{  border-style: ridge; }
      #inset{  border-style: inset; }
      #outset{  border-style: outset;     }
      #inherit{  border-style: inherit;   }
      </style>
  </head>
      <body>
        <p id="none">none</p>
        <p id="hidden">hidden</p>
        <p id="dotted">dotted</p>
        <p id="dashed">dashed</p>
        <p id="solid">solid</p>
        <p id="double">double</p>
        <p id="groove">groove</p>
        <p id="ridge">ridge</p>
        <p id="inset">inset</p>
        <p id="outset">outset</p>
        <p id="inherit">inherit</p>
      </body>
</html>
```



Wanneer we geen border willen zien gebruiken we de code:

```
selector { border-style:     none; }
```

We hoeven niet voor iedere zijde van een element hetzelfde type border te gebruiken. We kunnen namelijk ook elke zijde apart benaderen. We gebruiken hiervoor dezelfde syntax, echter met als selector één van de volgende:

Border-top-style, border-right-style, border-bottom-style, of border-left-style

We kunnen op de bovenstaande manier ook de andere border eigenschappen instellen voor de individuele zijden van een element. Dus door gebruik te maken van de termen *top*, *right*, *bottom* en *left* kunnen we ook *de border-width en border-color* instellen die hierna worden besproken.
Daarnaast is er een shorthand notatie mogelijk, waarin met één regel verschillende waarden kunnen worden meegegeven. We kunnen één, twee, drie of vier waarden meegeven voor border-style, border-width of border-color. Deze hebben dan de volgende betekenis:

waarden	betrekking op	voorbeeld
een	gehele border	border-style:solid;
twee	1: top en bottom, 2: right en left	border-style: none solid; (alleen links en rechts een border)
drie	1: top 2: left en right 3: bottom	border-style: none none solid; (alleen onder een border)
vier	1: top 2: right 3: bottom 4: left	border-style: none solid none none; (alleen rechts een border)

5.3.2 Border-width

Wanneer we een kader willen, is het ook van belang dat we aangeven hoe ‘dik’ dit kader is. De meest gebruikte dikte is 1 pixel: de standaard breedte van elementen met een border (zie eerder genoemde tabel).

De syntax is:

border-width syntax

syntax: `selector { border-width: sleutelwoord; }`
sleutelwoord: `[ thin | medium | thick ]| <lengte> | inherit`
default: `geen, maar per zijde is het: ‘medium’`

We zien dat we een beschrijvend sleutelwoord kunnen opgeven, of een lengtewaarde (px, pc, cm, mm et cetera).

5.3.3 Border-color

Met border-color kunnen we borders een kleur geven. We kunnen ook hier weer onderscheid maken tussen de afzonderlijke zijden van het element.

border-color syntax

syntax: `selector { border-color: sleutelwoord; }`
sleutelwoord: `<kleur> | transparent | inherit`
default: `De ingestelde waarde van color voor het element.`

Welke kleuren we kunnen invullen voor het sleutelwoord kunt u teruglezen in hoofdstuk 4.

```
<html>
      <head>
      <style>
    #navigatie li {
        /* hier invullen */
      }
      </style>
  </head>
      <body>
    <ul id="navigatie">
        <li><a href="test.html">Home</a></li>
        <li><a href="producten.html">Producten</a></li>
        <li><a href="referenties.html">Referenties</a></li>
        <li><a href="forum.html">Forum</a></li>
        <li><a href="contact.html">Contact</a></li>
    </ul>
  </body>
</html>
```

Open het bovenstaande voorbeeld. We passen nu de CSS code aan, zodat elk li element een border krijgt aan de rechterkant. Daarnaast worden deze element naast elkaar geplaatst. Voeg de volgende CSS code toe:

```
 #navigatie li  {
        border-style:   none solid none none;  // alleen
aan rechterkant
      border-color:   blue;
      border-width:   1px;
      padding-right:  6px;
      display:        inline;
}
```

Bekijk het resultaat. Deze border is bedoeld als scheidingsteken tussen de menu items. Er staat aan de rechterkant dus nog een border teveel. We kunnen dit in moderne browsers oplossen met een nieuwe CSS selector:

```
#navigatie li:last-child {border-style:none;}
```

Een voorlopig veiliger alternatief is het laatste li element een klasse te geven, bijvoorbeeld "laatste", en het dan op vergelijkbare manier op te lossen:

```
#navigatie li.laatste {border-style:none;}
```

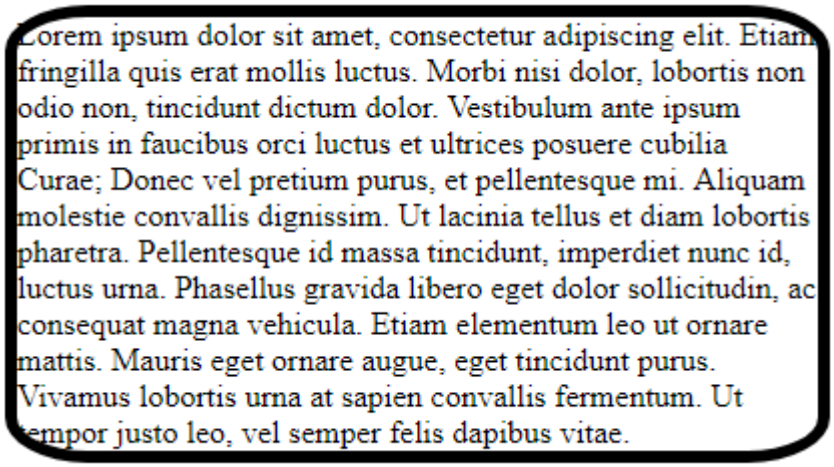
5.3.4 Afgeronde randen

Een erg welkome toevoeging aan CSS3 is geweest de border-radius eigenschap. Hiermee kunnen wij randen afronden. We kunnen de waarde opgeven voor alle hoeken in één keer, óf per hoek. We kunnen daarnaast ieder mogelijke maatvorm opnemen (px, em, %, enzovoorts).

Hoe ziet dit eruit? Zet bij het volgende voorbeeld de border-radius regel in commentaar (dus tussen de /* en de */). En bekijk het verschil:

```
<html>
      <head>
      <style>
    div {
        width:  400px;
        border: 6px solid black;
        border-radius: 10%;
      }
      </style>
  </head>
      <body>
  <div>
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Etiam fringilla quis erat mollis luctus. Morbi nisi dolor,
lobortis non odio non, tincidunt dictum dolor. Vestibulum
ante ipsum primis in faucibus orci luctus et ultrices
posuere cubilia Curae; Donec vel pretium purus, et
pellentesque mi. Aliquam molestie convallis dignissim. Ut
lacinia tellus et diam lobortis pharetra. Pellentesque id
massa tincidunt, imperdiet nunc id, luctus urna. Phasellus
gravida libero eget dolor sollicitudin, ac consequat magna
vehicula. Etiam elementum leo ut ornare mattis. Mauris eget
ornare augue, eget tincidunt purus. Vivamus lobortis urna
at sapien convallis fermentum. Ut tempor justo leo, vel
semper felis dapibus vitae.
  </div>
</body>
</html>
```

Resultaat zonder commentaar:



Haal de commentaartekens weer weg. Valt het op dat de letters in de hoeken nu wegvallen? Los dit op door de volgende CSS regel toe te voegen aan het opmaak van de DIV:
`padding: 10px;`

Dit zorgt ervoor dat er witruimte tussen de tekst en de rand van het element komt te staan. Meer daarover leest u verderop in deze module.

Een veel gebruikte toepassing van de border-radius eigenschap is bij het maken van invoervelden én zelfgemaakte buttons. Bekijk het volgende voorbeeld hiervoor:

```
<html>
      <head>
      <style>
    button {
  color: white;

        /* De volgende drie eigenschappen zijn nodig om een
button met afgeronden randen te maken: */
        background-color: gray;
        border: solid 10px gray;
        border-radius: 17px;
      }
      input {
        margin-right: 20px; /* wit ruimte náást het veld */
        padding-left: 35px; /* ruimte zodat we niet in het
icoon typen */
        width: 135px;

        border: none;
        border-bottom: solid 1px gray;
      }

      #username {
            background-image: url("https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/person_icon.png");

        background-repeat: no-repeat;
      }

      #password {
            background-image: url("https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/key_icon.png");

        background-repeat: no-repeat;
      }
      </style>
  </head>
<body>
  <form>
      <input type="text" id="username"
placeholder="Gebruikersnaam" />
      <input type="password" id="password"
placeholder="Wachtwoord" />
      <button>Registreer</button>
  </form>
</body>
</html>
```

5 Padding, margin en borders

5.3.5 Border achtergronden

We kunnen naast een gevulde border te gebruiken, met een bepaalde patroon (streep, onderbroken streep, stippen, enzovoorts), ook ervoor kiezen een achtergrond afbeelding in te stellen in onze border.

Wij gebruiken hiervoor de border-image eigenschap, met daar een url() aan meegegeven als waarde. Bekijk hiervoor het volgende voorbeeld ter illustratie:

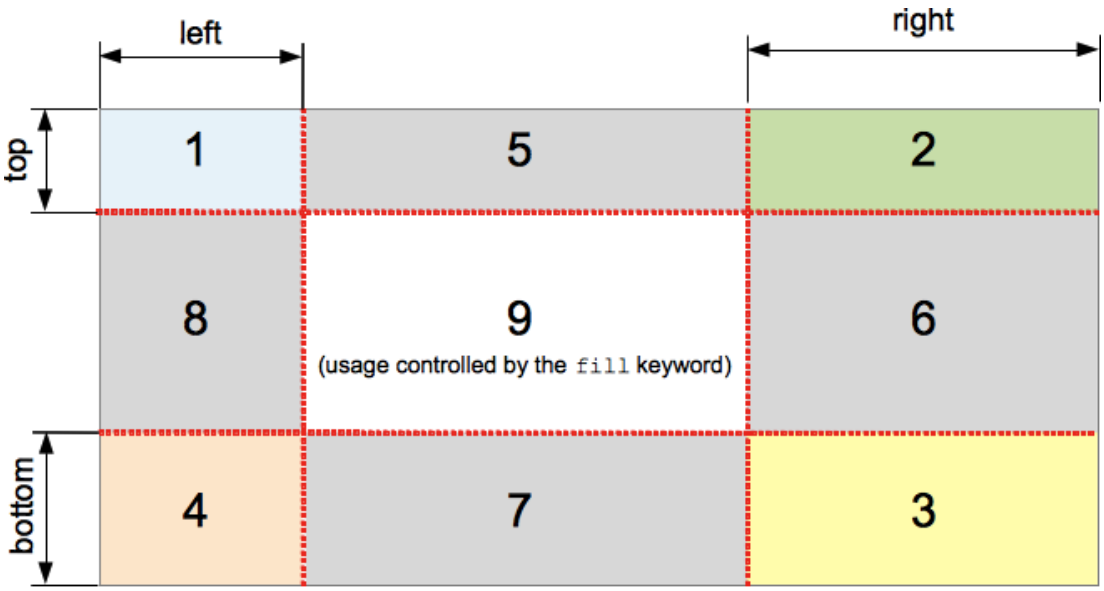
```
<html>
      <head>
      <style>
   button {
        color: white;

        /* De volgende drie eigenschappen zijn nodig om een
button met afgeronden randen te maken: */
        background-color: gray;
        border: solid 10px gray;
        border-radius: 17px;
      }

      input {
        margin-right: 20px; /* wit ruimte náást het veld */
        padding-left: 35px; /* ruimte zodat we niet in het
icoon typen */
        width: 135px;

        border: solid 10px;
        border-image: url("https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/congruent_pentagon.png")
20% round;
      }
      </style>
  </head>
  <body>
  <form>
      <input type="text" id="username"
placeholder="Gebruikersnaam" />
      <input type="password" id="password"
placeholder="Wachtwoord" />
      <button>Registreer</button>
  </form>
</body>
</html>
```

5 Padding, margin en borders

De border-image eigenschap is de korte weergave van de volgende eigenschappen:

- border-image-outset: hoe ver dient de image buiten de border uit te komen (default = 0)
- border-image-repeat: dient de achtergrond herhaald te worden weergegeven (default = stretch)
- border-image-slice: waar moet de achtergrond afbeelding afgeknipt worden langs de slice-lines (zie afbeelding hiernaast:)
- border-image-source: bron url van de afbeelding
- border-image-width: breedte van de border achtergrond afbeelding

Bekijk voor een gedetailleerde beschrijving van deze eigenschappen de MDN website.
https://developer.mozilla.org/nl/docs/Web/CSS/border-image



5.4 Margin en padding

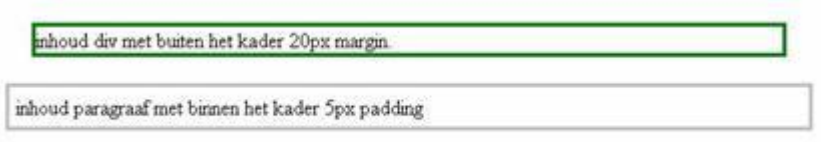
De visuele afstand tussen elementen wordt met twee eigenschappen ingesteld: margin en padding. Met margin geven we de afstand aan van een element tot een volgend element. Binnen elk element bestaat meestal ook ruimte tussen de inhoud en de border van het element. Dit wordt met padding ingesteld.

Wanneer we bijvoorbeeld tekst of plaatjes die binnen een element staan, verder van het elementkader willen plaatsen, dan moeten we hiervoor de *padding* verhogen. Indien we echter het gehele element (inclusief kader), verder van omringende elementen willen plaatsen, gebruiken we *margin*.

5 Padding, margin en borders

Padding en margin hebben voor veel elementen al een standaard waarde. Let hierop bij het indelen van pagina's. Zet ze bijvoorbeeld eerst op 0px.

In dit voorbeeld is gebruik gemaakt van padding op een paragraaf en van margin op een div:



We zien dus dat padding meer 'ademruimte' geeft aan het paragraaf element (binnen de borders). Terwijl we met margin meer ruimte om het gehele element creëren.

Wanneer margin of padding ingesteld worden op een element, heeft dat in veel gevallen effect op de positionering van andere elementen op de pagina.

De syntax voor het instellen van margin is:

margin syntax

syntax: selector { margin: sleutelwoord; }
sleutelwoord: <lengte> | inherit
default: browser afhankelijk

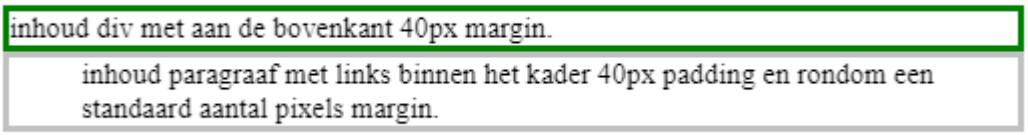
De syntax voor het instellen van padding is:

padding syntax

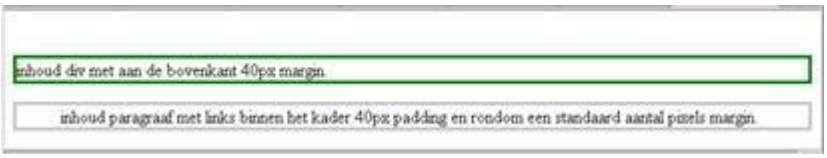
syntax: selector { padding: sleutelwoord; }
sleutelwoord: <lengte> | inherit
default: 0px

We kunnen zowel padding als margin ook voor individuele zijden van een element opgeven. We zien dit in het volgende voorbeeld. We hebben hier de div 40 pixels vanaf de bovenkant geplaatst.

```
<html>
      <head>
      <style>
   p {
        padding-left:    40px;
        border-style:    solid;
        border-color:    silver;
        margin:          0px;
      }

      div {
        margin-top:      40px;
        border-style:    solid;
        border-color:    green;
      }
      </style>
  </head>
 <body>
    <div>inhoud div met aan de bovenkant 40px margin.</div>
    <p>inhoud paragraaf met links binnen het kader 40px
padding en rondom een standaard aantal pixels margin.</p>
</body>
</html>
```



We zien dat de elementen nu boven elkaar liggen. Als we nu de regel margin: 0px voorzien van commentaar door er /* voor en */ achter te zetten, zal de standaard margin tevoorschijn komen:



Wanneer margin of padding ingesteld worden op een element, heeft dat in veel gevallen effect op de positionering van andere elementen op de pagina.

5 Padding, margin en borders

5.5 Box model

Het kennen van het boxmodel is van belang voor de grootte en positie ven elementen. Wanneer we elementen naar eigen wens willen plaatsen op een webpagina, kunnen gebruik maken van de *position* en de *float* eigenschap. Deze eigenschappen worden een ander hoofdstuk behandeld. Hier zullen we de grootte behandelen.

Binnen CSS kennen we verschillende soorten *boxes*. Dit is een onzichtbaar blok om elementen heen.

Element box
Elk element kan worden gezien als een box met – van buiten naar binnen – de marge, de border, de padding en de inhoud. Als we met CSS de positie van een element willen beïnvloeden, dan heeft dat betrekken op deze hele *element box*.

Containing block
De positie en grootte van een element box wordt meestal bepaald ten opzichte van het omliggende block element. Dit wordt de *containing block* genoemd. In het volgende voorbeeld is de div de containing block van het element h1:

```
<div>
 <h1>voorbeeld</h1>
</div>
```

Indien een element niet in een block element is geplaatst, dan is het *root element* (het html element) het containing block. Dit wordt ook wel het *initial containing* block genoemd. De grootte ervan wordt bepaald door het zichtbare deel van de pagina op het beeldscherm.

5.5.1 Widht en height

Voor inline elementen kunnen we alleen de breedte instellen, van block elementen ook de hoogte.

De syntax voor het instellen van breedte en hoogte is als volgt:

width en height syntax

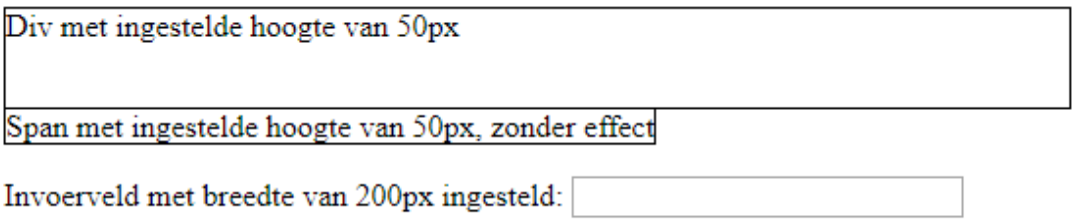
syntax:	selector { width: *sleutelwoord*; | height: *sleutelwoord* }
sleutelwoord:	auto | <lengte> | <percentage> | inherit
default:	`auto`

De standaard waarde is auto: CSS berekent dan zelf de grootte op basis van de inhoud. Een percentage wordt berekend op basis van het containing block. De lengte kan verder in de eerder genoemde meeteenheden worden uitgedrukt.

Houd er rekening mee dat de maten width en height betrekking hebben op de inhoud van een element: voor de totale grootte die een element inneemt komen daar nog de padding, border en marge bij.

Width en height hebben geen invloed op inline non-replaced elementen, zoals het span element, dit zien wij ook in het onderstaande voorbeeld:

```
<html>
      <head>
      <style>
      div, span {
        height:   50px;

        border: solid 1px black;
      }

      input {
        width: 200px;
      }
      </style>
  </head>
<body>
  <div>Div met ingestelde hoogte van 50px</div>
  <span>Span met ingestelde hoogte van 50px, zonder
effect</span>
  <p>Invoerveld met breedte van 200px ingesteld: <input
type="Text"/>
</body>
</html>
```



Voeg een nieuwe paragraaf toe -in- het div element. Zorg ervoor dat deze paragraaf altijd de helft van de breedte in beslag neemt van de div. Om het te testen kun je eventueel een kader (border) aan de nieuwe paragraaf toekennen.

Indien geen vaste breedte wordt opgegeven, maar auto of een percentage, dan kunnen we de hoogte en breedte nog begrenzen met vaste waarden voor min-width, min-height, max-width en/of max-height.

Een kolom met een breedte van 20% kan bijvoorbeeld al gauw te smal worden weergegeven op schermen met een lage resolutie. Door voor zulke gevallen een min-width in te stellen, bijvoorbeeld op 300px, kunnen we dergelijke problemen voorkomen.

5.5.2 resize

Om elementen qua afmetingen variabel te maken naar wens van de gebruiker, kunnen wij de CSS eigenschap 'resize' gebruiken. Indien deze is ingesteld, kan de gebruiker afhankelijk van de instelling ervoor kiezen de lengte, breedte of beide van een element te wijzigen.

De resize eigenschap heeft als mogelijke waarden: none (dit is de default waarde), horizontal, vertical en both. Allen dienen ze voor het gelijknamige wijzigen van het doel element.

Dit ziet er als volgt uit:

```
<html>
      <head>
      <style>
      div {
        width: 200px;
        height: 200px;

        border: solid 1px black;

        overflow: auto;

        resize: horizontal;
      }
      </style>
  </head>
   <body>
    <div>
    Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Donec et eros vel orci sollicitudin elementum ut ac
neque. Pellentesque sem tellus, vehicula vel enim eu,
suscipit scelerisque lectus. Suspendisse in purus sed magna
porttitor luctus. Fusce massa est, pulvinar eu velit eu,
varius venenatis lorem. Etiam sodales id risus quis congue.
In tortor velit, facilisis non quam non, aliquam
ullamcorper quam. Maecenas vitae faucibus mi.
    </div>
  </body>
</html>
```

Is het u in bovenstaande voorbeeld opgevallen dat ook de eigenschap 'overflow' op 'auto' is ingesteld? Het is noodzakelijk de overflow op een waarde anders dan 'visible' te zetten, want bij visible heeft resize geen effect. Wanneer dan de with en height te klein zijn zal de de tekst buiten het kader doorlopen.

5.6 Samenvatting

We hebben geleerd wat het box model is, hoe padding en margin werken en welke effecten deze hebben op de breedte en hoogte. Ook borders zijn uitgebreid aan bod gekomen, waarbij we duidelijk hebben kunnen zien welke soorten borders er zijn, welke elementen er standaard een border hebben ingesteld en hoe wij deze kaders vervolgens kunnen aanpassen.

6 Positionering

6.1 Inleiding

In dit hoofdstuk zullen wij één van de belangrijkste zaken leren bij het opmaken van een webpagina: het positioneren van elementen. Wij zullen zien dat dit op verschillende manieren kan, afhankelijk van het gewenste doel. Wij zullen als eerste het begrip content overflow kunnen uitleggen, static, absolute, relatieve, fixed en float positionering kunnen toepassen. Tot slot zullen wij de beginselen leren van het opmaken van een pagina middels flexbox positionering.

- Het kunnen verklaren en sturen van content overflow
- Het kunnen indelen van een pagina met behulp van static, absolute, relative, fixed en float element positionering
- Het kunnen indelen van een pagina met behulp van flexbox positionering
- Het kunnen benoemen van de voordelen van grid positionering

6.2 Positionering

Waar een element wordt geplaatst wordt standaard bepaald door de positie en marges van de overige elementen, en van de *document flow*. De document flow geeft aan hoe de volgorde van elementen in een web pagina van invloed is op de positie en grootte van die elementen. Zo zal een div element die in een HTML pagina na een p element is geplaatst, op het scherm onder dat p element worden weergegeven.

Met *position* kunnen we deze manier van positioneren van elementen aanpassen.
De syntax voor *position* is als volgt:

Position syntax:

syntax: `selector { position:`*`sleutelwoord`*`; }`
sleutelwoord: `static | absolute | fixed | relative | inherit`
default: `static`

Hieronder staan de vier sleutelwoorden beschreven die we kunnen gebruiken met de position eigenschap:

Static positionering
Dit is de standaard positionering. De positie van elementen wordt bepaald door de document flow van de elementen, en van de volgorde van die elementen. Bij andere vormen van positionering kunnen we met de afstand van een element ten opzichte van de containing block instellen met waarden voor *top, bottom, left* en *right* (waarover verderop meer). Bij static positionering kunnen dergelijke waarden niet worden gebruikt.

Relatieve positionering
Ook bij relatieve positionering gaan we uit van de gewone document flow. We kunnen de elementen hierbij echter verschuiven ten opzichte van hun normale positie. Dit doen we met behulp van maten voor *top, right, bottom* of l*eft*. Een positieve waarde voor top betekent bijvoorbeeld dat er aan de bovenkant meer ruimte komt: het element verplaatst dan naar

beneden. Ook negatieve waarden zijn toegestaan: een negatieve waarde voor *left* betekent dus minder ruimte aan de linker kant: het element verschuift dan naar links.

```
<html>
      <head>
      <style>
            div, p {
        margin:4px;
        padding:4px;
        border-style:solid;
        border-width:1px;
      }

      p {
        background-color: #EEEEEE;
      }
      </style>
  </head>
    <body>
        <div >
                  <p id="p1">Hier staat de eerste
paragraaf.<br>
                  Deze tekst bevat ook een plaatje:
      <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/3310.png" width="50"
height="100"/>
                  die in de document flow deel uitmaakt van
<span>de tekst</span></p>
                  <p id="p2">
                  Dit is de tweede paragraaf.
                  </p>
        </div>
  </body>
</html>
```

Bekijk de code. De elementen p en div hebben een border en ze hebben enige ruimte gekregen met behulp van padding en margin. Verder hebben ze een default (static) position.

We zullen nu de positie van het plaatje wijzigen. We moeten daarvoor eerst de position op relative zetten, en vervolgens de ruimte aan de bovenkant vergroten, of de ruimte onder het plaatje verkleinen.

Voeg de volgende code toe aan het bovenstaande voorbeeld:

```
img {
   position:relative;
   top:5px;
}
```

Wijzig daarna top: 5px; in bottom: -5px; en merk op dat de positie gelijk blijft.
Merk op dat we dit effect ook hadden kunnen bereiken met vertical-align:middle;

Wijzig de code daarna als volgt en bekijk het resultaat.

```
img {
   position: relative;
   top: 40px;
   right: 40px;
}
```

Het lijkt er nu op dat het plaatje is verplaatst naar de tweede paragraaf. Elementen kunnen dus voorbij de grenzen van hun 'containing box' worden verplaatst.

6.2.1 Absolute positionering

Met absolute positionering kunnen we aangeven dat een element een vaste positie ten opzichte van het containing block. Dit containing block moet dan wel een position hebben die niet static is. Als zo'n parent element niet wordt gevonden, wordt het initial containing block (het html element) als referentie genomen.

In het vorige voorbeeld staat een span element binnen het eerste p element (id=p1).

Voeg de volgende code toe en bekijk het resultaat:

```
#p1 span {position:absolute; top:0; right:0;}
```

Hiermee zetten we het span element rechts bovenaan de pagina. Om het span element te positioneren ten opzichte van het p element, moeten we het p element een andere positie dan static geven. We maken er daarom een relatieve positie van, zonder deze te verschuiven. De document flow blijft dan gehandhaafd, en het p element kan nu gebruikt worden als basis voor de positie van het span element.

Voeg nu de volgende code toe en bekijk het resultaat.

#p1 {position:relative; }

Zoals verwacht zien we het span element nu rechts bovenaan het p element staan. De tekst staat echter wat hoger dan de rest van de tekst (verklein eventueel het browser window om dit te controleren). Dit komt doordat we de span uit de p hebben gehaald. Het p element had een padding van 4px: dit moeten we nu dus ook voor het span element instellen om de tekst op dezelfde hoogte te houden.

Geef het span element een padding van 4px en controleer of de tekst nu op dezelfde hoogte staat als de bovenste tekst in het p element.

6.2.2 Fixed positionering

Deze vorm van positionering lijkt veel op absolute positionering. Het element zal geen deel meer uitmaken van de document flow. Er wordt bij fixed positioning niet uitgegaan van het omliggende element voor het bepalen van de positie, maar van het zichtbare deel van het browser venster.

Het verschil tussen position:fixed en position:absolute met initial containing block als referentie wordt duidelijk op grote webpagina's, waarbij scrollbars nodig zijn om de rest van de informatie te zien.

```
<html>
      <head>
      <style>
            body {
              margin-top:       200px;
              font-size:        20px;
              background-color: gray;
            }

            #p1, #p2 {
              width:             300px;
              height:            300px;
              position:          absolute;
            }

            #p1 {
              top:               50px;
              left:              50px;
              background-color:  orange;
            }

            #p2 {
              top:               150px;
              left:              200px;
              background-color:  blue;
              color:             #FFF;
            }
      </style>
  </head>
    <body>
  <p id="p1">paragraaf op positie:<br> top: 50px, left:
50px</p>
  <p id="p2"> paragraaf op positie:<br> top: 150px, left:
200px</p>
  <p>
    tekst<br><br>  tekst<br><br>  tekst<br><br>
tekst<br><br>  tekst<br><br>
    tekst<br><br>  tekst<br><br>  tekst<br><br>
tekst<br><br>  tekst<br><br>
    tekst<br><br>  tekst<br><br>  tekst<br><br>
tekst<br><br>  tekst<br><br>
    tekst<br><br>  tekst<br><br>  tekst<br><br>
tekst<br><br>  tekst<br><br>
    tekst<br><br>  tekst<br><br>  tekst<br><br>
tekst<br><br>  tekst<br><br>
  </p>
</body>
</html>
```

6 Positionering

In de code hierboven staan twee p elementen in met position:absolute. Als we naar beneden scrollen verdwijnen beide p elementen uit beeld.

Wijzig position:absolute in position:fixed. Probeer opnieuw uit wat er gebeurt bij het scrollen door de pagina. In dit geval blijven de p elementen op dezelfde positie staan, en gaat de tekst er onderdoor.
Een fixed positie wordt onder meer voor kop en voetteksten gebruikt. Denk ook aan 'vervelende' reclameblokken die mee scrollen met de pagina en altijd in beeld blijven. Maar óók zaken als handige uitklap menu's die mee scrollen aan de linker of rechterzijde van een pagina:

Internet Explorer ondersteunt position:fixed pas vanaf versie 7.

6.3 Float

Waarschijnlijk kent u het *align* HTML attribuut dat regelmatig werd gebruikt op plaatjes (<img>)om deze links of rechts op een pagina te positioneren. De tekst liep er dan keurig omheen. Deze image werd dan een zogeheten *float* element. Het image element was het enige type element dat kon *floaten*. Met CSS kunnen we ook andere elementen zo'n float eigenschap meegeven. Dit is een erg nuttige toepassing bij de positionering van elementen. Vanaf hier zullen we naar een element dat een float eigenschap heeft verwijzen met *float element.*

De *float* eigenschap voor een element kan als volgt worden ingesteld:

float syntax

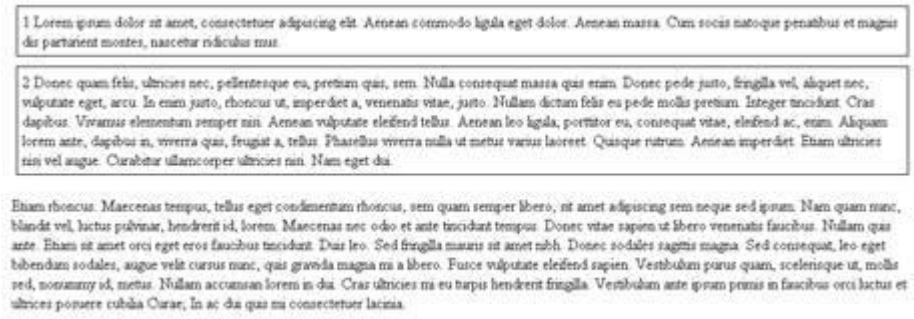
syntax:	`selector { float: sleutelwoord; }`
sleutelwoord:	`left | right | none | inherit`
default:	`none`

We zien dat bij een element standaard `float: none` staat ingesteld.

Een element waarbij de *float* is ingesteld genereert een block box, ongeacht het type element (inline of block). Als we dus een *float* element van een span maken, dan zal deze worden weergegeven alsof het een div is. Deze omschakeling gebeurt ook met het instellen van een *display* eigenschap zoals in een eerder hoofdstuk is beschreven.
Het float element wordt vervolgens uit de document flow naar links of naar rechts verplaatst, tot de grens van het containing block, of tot de buitengrens van het volgende float element.

```
<html>
      <head>
      <style>
            p {
              padding:5px;
            }

            div {
              border: solid 1px;
              margin:10px;
              padding:5px;
            }
      </style>
  </head>
<body>
      <div id="d1">1 Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Aenean commodo ligula eget
dolor. Aenean massa. Cum sociis natoque penatibus et magnis
dis parturient montes, nascetur ridiculus mus. </div>

      <div id="d2">2 Donec quam felis, ultricies nec,
pellentesque eu, pretium quis, sem. Nulla consequat massa
quis enim. Donec pede justo, fringilla vel, aliquet nec,
vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet
a, venenatis vitae, justo. Nullam dictum felis eu pede
mollis pretium. Integer tincidunt. Cras dapibus. Vivamus
elementum semper nisi. Aenean vulputate eleifend tellus.
Aenean leo ligula, porttitor eu, consequat vitae, eleifend
ac, enim. Aliquam lorem ante, dapibus in, viverra quis,
feugiat a, tellus. Phasellus viverra nulla ut metus varius
laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies
nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam
eget dui.</div>

      <p id="p1">Etiam rhoncus. Maecenas tempus, tellus
eget condimentum rhoncus, sem quam semper libero, sit amet
adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel,
luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et
ante tincidunt tempus. Donec vitae sapien ut libero
venenatis faucibus. Nullam quis ante. Etiam sit amet orci
eget eros faucibus tincidunt. Duis leo. Sed fringilla
mauris sit amet nibh. Donec sodales sagittis magna. Sed
consequat, leo eget bibendum sodales, augue velit cursus
nunc, quis gravida magna mi a libero. Fusce vulputate
```

```
eleifend sapien. Vestibulum purus quam, scelerisque ut,
mollis sed, nonummy id, metus. Nullam accumsan lorem in
dui. Cras ultricies mi eu turpis hendrerit fringilla.
Vestibulum ante ipsum primis in faucibus orci luctus et
ultrices posuere cubilia Curae; In ac dui quis mi
consectetuer lacinia.</p>

      <p id="p2">Nam pretium turpis et arcu. Duis arcu
tortor, suscipit eget, imperdiet nec, imperdiet iaculis,
ipsum. Sed aliquam ultrices mauris. Integer ante arcu,
accumsan a, consectetuer eget, posuere ut, mauris. Praesent
adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Nunc
nonummy metus. Vestibulum volutpat pretium libero. Cras id
dui. Aenean ut eros et nisl sagittis vestibulum. Nullam
nulla eros, ultricies sit amet, nonummy id, imperdiet
feugiat, pede. Sed lectus. Donec mollis hendrerit risus.
Phasellus nec sem in justo pellentesque facilisis. Etiam
imperdiet imperdiet orci. Nunc nec neque. Phasellus leo
dolor, tempus non, auctor et, hendrerit quis, nisi.</p>
</body>
</html>
```



Bekijk de bovenstaande code. De bovenste twee divs hebben een border. We gaan de float eigenschap hiervan uitproberen.

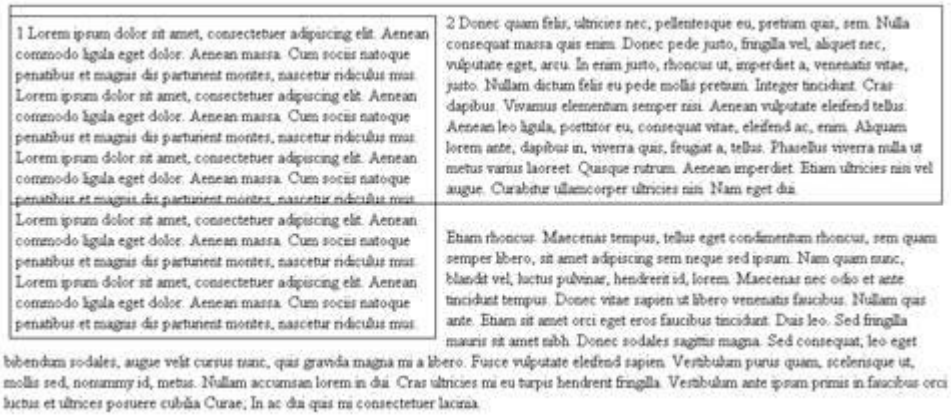
t
Float elementen krijgen een breedte mee, omdat het effect anders onvoorspelbaar is, en afhankelijk van de gebruikte browser. Voeg de volgende CSS code oe aan het bovenstaande voorbeeld en bekijk het resultaat:

```
#d1 {width: 400px; float:left;}
```

We zien nu dat het eerste div element links is uitgelijnd, en dat de tekst van het tweede div element er rechts omheen komt te staan. De daarop volgende tekst wordt daar rechts omheen geplaatst.

We zien ook dat de border van de tweede div links om die van de eerste div wordt geplaatst. Dat geldt echter niet voor de onderkant: als het float element hoger is dan het volgende element, wordt de border van dat volgende element door het float element geplaatst, zoals in het volgende voorbeeld (u hoeft dit nu niet uit te voeren):

6 Positionering



Zorg er nu voor dat zowel #d1 als #d2 de waarde float:left krijgt en bekijk het resultaat. (NB zorg ervoor dat zowel de HTMLcode als de CSScode "uit" staan en de browser de volle breedte heeft.)

```
#d1, #d2 {width: 400px; float:left;}
```

We zien nu dat de tweede div rechts van de eerste div wordt geplaatst en dat de daarop volgende tekst daar omheen komt te staan.

Probeer daarna uit wat er gebeurt als beide divs met float:right worden gepositioneerd.

Om goed alle aspecten van floating elementen te kunnen begrijpen, zijn er een aantal regels opgesteld die het gebruik ervan uiteindelijk zullen vereenvoudigen. We zullen onderstaand de belangrijkste regels beschrijven.

1. Een float element kan horizontaal niet buiten de kaders van het containing block komen en ligt dan dus bij een maximale breedte tegen het kader aan van het containing block.
2. Als een float element op float:left staat en een ander element op dezelfde verticale hoogte ook op float:left staat, dan komen ze naast elkaar te staan, waarbij het eerste element helemaal links tegen de rand van het containing block staat en het tweede element tegen het rechter kader van het eerste element.
3. Wanneer we meerdere float elementen naast elkaar willen plaatsen, maar de totale breedte van de elementen (bijvoorbeeld: 400+300+400=1100) groter is dan de breedte van het scherm (uitgaande van 1024 pixels), dan zal het laatste element naar beneden verplaatst worden om overlapping te voorkomen. Het element dat verplaatst wordt zal dan met zijn bovenrand tegen de onderrand van het element komen dat nu helemaal rechts uitgelijnd staat. Het verplaatste element zal dan dus ook niet automatisch naar links uitlijnen.

6.3.1 Clearing

Wanneer we niet willen dat bepaalde inhoud van een pagina om een float element heen loopt, kunnen we de clear CSS eigenschap gebruiken. Wanneer we deze eigenschap op een element toepassen en dit element ligt naast een float element, dan zal het element naar beneden verschuiven tot hij onder het andere float element ligt.

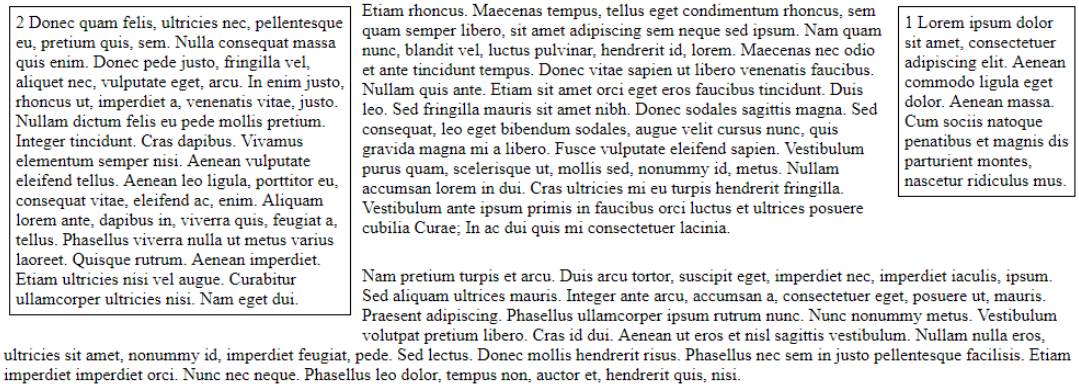
clear syntax

syntax:	`selector { clear: sleutelwoord; }`
sleutelwoord:	`left | right | both | none | inherit`
default	`none`

We kunnen als sleutelwoord kiezen voor left, right, both of none. Wanneer we voor left of right kiezen, dan zal alleen aan die zijde niet om het float element worden gelopen.
Bekijk het de code van het volgende voorbeeld. Open dit voorbeeld in een scherm op volle breedte. We zien hier tekst met twee float elementen.

```
<html>
      <head>
      <style>
            p {
              padding:5px;
            }

            div {
              border: solid 1px;
              margin:10px;
              padding:5px;
            }

            #d1 {
              width:150px; float:right;
            }

            #d2 {
              width:300px; float:left;
            }
      </style>
  </head>
<body>
      <div id="d1">1 Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Aenean commodo ligula eget
dolor. Aenean massa. Cum sociis natoque penatibus et magnis
dis parturient montes, nascetur ridiculus mus. </div>

      <div id="d2">2 Donec quam felis, ultricies nec,
pellentesque eu, pretium quis, sem. Nulla consequat massa
quis enim. Donec pede justo, fringilla vel, aliquet nec,
vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet
a, venenatis vitae, justo. Nullam dictum felis eu pede
mollis pretium. Integer tincidunt. Cras dapibus. Vivamus
elementum semper nisi. Aenean vulputate eleifend tellus.
Aenean leo ligula, porttitor eu, consequat vitae, eleifend
ac, enim. Aliquam lorem ante, dapibus in, viverra quis,
feugiat a, tellus. Phasellus viverra nulla ut metus varius
laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies
nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam
eget dui.</div>
```

```
      <p id="p1">Etiam rhoncus. Maecenas tempus, tellus
eget condimentum rhoncus, sem quam semper libero, sit amet
adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel,
luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et
ante tincidunt tempus. Donec vitae sapien ut libero
venenatis faucibus. Nullam quis ante. Etiam sit amet orci
eget eros faucibus tincidunt. Duis leo. Sed fringilla
mauris sit amet nibh. Donec sodales sagittis magna. Sed
consequat, leo eget bibendum sodales, augue velit cursus
nunc, quis gravida magna mi a libero. Fusce vulputate
eleifend sapien. Vestibulum purus quam, scelerisque ut,
mollis sed, nonummy id, metus. Nullam accumsan lorem in
dui. Cras ultricies mi eu turpis hendrerit fringilla.
Vestibulum ante ipsum primis in faucibus orci luctus et
ultrices posuere cubilia Curae; In ac dui quis mi
consectetuer lacinia.</p>

      <p id="p2">Nam pretium turpis et arcu. Duis arcu
tortor, suscipit eget, imperdiet nec, imperdiet iaculis,
ipsum. Sed aliquam ultrices mauris. Integer ante arcu,
accumsan a, consectetuer eget, posuere ut, mauris. Praesent
adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Nunc
nonummy metus. Vestibulum volutpat pretium libero. Cras id
dui. Aenean ut eros et nisl sagittis vestibulum. Nullam
nulla eros, ultricies sit amet, nonummy id, imperdiet
feugiat, pede. Sed lectus. Donec mollis hendrerit risus.
Phasellus nec sem in justo pellentesque facilisis. Etiam
imperdiet imperdiet orci. Nunc nec neque. Phasellus leo
dolor, tempus non, auctor et, hendrerit quis, nisi.</p>
</body>
</html>
```



Stel dat we in dit voorbeeld met alleen de tweede paragraaf onder de rechter div willen plaatsen, maar nog wel tegen de linker div aan. Hiervoor kunnen we clear:right meegeven aan de tweede paragraaf.

6 Positionering

Voeg de volgende code toe. Bekijk het resultaat -fullscreen- door naar deze pagina te gaan om het effect goed te kunnen zien!

```
#p2 {clear:right;}
```

We zien nu dat de tweede paragraaf verplaatst is onder de rechter div. Om deze paragraaf onder beide divs te verplaatsen kunnen we clear:both meegeven, in plaats van clear:right. Probeer dit uit.

6.4 Flexbox

Flexbox is vanaf CSS3 een alternatieve manier om webpagina's op te maken. Hoewel oorspronkelijk vooral bedoeld voor het netjes uitlijnen van o.a. formulieren (als alternatief voor tabellen), wordt dit systeem nu veel toegepast om een opmaak te creëren die zich ook goed aanpast aan andere scherm formaten.

Het flexbox systeem bestaat altijd uit een flexbox container en flex elementen daar binnen. Een flexibel (flex) element kan zelf weer andere flex elementen bevatten. Een flex element kan in iedere richting worden geplaatst en verkleind zich qua formaat indien nodig. Dit maakt het systeem zeer veelzijdig, maar ook complex. Wees er daarom van bewust dat werken met het flexbox systeem de nodige oefening en ervaring vereist.

6.4.1 Flexcontainer('parent')

Om van een element een flexcontainer te maken, zullen we van het element de display eigenschap op 'flex' of 'inline-flex' moeten zetten. Dit wordt ook wel het maken van de flex formatting context genoemd. Hierbij zorgt de display waarde 'flex' voor een element op blokniveau, 'inline-flex' zorgt voor een container op inline niveau.

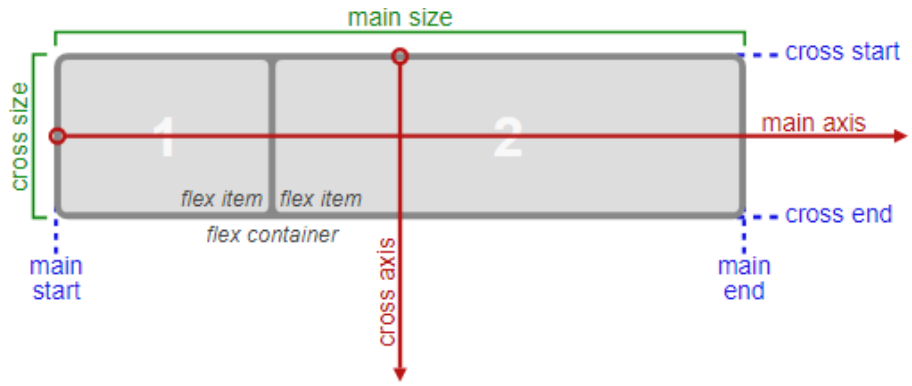
Ieder element in een flex container wordt als flex element behandeld. Maar pas zodra een element erbinnen ook de eigenschap 'flex' heeft ingesteld, kan deze zich ernaar gedragen. Deze elementen zullen vervolgens de beschikbare ruimte binnen de container zo optimaal als mogelijk benutten. De container bepaalt de richting van de elementen in het geheel; dus of ze horizontaal of verticaal ten opzicht van elkaar gepositioneerd worden.

Verderop in het hoofdstuk zullen wij ingaan op de items binnen een flex container.

Onderstaand ziet u een aantal eigenschappen van een flex container zelf:

flex-direction:	plaatsing v/d items in rijen of kolommen
flex-wrap:	plaatsing v/d items op (eventueel meerdere) regels
flex-flow:	korte notatie voor de flex-direction en flex-wrap

Wij zullen deze eigenschappen in de komende paragrafen aan de hand van duidelijke voorbeelden verder toelichten. Er volgt nu eerst een diagram waarin wij tonen hoe een container met virtuele lijnen is opgebouwd. Het is voor het begrip van het flexbox model van belang om deze termen te kennen en te begrijpen. Met name bij het uitlijnen van elementen binnen de container komt deze afbeelding weer terug. Voor nu is het voldoende hem gezien te hebben:



Afbeelding bron: W3C.

6.4.2 Flex-direction

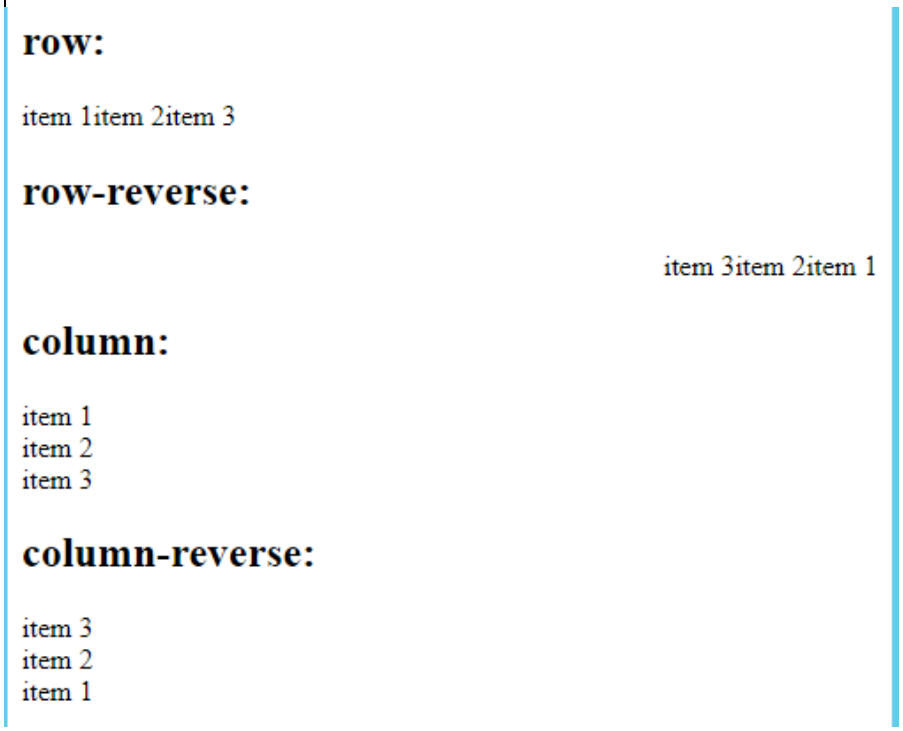
Met deze eigenschap stellen wij in in welke richting de child elementen van de flex container komen te staan. De standaard volgorde is van links naar rechts, maar met deze eigenschap kunnen wij het instellen op:

row:	een rij, van links naar rechts
row-reverse:	een rij, van rechts naar links
column:	een kolom van boven naar beneden
column-reverse:	een kolom van beneden naar boven

Onderstaand volgt een voorbeeld van de uitwerking van deze eigenschappen:

```
<html>
      <head>
      <style>
            #container1 {
              display: flex;
              flex-direction: row;
            }

            #container2 {
              display: flex;
              flex-direction: row-reverse;
            }

            #container3 {
              display: flex;
              flex-direction: column;
            }

            #container4 {
              display: flex;
              flex-direction: column-reverse;
            }
      </style>
  </head>
<body>
```

```
<main>
                  <h2>row:</h2>
                  <div id="container1">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                  </div>

                  <h2>row-reverse:</h2>
                  <div id="container2">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>

                  </div>

                  <h2>column:</h2>
                  <div id="container3">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                  </div>

                  <h2>column-reverse:</h2>
                  <div id="container4">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                  </div>
            </main>
</body>
</html>
```



Probeer in het bovenstaande voorbeeld zelf ook een margin en/of padding toe te voegen aan de elementen.

6.4.3 Flex-wrap

De flex container zal normaliter altijd proberen de verschillende child elementen (horizontaal) op dezelfde regel te tonen. Er wordt dan geprobeerd om de betreffende elementen gelijkmatig te verkleinen totdat ze wél op de regel passen. Meestal is dit mogelijk. Indien we de child elementen al voorzien hebben van (bijvoorbeeld) de eigenschap en waarde: flex: none; zal er een scrollbalk verschijnen om alle items op de regel te kunnen zien. Ze worden niet automatisch verkleind. Dat kan betekenen dat sommige elementen buiten het beeld kunnen vallen. Wanneer beide ongewenst is, de items mogen niet worden verkleind en niet horizontaal buiten beeld vallen dan is er de eigenschap flex-wrap:

no-wrap:	alle flex elementen komeen op één regel te staan (default)
wrap:	indien elementen buiten het zichtveld zouden komen, verschijnen ze op de volgende regel.
wrap-reverse:	hetzelfde als 'wrap', maar dan worden vanaf de onderste regel de regels gevuld.

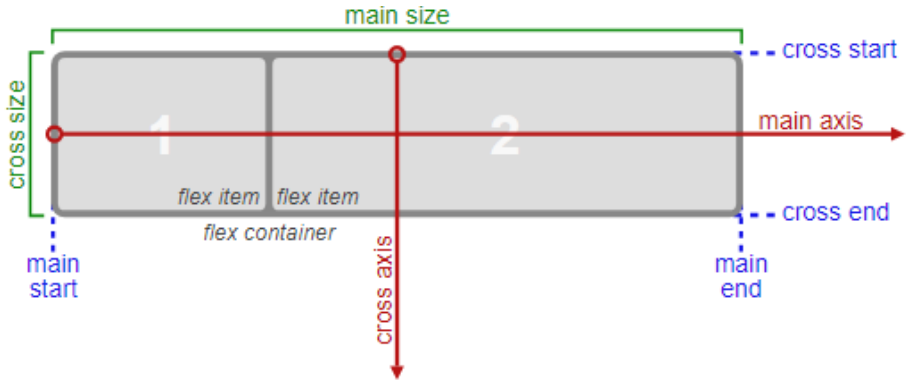
Bekijk het volgende voorbeeld op verschillende schermbreedtes voor het effect van de verschillende waarden van de 'flex-wrap' eigenschap. Let op dat de wrap waarden een gelijk effect kunnen hebben indien je dit voorbeeld op een device met een kleine viewport (klein zichtbaar scherm) bekijkt. Pas in dit bovenstaande voorbeeld de waarde van de eigenschap van 'flex-wrap' aan naar 'wrap-reverse', bekijk het resultaat en wijzig hem dan naar 'no-wrap' (dus de default waarde).

```
<html>
      <head>
      <style>
            #container1 {
              display: flex;
              flex-wrap: wrap; /* = usage maken van
ingesteld elem. width! */
            }

            #container1 div {
              border: 1px solid black;
              width:  400px;
            }
      </style>
  </head>
<body>
   <main>
            <h2>wrap:</h2>
            <div id="container1">
                  <div class="element1">item 1</div>
                  <div class="element2">item 2</div>
                  <div class="element3">item 3</div>
                  <div class="element4">item 4</div>
                  <div class="element5">item 5</div>
                  <div class="element6">item 6</div>
```

```
                    </div>
        </main>
</body>
</html>
```

6.4.4 Uitlijnen van flex elementen

Bij het uitlijnen van flex elementen in een flex container is het belangrijk het verschil te weten tussen de zogeheten 'main axis' en 'cross axis', ofwel de middenlijnen die respectievelijk horizontaal en verticaal door de container lopen. Onderstaande afbeelding heeft u aan het begin van dit hoofdstuk ook gezien. Kijk of deze ondertussen beter te plaatsen is. Vooral de voorbeelden in de volgende paragrafen zullen u hierbij helpen.



Afbeelding bron: W3C.

6.4.5 Uitlijnen met de justify-content eigenschap

Hoe kunnen wij elementen verspreiden binnen de container? Standaard gedrag is dat de elementen direct achter elkaar in het begin van de container worden weergegeven. Wij kunnen in de container aangeven dat de elementen een specifieke plaats binnen de container krijgen. Daarmee verdelen we overigens ook de witruimte gelijkmatiger.

Mogelijke waarden van de 'justify-content' eigenschap zijn: flex-start, center, flex-end, space-between en space-around.

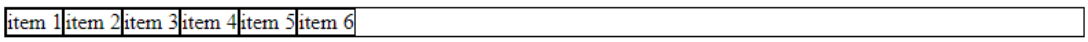
Probeer de code in het volgende voorbeeld uit en kijk of je alle waarden van de 'justify-content' eigenschap begrijpt.

```
<html>
      <head>
      <style>
            #container1 {
              display:   flex;
              justify-content: flex-start;
              border: solid 1px black;
            }

            #container2 {
              display:   flex;
              justify-content: center;
              border: solid 1px black;
```

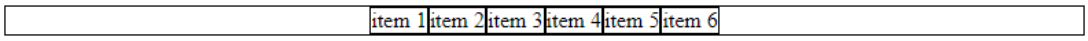
```
            }

            #container3 {
              display:   flex;
              justify-content: flex-end;
              border: solid 1px black;
            }

            #container4 {
              display:   flex;
              justify-content: space-between;
              border: solid 1px black;
            }

            #container5 {
              display:   flex;
              justify-content: space-around;
              border: solid 1px black;
            }

            #container1 div, #container2 div, #container3
div, #container4 div, #container5 div {
              border:    solid 1px black;
            }
      </style>
  </head>
<body>
   <main>
                  <h2>flex-start</h2>
                  <div id="container1">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

                  <h2>center</h2>
                  <div id="container2">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

                  <h2>flex-end</h2>
                  <div id="container3">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
```

```
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

      <h2>space-between</h2>
                  <div id="container4">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

                  <h2>space-around</h2>
                  <div id="container5">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>
            </main>
</body>
</html>
```

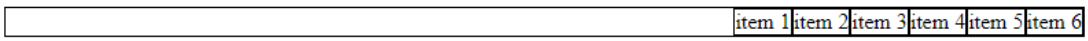
flex-start



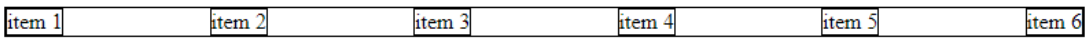
center



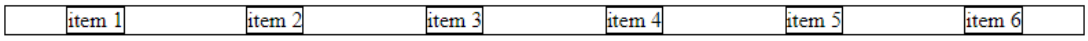
flex-end



space-between



space-around



6.4.6 Uitlijnen met de align-items eigenschap

Om elementen verticaal uit te lijnen op de zogeheten 'cross axis', kunnen wij gebruik maken van de 'align-items' eigenschap. Deze eigenschap kan de waarden: flex-start, center, flex-end, baseline of stretch (default) ingesteld hebben.

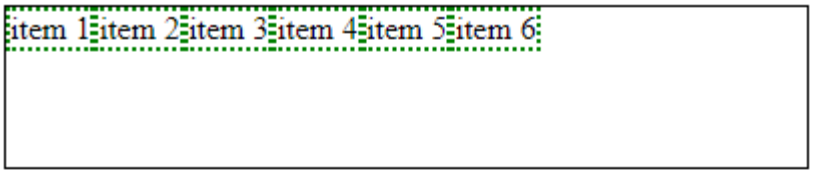
De container dient overigens een hoogte ingesteld te hebben, anders is er uiteraard geen witruimte om het effect te kunnen zien.

```
<html>
      <head>
      <style>
            #container1 {
              display:   flex;
              align-items: flex-start;

              width:     400px;
              height:    80px;
              border:    solid 1px black;
            }

            #container2 {
              display:   flex;
              align-items: center;

              width:     400px;
              height:    80px;
              border:    solid 1px black;
            }

            #container3 {
              display:   flex;
              align-items: flex-end;

              width:     400px;
              height:    80px;
              border:    solid 1px black;
            }

            #container4 {
              display:   flex;
              align-items: baseline;

              width:     400px;
              height:    80px;
              border:    solid 1px black;
            }

            #container5 {
              display:   flex;
              align-items: stretch;

              width:     400px;
              height:    80px;
              border:    solid 1px black;
            }

            #container1 div, #container2 div, #container3
div, #container4 div, #container5 div {
```

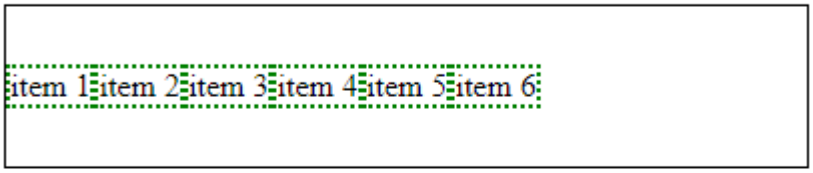
```
            border:    dotted 2px green;
          }
      </style>
  </head>
<body>
   <main>
                <h3>flex-start</h3>
                <div id="container1">
                      <div class="element1">item 1</div>
                      <div class="element2">item 2</div>
                      <div class="element3">item 3</div>
                      <div class="element4">item 4</div>
                      <div class="element5">item 5</div>
                      <div class="element6">item 6</div>
                </div>

                <h3>center</h3>
                <div id="container2">
                      <div class="element1">item 1</div>
                      <div class="element2">item 2</div>
                      <div class="element3">item 3</div>
                      <div class="element4">item 4</div>
                      <div class="element5">item 5</div>
                      <div class="element6">item 6</div>
                </div>

                <h3>flex-end</h3>
                <div id="container3">
                      <div class="element1">item 1</div>
                      <div class="element2">item 2</div>
                      <div class="element3">item 3</div>
                      <div class="element4">item 4</div>
                      <div class="element5">item 5</div>
                      <div class="element6">item 6</div>
                </div>

     <h3>baseline</h3>
                <div id="container4">
                      <div class="element1">item 1</div>
                      <div class="element2">item 2</div>
                      <div class="element3">item 3</div>
                      <div class="element4">item 4</div>
                      <div class="element5">item 5</div>
                      <div class="element6">item 6</div>
                </div>

                <h3>stretch</h3>
                <div id="container5">
                      <div class="element1">item 1</div>
                      <div class="element2">item 2</div>
                      <div class="element3">item 3</div>
                      <div class="element4">item 4</div>
                      <div class="element5">item 5</div>
```

```
                        <div class="element6">item 6</div>
                    </div>
                </main>
</body>
</html>
```

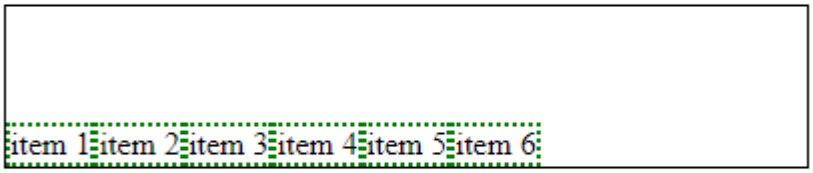
flex-start



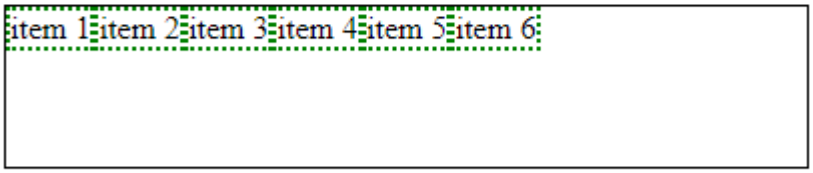
center



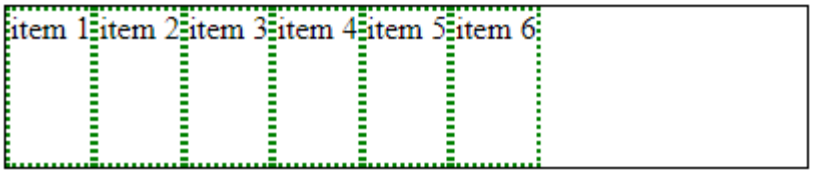
flex-end



baseline



stretch



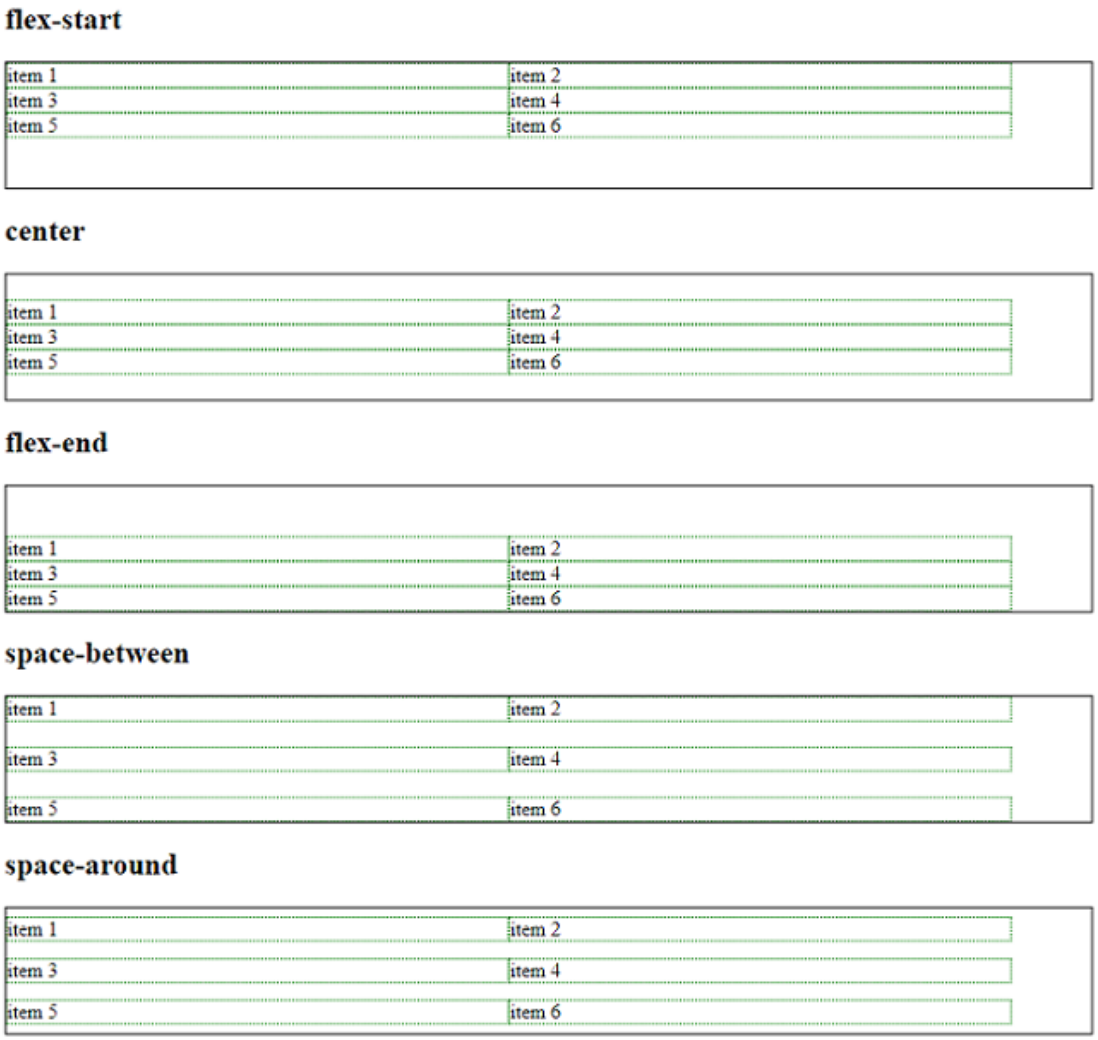
6.4.7 Uitlijnen met de align-content eigenschap

Voor het uitlijnen van de regels flex elementen indien ze op meer dan één regel staan, gebruiken wij de 'align-content' eigenschap . Hierdoor worden verticaal gezien de gehele regels binnen een container over de Y-as verspreid.

Om dit te verduidelijken bekijken wij het onderstaande voorbeeld. Let op dat deze eigenschap alleen werkt indien de elementen meer dan één regel in beslag nemen. De container dient overigens een hoogte ingesteld te hebben, anders is er uiteraard geen witruimte tussen de regels.

```
<html>
      <head>
      <style>
            #container1 {
              align-content: flex-start;
            }
            #container2 {
              align-content: center;
            }
            #container3 {
              align-content: flex-end;
            }
            #container4 {
              align-content: space-between;
            }
            #container5 {
              align-content: space-around;
            }
        #container1 , #container2 , #container3 ,
#container4 , #container5  {
               border:    solid 1px black;
                 display:   flex;
               flex-wrap: wrap;
                 height:    100px;
            }
            #container1 div, #container2 div, #container3
div, #container4 div, #container5 div {
              width:     400px;
              border:    dotted 1px green;
            }
      </style>
  </head>
<body>
   <main>
                  <h2>flex-start</h2>
                  <div id="container1">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

                  <h2>center</h2>
                  <div id="container2">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
```

```
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

                  <h2>flex-end</h2>
                  <div id="container3">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

      <h2>space-between</h2>
                  <div id="container4">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

                  <h2>space-around</h2>
                  <div id="container5">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>
            </main>
</body>
</html>
```



Speel weer met de grootte van je scherm om een gevoel te krijgen voor wat er gebeurt bij een ruimte tekort.

6.5 Flexbox items ('child elements')

Iedere flex container wordt pas krachtig zodra deze elementen bevat met een ingestelde 'flex' eigenschap. Net als de flex-containers zijn ook de bevattende flex elementen redelijk uitgebreid te configureren.

De volgende flex element eigenschappen zullen in de komende paragrafen aan bod komen: flex-basis, flex-shrink, flex-grow, flex, order en align-self.

6.5.1 Flex

Onderstaand tonen wij een korte beschrijving, gevolgd door een aantal voorbeelden waarin wij de flex-grow, flex-shrink en flex-basis tonen. Zodra deze drie eigenschappen voor u bekend zijn, gaan we kijken naar de korte notatie (flex).

6.5.2 Flex basis

Deze eigenschap heeft als waarde een getal om aan te geven hoe hoog een element is in geval de flex-direction is ingesteld op 'column' en geeft aan hoe breed een element is in

geval de flex-direction is ingesteld op 'row'. Ook kan deze eigenschap de waarden 'auto' en 'content' bevatten, ofwel: "kijk naar mijn ingestelde width en height". 'Content' stelt de breedte of hoogte in op de afmeting van de inhoud van het element.

6.5.3 flex-shrink

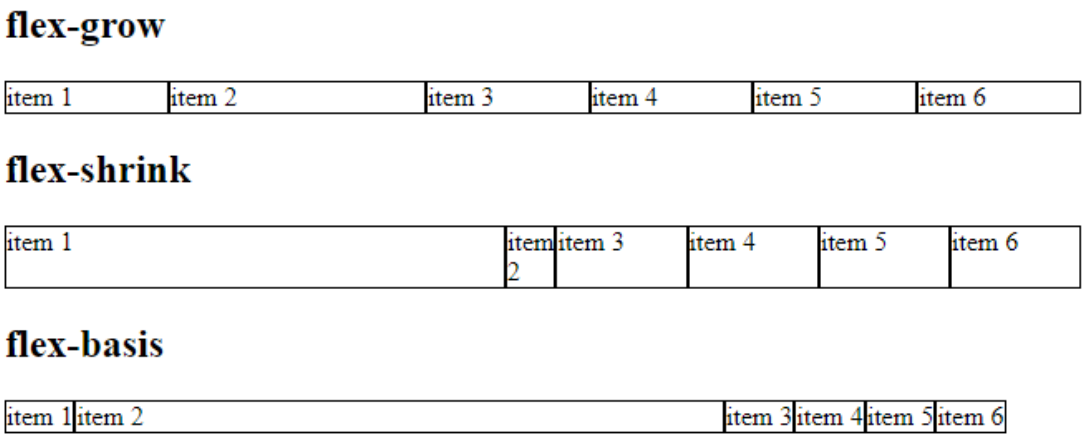
Hiermee stellen wij een deel of vermenigvuldig waarde van de verkleining van een de afmeting van een element in, in verhouding tot de andere elementen binnen de container.

6.5.4 flex-grow

Hiermee stellen wij een deel of vermenigvuldig waarde van de vergroting van een de afmeting van een element in, in verhouding tot de andere elementen binnen de container.

```
<html>
      <head>
      <style>
      #container1, #container2, #container3 {
              display:   flex;
            }

            #container1 div, #container2 div, #container3
div {
              border:    solid 1px black;
            }

            /* Flex-grow */
            #container1 .element2 {
              flex-grow: 2;
            }
            #container1 div {
              flex-grow: 1;
            }


            /* Flex-shrink */
            #container2 div {
              /* standaard te grote grootte van de div's */
              width: 300px;
            }

            #container2 .element1 {
              /* krimpt niet */
              flex-shrink:0;
            }
            #container2 div {
              flex-shrink: 1; /* krimpt wel */
            }
            #container2 .element2 {
              flex-shrink: 1.5; /* krimpt meer dan rest */
            }

            /* Flex-basis */
            #container3 .element2 {
```

```
              /* 60% ruimte al voordat overige regelruimte
evt wordt benut */
              flex-basis: 60%;
            }
      </style>
  </head>
  <body>
   <main>
                  <h2>flex-grow</h2>
                  <div id="container1">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

                  <h2>flex-shrink</h2>
                  <div id="container2">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>

                  <h2>flex-basis</h2>
                  <div id="container3">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                        <div class="element4">item 4</div>
                        <div class="element5">item 5</div>
                        <div class="element6">item 6</div>
                  </div>
            </main>
</body>
</html>
```



6.5.5 Flex eigenschap

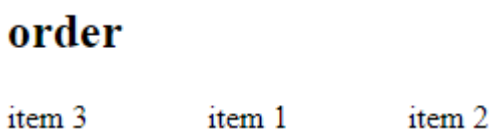
Deze eigenschap is enkel en alleen een samengevoegde notatie van de drie andere bovengenoemde flex eigenschappen: flex-basis, flex-grow en flex-shrink. Het wordt in de praktijk sterk aangeraden om deze notatie te gebruiken voor de overzichtelijkheid.

6.5.6 Order eigenschap

Deze eigenschap geeft aan in welke andere volgorde van elementen op te geven dan eigenlijk in de HTML code gedefinieerd is. 0 is het eerste element in de container. Zie onderstaande voorbeeld voor een verduidelijking:

```
<html>
      <head>
      <style>
      #container1 {
              display: flex;
            }

            div {
              width:   300px;
            }

            .element1 {
              order: 2
            }

            .element2 {
              order: 3
            }

            .element3 {
              order: 1
            }
      </style>
  </head>
```

```
<body>
   <main>
                  <h2>order</h2>
                  <div id="container1">
                        <div class="element1">item 1</div>
                        <div class="element2">item 2</div>
                        <div class="element3">item 3</div>
                  </div>
            </main>
</body>
</html>
```



6.5.7 Align-self eigenschap

Hiermee overrulen wij de 'align-items' eigenschap van een container, enkel voor dit specifieke element.

6.6 Extra aandachtspunten voor positionering

We hebben nu geleerd hoe we elementen kunnen positioneren op een pagina. Echter zijn er nog een aantal belangrijke zaken waarmee u rekening dient te houden als u een pagina ontwerpt. We zullen wel in deze paragraaf nog enkele praktische zaken opsommen waar u in de praktijk tegenaan kunt lopen.

6.6.1 Overflow

Indien een element een hoop meer tekst bevat dan er eigenlijk weergegeven kan worden in het element, zal de tekst over de randen van het element 'vloeien'. Deze overflow is te sturen middels CSS.

De overflow eigenschap kan de volgende waarden bevatten:

- visible - De overflow inhoud wordt niet afgeknipt. Dus alle content die buiten het element valt, wordt (e.v.t. over andere elementen) getoond. Dit is de default instelling.
- hidden - De overflow inhoud wordt afgeknipt. Buiten het element wordt niet meer getoond. Er kunnen dus stukken tekst wegvallen.
- scroll - De overflow is zichtbaar, wanneer men met de toegevoegde scrollbar scrolled.
- auto - Indien er overflow inhoud is, wordt deze middels het toevoegen van een scrollbar zichtbaar.

In veel gevallen zul je de overflow op 'auto' willen instellen, zodat er een scrollbalk verschijnt in het element om zo toch, en netjes, de gehele inhoud te kunnen lezen.

6.6.2 Breedte van een element

Wanneer we kijken naar de width eigenschap, moeten we onthouden dat deze eigenschap staat voor de breedte van het element inclusief padding en breedte van de border. Als we de breedte van het border van een element wijzigen, dan wordt de totale breedte van het element dus groter.

Stel we hebben een paragraaf met een breedte van 200 pixels met daarbij inbegrepen een border van 1 pixel breed. Vervolgens willen we de border 4 pixels breed maken. Hierdoor komt er zowel links als rechts 3 pixels bij. De totale breedte van de paragraaf wordt dan: 200+3+3 = 206 pixels.

Dit klinkt logisch en eenvoudig, echter verkijkt men zich hierop.

De totale breedte of hoogte van een element is inclusief borders en padding

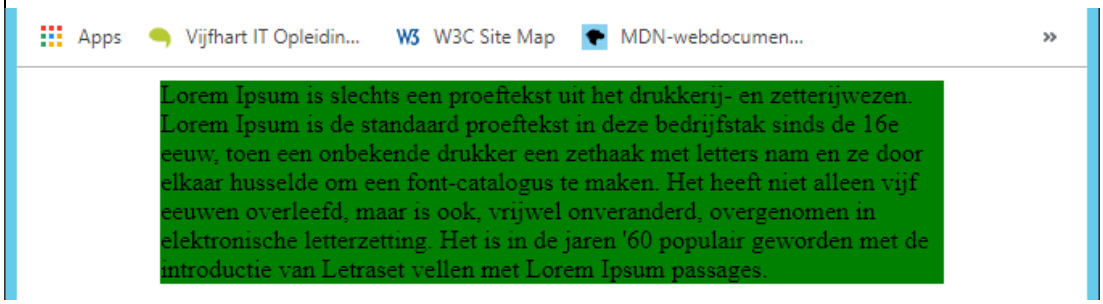
6.6.3 Horizontaal uitlijnen

Om een element zoals een div horizontaal in het midden van een pagina uit te lijnen, kunnen we gebruik maken van een ingestelde breedte op zowel zichzelf als op het parent element. Daarnaast dienen we een margin in te stellen op de (bijvoorbeeld) div, zodat deze daadwerkelijk gecentreerd wordt weergegeven. Én, indien je de breedte van in dit geval de div op een percentage instelt (dus ten opzichte van de parent), zal hij ook keurig meeschalen bij andere beeld formaten.

Bekijk het volgende voorbeeld en pas hem indien gewenst aan om te kijken wat het effect is.

```
<html>
      <head>
      <title>Horizontaal centreren</title>
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
      <style>
      #midden {
              width:    75%;
              margin:   0px auto;
              background-color: green;
            }
      </style>
  </head>
      <body>
            <div id="midden">
                  Lorem Ipsum is slechts een proeftekst uit
het drukkerij- en zetterijwezen.
                  Lorem Ipsum is de standaard proeftekst in
deze bedrijfstak sinds de 16e eeuw,
                  toen een onbekende drukker een zethaak
met letters nam en ze door elkaar husselde
                  om een font-catalogus te maken. Het heeft
niet alleen vijf eeuwen overleefd,
                  maar is ook, vrijwel onveranderd,
overgenomen in elektronische letterzetting.
```

```
                  Het is in de jaren '60 populair geworden
met de introductie van Letraset vellen
                  met Lorem Ipsum passages.
            </div>
      </body>
</html>
```

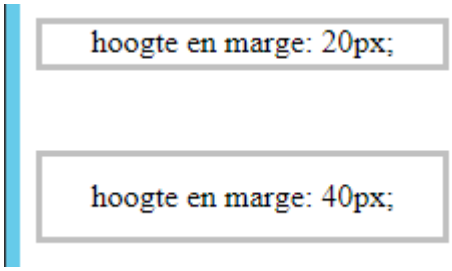


Dit is zéér handige code, die regelmatig wordt toegepast.

6.6.4 Verticale margins

Een belangrijk aspect van verticale vormgeving is het inklappen van verticaal aangrenzende margins. Het inklappen is alleen van toepassing op margins en niet op borders of padding. Dit inklappen of overlappen, gebeurt wanneer er twee elementen boven elkaar gepositioneerd zijn en er op de ene een bottom- en op de andere een top-margin is ingesteld. Van deze margins die naar elkaar toe lopen, wordt de hoogste waarde aangenomen als leidend.

```
<html>
   <head>
      <style>
      div {
        border-style:   solid;
        border-color:   silver;
        width:200px;
        text-align:center;
      }

      #div1 {
        margin-bottom:20px;
        height:20px;
      }

      #div2 {
        margin-top:40px;
        height:40px;
        line-height:40px;
      }
      </style>
  </head>
    <body>
      <div id="div1">hoogte en marge: 20px;</div>
      <div id="div2">hoogte en marge: 40px;</div>
    </body>
</html>
```



We zouden nu verwachten dat de afstand tussen beide divs gelijk is aan 40 + 20 = 60px. We zien echter dat de afstand tussen beide divs even groot is als de hoogte van de tweede div: 40px. De marges zijn ingeklapt, de grootste van beide marges (40px) blijft over tussen beide divs.

De afstand rond float elementen wordt anders berekend. Als een block element met clear onder een float element wordt geplaatst, geldt alleen de marge van het float element – ook als deze marge kleiner is dan die van het volgende block element. Als dat volgende block element ook een float element is, dan worden beide marges bij elkaar opgeteld..

Maak van het eerste div element een float element, en zet het volgende div element met clear daaronder. Bekijk het resultaat. De afstand wordt nu 20px.

Maak vervolgens ook het tweede div element een float element. Bekijk het resultaat: de afstand is nu 60 px geworden.

6.7 Grid

Het grid opmaak systeem in CSS is een vijfde opmaak mogelijkheid naast het gebruik van tabellen, positionering met de 'position' eigenschap, floats en het flexbox systeem. Dit grid systeem is echter mogelijk de meest interessante gezien de problemen waar wij met veel van de andere methoden tegenaan lopen.

Met name zaken als verticale uitlijning en het creëren van tabel-achtige layouts (met rijen en kolommen) is vaak omslachtig. Met behulp van het grid systeem gaan wij hieraan werken.

Net als bij het flexbox systeem zijn er eigenschappen voor een container element en eigenschappen voor de child elementen (de zogeheten grid items). Wij zullen nu eerst ingaan op de eigenschappen van de container elementen.

6.7.1 Grid container parent eigenschappen

Om te beginnen dienen wij de 'display' eigenschap, die al in een eerder hoofdstuk aan bod is gekomen, als waarde 'grid' mee te geven. Dit vertelt het systeem dat dit element de container wordt voor grid items. Ook zijn de opties 'subgrid' en 'inline-grid' mogelijke waarden, nuttig voor respectievelijk een grid binnen een grid en een inline niveau grid. Maar op deze twee waarden zullen wij in deze cursus niet verder op ingaan.

6.7.2 Grid template

De grid-template eigenschap is een combinatie (korte notatie) van de grid-template-rows, grid-template-columns en grid-template-areas eigenschappen.

Een grid is opgebouwd uit kolommen en rijen. Wij kunnen bij de definitie aangeven hoeveel kolommen en rijen wij willen gebruiken, ieder met de door ons in te vullen breedte (of hoogte, voor rijen). Daarnaast kunnen wij iedere kolom en rij een naam geven om later naar terug te kunnen verwijzen.

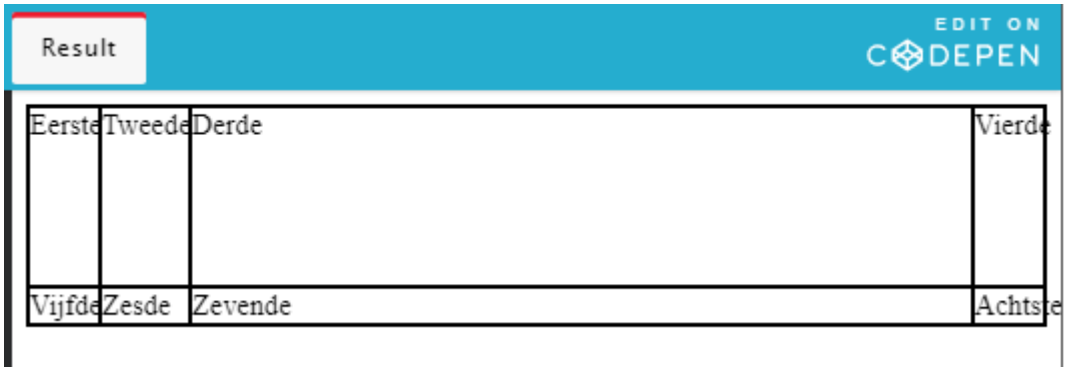
Het is mogelijk om **'fr'** als maatvorm in een grid te gebruiken. Deze afkorting staat voor een bepaalde fractie van de hoeveelheid vrije ruimte.

Wij zullen nu eerst op de -rows, -columns en -areas eigenschappen in gaan voordat wij deze combineren tot de 'grid-template' eigenschap.

6.7.3 grid-template-columns en grid-template-rows

Om het aantal kolommen en rijen in te stellen gebruiken wij de twee bovengenoemde eigenschappen. Onderstaand volgt een voorbeeld om het effect van deze twee eigenschappen te verduidelijken:

```
<html>
      <head>
      <style>
            #container{
              display: grid;
              grid-template-columns: 40px 50px auto 40px;
              grid-template-rows: 100px auto;
            }

            div {
              border: solid 1px black;
            }
      </style>
  </head>
  <body>
        <div id="container">
            <div>Eerste</div>
            <div>Tweede</div>
            <div>Derde</div>
            <div id="item">Vierde</div>
            <div>Vijfde</div>
            <div>Zesde</div>
            <div>Zevende</div>
            <div>Achtste</div>
        </div>
      </body>
</html>
```



Wij kunnen zien dat wij hebben aangegeven vier kolommen te willen gebruiken (40px, 50px, auto en 40px). Daarnaast hebben wij aangegeven twee rijen te willen gebruiken (100px en auto). Het aantal waarden dat wij hier dus benoemen is gelijk aan het aantal kolommen of rijen.

Het is wellicht ook opgevallen dat waar wij de waarde 'auto' hebben ingevuld, de ruimte wordt opgevuld tot de volledige breedte. Bij de rijen wordt dan de rij zo klein mogelijk gehouden.

6.7.4 Grid-gap

Deze eigenschap stelt witruimte in tussen de rijen en/of kolommen van een grid. Het is een soortgelijk effect dat bij tabellen wordt gerealiseerd door de cellspacing eigenschap. Grid-gap is een verkorte notatie van de grid-row-gap en grid-column-gap eigenschap.

Een oefening met deze eigenschap gaan wij zo dadelijk zien.

6.7.5 Overige container eigenschappen

De uitlijning van de grid items in de grid langs de x-as en y-as doen wij met de justify-items en align-items eigenschap. Dit gaat dus niet over de uitlijning van tekst binnen de grid items.

Justify-content en align-content gebruiken we om de grid elementen in te delen in de container indien de elementen met een
vaste (niet-flexibele) waarde zijn ingesteld, waardoor er ruimte ontstaat tussen de items binnen de container.

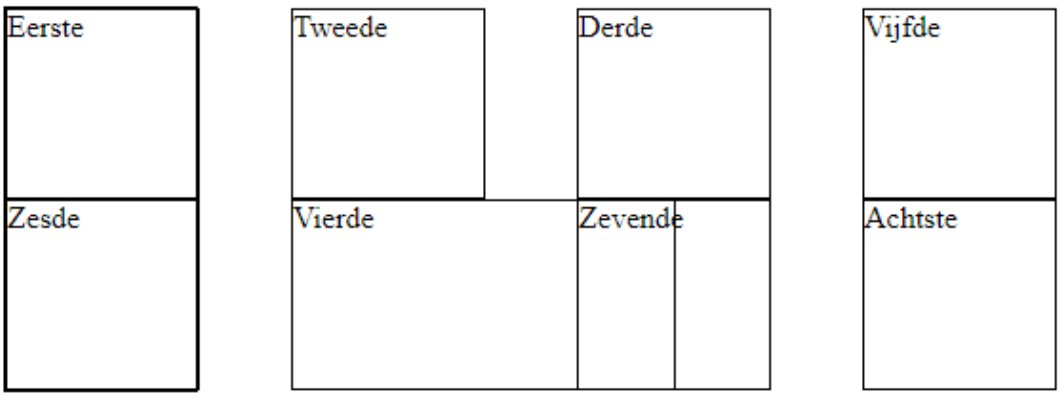
Meer over deze vier eigenschappen leest u op bijvoorbeeld de volgende webpagina: https://css-tricks.com/snippets/css/complete-guide-grid/.

6.7.6 grid-column-start, grid-column-end, grid-row-start en grid-row-end

Dit zijn de belangrijkste eigenschappen van grid items. Hiermee geven wij aan op welke rij en op welke kolom dit huidige grid item geplaatst dient te worden.

Dit gebeurt op basis van een op te geven start en eind positie. Deze posities mogen (oplopende) nummers zijn (bijvoorbeeld respectievelijk '1' voor het begin en '3' als einde), maar mogen ook een span waarde zijn. Dit betekent een overspanning van verschillende kolommen of rijen. Zie onderstaand voorbeeld voor de overeenkomst / het verschil:

```
<html>
      <head>
      <style>
            #container{
              display: grid;
              grid-template-columns: 150px 150px 150px
150px;
              grid-template-rows: 100px 100px;
            }

            div {
              border: solid 1px black;
              width: 100px;
            }

            #item {
              grid-column-start: 2;
              grid-column-end: 2;
              grid-row-start: 2;
              grid-row-end: 2;

              width: 200px;
            }
      </style>
  </head>
  <body>
        <div id="container">
            <div>Eerste</div>
            <div>Tweede</div>
            <div>Derde</div>
            <div id="item">Vierde</div>
            <div>Vijfde</div>
            <div>Zesde</div>
            <div>Zevende</div>
            <div>Achtste</div>
        </div>
      </body>
</html>
```



We zien hier de grid items keurig verdeeld. Ieder heeft de positie zoals de div's zijn gedeclareerd. Echter heeft de div met als id 'item' een andere positie. Probeer te beredeneren waarom deze div op deze positie staat.

Wijzig nu de regel:
`grid-column-end: 2;`
naar:
`grid-column-end: span 2;`

We zien doordat wij het element met als id 'item' een overspanning (span) van waarde '2' hebben gegeven, de andere elementen dus een kolom moeten opschuiven. Dit is ook de reden dat er een nieuwe rij getoond wordt, ondanks dat wij in onze container definitie hebben opgegeven maar twee rijen te willen zien.

Het is wellicht ook opgevallen dat het kader van de div niet over de twee kolommen heen gaat. Dat heeft ermee te maken dat de div zelf niet de twee kolom posities in beslag neemt, maar de grid item positie twee plekken in beslag neemt. De div staat daar weer in. Maak het #item element maar eens 200px breed, dan zien we dit effect.

De witruimte tussen de elementen heeft te maken met de ruimte die de grid items hebben (150px breedte per stuk) en de ruimte die wij aan onze div's hebben toegekend (100px).

Wijzig de breedte van onze div's van 100px naar 150px en bekijk het resultaat.

Zodra dit is gelukt, voeg dan de volgende eigenschap toe aan de CSS code voor de container (#container {}):
grid-gap: 20px 0px;
Probeer voor jezelf te verklaren wat dit doet!

Toelichting nadien:
De eigenschap grid-gap is een samenvoeging van de container eigenschappen grid-column-gap en grid-row-gap. Dit is een soortgelijke instelling als wij kennen van de cellspacing bij tabellen en zorgt dus voor witruimte tussen de rijen en/of kolommen van een grid.

6.7.7 Overige grid item eigenschappen

De uitlijning van de content in de grid items langs de x-as en y-as doen wij met de justify-self en align-self eigenschap. Deze eigenschappen lijken qua syntax op de justify-content en align-content eigenschappen, echter gelden deze niet voor de plaatsing van de grid items binnen de container, maar voor de content binnen de grid-items, ieder voor zich.

6.7.8 Flexbox, grid en formulieren

Het opmaken van velden in formulieren is naast de manieren die eerder zijn genoemd, ook goed mogelijk middels het gebruik van flexbox en/of grid.

Omdat dit wat buiten de scope van deze cursus valt, kunt u er eventueel hier meer over lezen:

https://webdesign.tutsplus.com/tutorials/how-to-build-web-form-layouts-with-css-grid--cms-28776

en via deze link:

https://www.sitepoint.com/make-forms-fun-with-flexbox/

6.8 Samenvatting

Met positionering kunnen we een pagina indelen. Dit kan op vijf verschillende, eventueel te combineren, manieren.

Bij positionering met behulp van de 'position' eigenschap wordt altijd gekeken naar de element positie ten opzichte van de canvas of een parent element. We dienen hiervoor dus de 'position' eigenschap in te stellen. We kunnen de top en left eigenschappen instellen om dit te coördineren. Ook kunnen wij gebruik maken van de float eigenschap op elementen op een ander niveau te positioneren. Vervolgens is het flexbox model een redelijk recent toegevoegde functionaliteit om elementen ook mee te kunnen positioneren op een pagina. Dit model biedt wat meer flexibiliteit, maar is een stuk complexer dan de andere mogelijkheden tot zover. Tot slot zijn met de komst van het grid-layout model de deuren geopend voor overzichtelijke en geavanceerde designs, zonder in te hoeven leveren op de (relatieve) eenvoud.

7 Responsive design

7.1 Inleiding

- Het mobile first concept kunnen toelichten
- De eigenschappen van, en het verband kunnen benoemen tussen de viewport, breakpoints en de pixel density
- Het kunnen aanpassen van een webpagina zodat deze om kan gaan met touch events
- Het kunnen maken van media queries in CSS en in HTML
- Het kunnen benoemen van responsive table design patterns
- Het kunnen benoemen van het nut van responsive fonts
- De verschillen kunnen benoemen tussen de verschillende responsive image mogelijkheden
- Het kunnen maken van een picture element met alternatieve bronnen en afmetingen

Bij het ontwerpen van web omgevingen is het inmiddels een goed gebruik om eerst voor mobiele devices te ontwikkelen en vervolgens voor een desktop omgeving. Dat biedt een heel aantal voordelen:

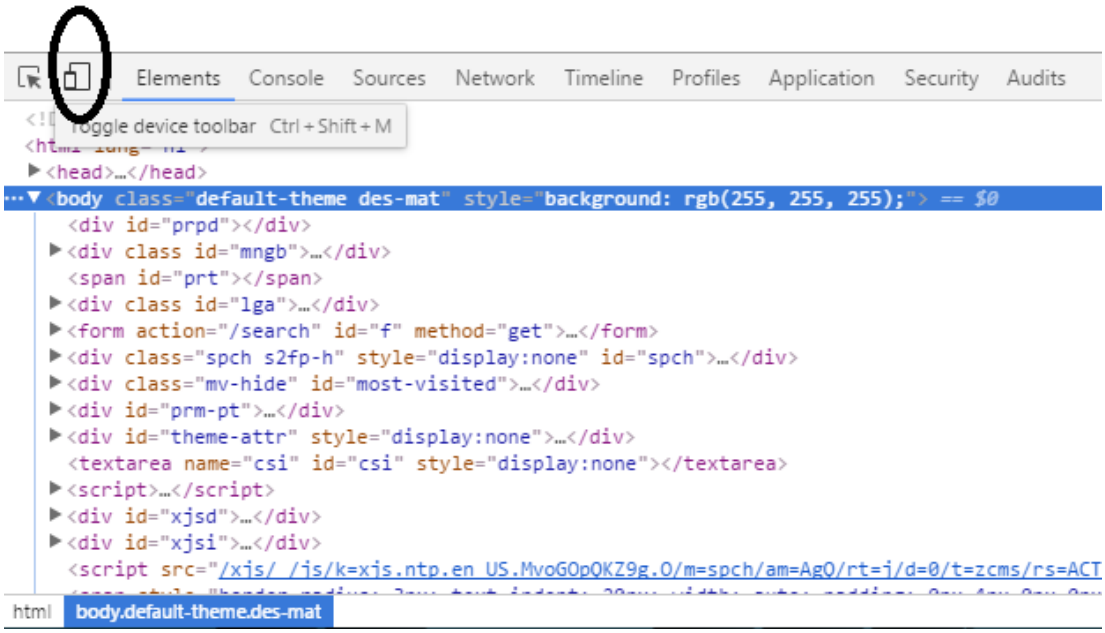
- Doordat we de beschikking hebben over relatief weinig ruimte op een mobiel device (zelfs tot minimaal 200 pixels breed), worden we gedwongen de meest belangrijke content de filteren van minder belangrijke content. Dit op zich heeft al weer tal van voordelen namelijk het beter semantisch kunnen organiseren van een pagina alsook het kunnen maken van efficiënte formulieren.
- Omdat een webpagina inclusief bijvoorbeeld afbeeldingen en video's goed moet performen op een mobiel device, ontwikkelen we daar ook naar. Hij zal dan automatisch ook goed performen op een desktop omgeving.

We gaan in dit hoofdstuk dieper in op hoe wij een website zo responsive als mogelijk kunnen maken. Responsive betekent dat de webpagina er zo goed mogelijk uit ziet op verschillende apparaat formaten, zowel in landscape (horizontaal) en in portrait (verticaal vastgehouden) modus.

We zullen gaan kijken naar viewports, pixel density, touch events, break points, media queries, responsive tables, responsive fonts en responsive images.

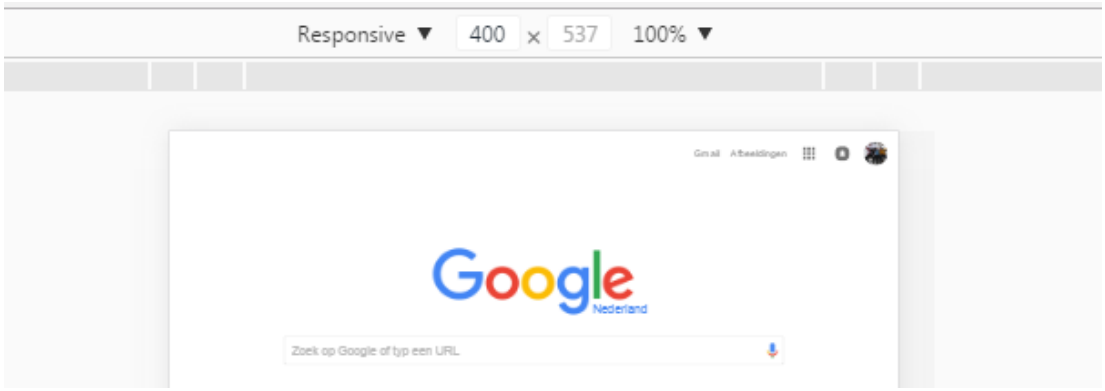
Klik vervolgens op het icoon met vierkantjes om de device modus in te schakelen ("Toggle device toolbar", ofwel <ctr><shift><m>):

7 Responsive design



We krijgen nu een device weergave te zien van de huidige webpagina. Hiermee kunnen wij dus simuleren dat wij op een device met een zogeheten andere viewport werken dan waar wij nu op werken. Bovenin beeld kunnen wij het gewenste device kiezen, óf zelf een device toevoegen met een gewenste viewport. Ook kunnen we het zoom-level instellen; hoe ver dient de pagina ingezoomd te zijn?

Tot slot zien ook dat onze muis aanwijzer binnen de webpagina is veranderd naar een ronde cirkel. Dat stelt onze vinger voor wanneer wij over een site heen bewegen op een mobiel device.



7.2 Viewport, pixel density en relatieve afmetingen

7.2.1 Viewport

De viewport is het voor gebruikers zichtbare deel van een webpagina. Elementen die bijvoorbeeld breder zijn dan de viewport, zijn alleen volledig te zien wanneer de gebruiker (horizontaal) gaat scrollen. Niet gebruiksvriendelijk dus!

7 Responsive design

Het instellen van de viewport voor op je webpagina kan middels een <meta> tag. En gaat als volgt:

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

Dit geeft aan dat de aan te houden viewport, de gehele breedte van het device scherm is. Daarnaast staat de initial-scale voor hoe ver de pagina bij openen ingezoomd dient te zijn. Standaard is dat dus 1.0 (geen zoom).

Het is zéér aan te bevelen te allen tijde de viewport aan te geven van je device in webpagina middels de bovengenoemde meta tag. Is de viewport niet middels een meta tag ingesteld? Dan gebruikt de browser een standaard instelling van 980 device independent pixels (DIPS). Is je device eigenlijk maar 300 DIPS breed, dan gaat hij de pagina dusdanig verkleinen (meer dan 3x!) om aan de 300 DIPS te voldoen. De pagina kan hierdoor bijna onleesbaar worden. Daarnaast zal de browser proberen (in zijn ogen) belangrijke content groter te maken.

7.2.2 Device independent pixels

Wanneer we een element een breedte willen geven van 200 pixels op het scherm, hoe breed is dat dan daadwerkelijk? Dat heeft alles te maken met device independent pixels (DIPS. Een device heeft altijd een fysiek aantal pixels (hardware pixels genoemd). Om content mooi op verschillende devices van verschillende formaten te tonen, bekijkt de browser pixels als DIPS. Dit kan een veelvoud of deling zijn van de hardware pixels. Hierdoor nemen de elementen dezelfde relatieve ruimte in op ieder willekeurig device. Hoe berekenen we nu het daadwerkelijk aantal DIPS dat we dienen te gebruiken op onze website? Deel het aantal fysieke schermpixels door de device pixel ratio om het aantal DIPS te achterhalen. De device pixel ratio kan op het device uitgelezen worden (zie verderop dit hoofdstuk).

7.2.3 Relatieve afmetingen

Probeer waar mogelijk je elementen altijd relatieve maten op te geven. Hierdoor kunnen de elementen mee schalen met de viewport. Kies voor een indicator op basis van een percentage (%) of view-width (vw). Voor teksten kan ook em (element fontsize) gebruikt worden. Waarbij bijvoorbeeld 3em staat voor 3 maal de grootte van het huidige font.

7.3 Touch events

Het is al een lange tijd niet meer het geval dat door webpagina's enkel met een toetsenbord en een muis genavigeerd wordt. Vooral touch events, ofwel aanraak momenten, nemen een niet te onderschatten grote rol in bij het navigeren van, naar en binnen webpagina's.

Het belangrijkste waar wij rekening mee moeten houden is dat een klik met de muis niet hetzelfde is als een druk met de vinger. Vooral gezien een vinger qua doorsnede veel meer ruimte in beslag neemt dan een cursor op het scherm, zullen wij daar bij het maken van klikbare elementen op onze pagina rekening mee willen houden.

Richtlijn: houd elementen die wij met onze vingen moeten kunnen aanraken voor een effect minimaal 40 x 40 pixels breed. Dat is gemiddeld de lengte en breedte van een vinger. Voor de zekerheid kunnen we nog een marge inbouwen van 5 pixels. Dus houdt voor de zekerheid 45 x 45 pixels aan.

7 Responsive design

Maak een verticaal of horizontaal (display: inline) menu van een ongeordende lijst (ul met li elementen). Zorg ervoor dat ieder menu element minstens de juiste afmetingen heeft om erop te kunnen ‘klikken’ met onze vinger.

7.4 Break points

Om een pagina op zo veel mogelijk devices goed weer te geven, kunnen we gebruik maken van break points en media queries. Zoals je wellicht hebt gemerkt bij het vorige voorbeeld of bij andere webpagina’s die je bent doorlopen, wordt de opmaak soms aangepast aan de hand van het scherm formaat. Ieder omslagpunt is een break point.

Bekijk deze pagina: http://art.yale.edu/. Deze is nog niet aangepast voor mobiele devices. Verklein die pagina maar eens en kijk of alle belangrijke content nog zichtbaar is en of de site er nog overzichtelijk en gebruiksvriendelijk uit ziet.

Zoals je in het vorige voorbeeld hebt kunnen zien, is deze webpagina niet responsive. Als je het venster hebt verkleind, zul je zien dat alles blijft zoals het is én er dus essentiële informatie wegvalt. Er dient dus bij het ontwikkelen van een webpagina véél rekening te worden gehouden met mobiele devices.

Bij het responsive maken van een webpagina kiezen we voor een aantal (meestal 2 of 3) breakpoints waarop de layout van de pagina aangepast wordt aan de device width (breedte is het meest van belang). Dit gaan we controleren en toepassen met media queries.

Het verschilt per pagina op welke punten wij breakpoints kunnen instellen. Wél is het handig om in gedachten te houden welke formaten devices er zoal zijn.

Een veel toegepaste verdeling is:

- Max-width van 480px voor mobiele telefoons
- Gemiddeld formaat tablets tot 700px
- Grotere devices tot 1024px
- Alles boven de 1024px

Maar houd in gedachten: meer is niet altijd beter. Als een pagina té veel veranderd qua formaat, kan dat ook erg vervelend overkomen (wanneer je bijvoorbeeld regelmatig je browservenster formaat wijzigt op je desktop PC).

7.5 Media Queries

Media queries zijn commando’s waarmee wij op basis van de huidige viewport afmeting één of meerdere CSS stijlen kunnen toepassen. Vaak koppelen wij media queries aan break points om onze gewenste wijziging van de lay-out te relateren aan de huidige viewport (scherm afmeting).

We kunnen een media query op drie plekken in onze code toepassen:

- In een @import commando, om een aparte CSS file aan te geven voor een specifieke viewport
- In een <link> commando, om een aparte CSS file aan te geven voor een specifieke viewport
- In de CSS code, om één of meerdere CSS stijlen aan te geven voor een specifieke viewport

7 Responsive design

Hierbij worden media queries in de CSS code in de praktijk het meest gebruikt. In de opdrachten van dit hoofdstuk zullen wij voornamelijk werken met media queries in het <link> element, ten behoeve van overzichtelijkheid.

Hoe ziet een media query eruit?

```
<html>
      <head>
      <style>
          div {color : yellow;}
            @media screen and (min-width: 800px)
            /* browser breedte van minimaal 800px */
             {div {color : blue;}
             }
            @media screen and (min-width: 400px) and (max-
width: 600px)
             {div {color : red;}
             }
            @media screen and (max-width: 400px), (min-
width: 1000px)
            /* browser breedte van maximaal 400px OF
minimaal 1000px */
            { div {color : green;}
            }
      </style>
  </head>
  <body>
        <div id="container">
            <div>Eerste</div>
            <div>Tweede</div>
            <div>Derde</div>
            <div id="item">Vierde</div>
            <div>Vijfde</div>
            <div>Zesde</div>
            <div>Zevende</div>
            <div>Achtste</div>
        </div>
      </body>
</html>
```

In het code voorbeeld zagen we dus dat we meerdere voorwaarden kunnen combineren. Het keyword ‘AND’ gebruiken we voor als de voorwaarde óók ergens anders aan moet voldoen. We gebruiken een komma ‘,’ wanneer het een alternatief is (optie 1 óf optie 2, beiden zijn goed).

Er is verschil in laadtijd tussen deze manieren. De @import doet er het langs over. Deze is ook minder flexibel. Als je deze manier toepast hoort hij vóór alle overige CSS statements te staan in een style element.. Gebeurt dit niet, dan zal de @import simpelweg niet uitgevoerd worden.

7 Responsive design

7.6 Responsive tables

Bij tabellen is het van extra belang dat er gekeken wordt naar wat de meest nuttige informatie is om te tonen aan de eindgebruikers. De informatie binnen een tabel willen we vaak overzichtelijk weergeven. Om dat te kunnen doen zullen we moeten streven naar (in volgorde van belang):
Duidelijke kolom koppen
Niet teveel kolommen
Niet teveel rijen
Het bedenken van goede kolomkoppen is natuurlijk belangrijk. Zorg er met CSS code voor dat deze ook de juiste gewenste opmaak krijgen.

Indien we teveel kolommen gebruiken, zullen sommigen zelfs buiten de viewport vallen en dus uit het zicht komen. Hiervoor zijn twee design patterns toe te passen, ieder met een ander resultaat. Deze zullen wij zo direct gaan behandelen.

Het aantal rijen beperken is voornamelijk te doen met JavaScript code of een server programmeertaal. We kunnen bijvoorbeeld het resultaat pagineren. Zodat er op een enkele pagina maar maximaal 25 rijen getoond worden, in plaats van alle 150 tegerlijk.

We gaan nu nog even dieper in op het gebruik van design patterns voor het weergeven van veel kolommen.

7.6.1 Contain table pattern

We kunnen ervoor zorgen dat alle kolommen zichtbaar blijven, door de gebruiker de mogelijkheid te geven (horizontaal) te scrollen binnen een tabel. Bekijk onderstaande voorbeeld op een desktop / tablet formaat en verklein vervolgens het venster om te zien dat er een horizontale scrollbalk verschijnt.

```
<!DOCTYPE html>
<html>
<head>
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Contain table responsive table design</title>
      <style>
      /* Géén media query, resultaat middels overflow is
scrollbalk bij te klein venster*/
            #tabel-wrapper {
              overflow-x: auto;
              width:      100%;
            }
      </style>
</head>
<body>
    <div id="tabel-wrapper">
        <table>
            <!-- Probleem is normaliter bij het verkleinen
van het venster, dat de tabel steeds minder zichtbaar wordt
-->
```

```
            <thead>
                <th>Student</th>
                <th>Januari</th>
                <th>Februari</th>
                <th>Maart</th>
                <th>April</th>
                <th>Mei</th>
                <th>Juni</th>
                <th>Juli</th>
                <th>Augustus</th>
                <th>September</th>
                <th>Oktober</th>
                <th>November</th>
                <th>December</th>
                <th>Eindcijfer</th>
            </thead>
            <tbody>
                <tr>
                    <td>Qwin</td>
                    <td>6</td>
                    <td>8</td>
                    <td>7</td>
                    <td>7</td>
                    <td>9</td>
                    <td>6</td>
                    <td>9</td>
                    <td>9</td>
                    <td>8</td>
                    <td>10</td>
                    <td>7</td>
                    <td>8</td>
                    <td>10</td>
                </tr>
                <tr>
                    <td>Joah</td>
                    <td>6</td>
                    <td>7</td>
                    <td>8</td>
                    <td>5</td>
                    <td>6</td>
                    <td>6</td>
                    <td>7</td>
                    <td>8</td>
                    <td>7</td>
                    <td>6</td>
                    <td>7</td>
                    <td>8</td>
                    <td>8</td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
```

```
</html>
```

Student	Januari	Februari	Maart	April	Mei	Juni	Juli	Augustus	Septen
Qwin	6	8	7	7	9	6	9	9	8
Joah	6	7	8	5	6	6	7	8	7

Zoals we zien is de bovenstaande oplossing niet de mooiste oplossing, maar ook niet een hele complexe.

7.6.2 No table pattern

Indien we merken dat we teveel kolommen gebruiken, kunnen wij middels een truc min of meer de kolommen omwisselen met de rijen. Bekijk het onderstaande voorbeeld op een desktop / tablet formaat en verklein vervolgens het venster om te zien dat de kolommen, rijen worden.

```
<!DOCTYPE html>
<html>
<head>
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Contain table responsive table design</title>
      <style>
      /* Géén media query, resultaat middels overflow is
scrollbalk bij te klein venster*/
             /* Stap 1 (test het effect van alleen deze
stijl */
            table, thead, tbody, th, tr, td {
                display:    block;
            }

            /*  Stap 2 headers uit zicht halen. Positie
buiten beeld was ook voldoende geweest. */
            thead {
               display: none;
            }

            /* Stap 3 kolommen */
            td {
                position: relative;
                padding-left: 50%;          /* Zodat de
onderstaande content vóór de td kan komen te staan */
            }

            /* Stap 4 content uit 'data-cel' attribuut nu
als content vóór de td elementen plaatsen. */
            td:before {
                position:   absolute;
                left:       4px;
                content:    attr(data-cel);
            }
        }
```

```
      </style>
</head>
<body>
    <div id="tabel-wrapper">
        <table>
            <!-- Probleem is normaliter bij het verkleinen
van het venster, dat de tabel steeds minder zichtbaar wordt
-->

             <thead>
                <th>Student</th>
                <th>Januari</th>
                <th>Februari</th>
                <th>Maart</th>
                <th>April</th>
                <th>Mei</th>
                <th>Juni</th>
                <th>Juli</th>
                <th>Augustus</th>
                <th>September</th>
                <th>Oktober</th>
                <th>November</th>
                <th>December</th>
                <th>Eindcijfer</th>
            </thead>
            <tbody>
                <tr>
                    <td data-cel="Student">Qwin</td>
                    <td data-cel="Januari">6</td>
                    <td data-cel="Februari">8</td>
                    <td data-cel="Maart">7</td>
                    <td data-cel="April">7</td>
                    <td data-cel="Mei">9</td>
                    <td data-cel="Juni">6</td>
                    <td data-cel="Juli">9</td>
                    <td data-cel="Augustus">9</td>
                    <td data-cel="September">8</td>
                    <td data-cel="Oktober">10</td>
                    <td data-cel="November">7</td>
                    <td data-cel="December">8</td>
                    <td data-cel="Eindcijfer">10</td>
                </tr>
                <tr>
                    <td data-cel="Student">Joah</td>
                    <td data-cel="Januari">6</td>
                    <td data-cel="Februari">7</td>
                    <td data-cel="Maart">8</td>
                    <td data-cel="April">5</td>
                    <td data-cel="Mei">6</td>
                    <td data-cel="Juni">6</td>
                    <td data-cel="Juli">7</td>
                    <td data-cel="Augustus">8</td>
                    <td data-cel="September">7</td>
```

```
                    <td data-cel="Oktober">6</td>
                    <td data-cel="November">7</td>
                    <td data-cel="December">8</td>
                    <td data-cel="Eindcijfer">8</td>
                </tr>
            </tbody>

        </table>
    </div>
</body>
</html>
```

Student	Qwin
Januari	6
Februari	8
Maart	7
April	7
Mei	9
Juni	6
Juli	9
Augustus	9

We zien dat dit een beter esthetisch resultaat geeft dan het contain table pattern, máár, deze oplossing meer handwerk vereist omdat we:

- Ieder td element een data-cel attribuut meegeven en invullen. Die noemen we dus gelijk aan de (normaliter) kolomkop van die kolom.
- Middels CSS code halen wij de thead kolom uit het zicht middels: display: none;
- Middels CSS code tonen wij voor iedere regel (td element, maar als block weergegeven) de naam van de kolomkop middels: content: attr(data-cel);

7.7 Responsive fonts

Probleem: Te korte regels, te lange regels, op de verkeerde plaats afgekapte regels of wanneer het afkappen van regels zorgt soms voor onleesbare stukken tekst. Hierdoor moet men soms zelfs stukken overnieuw lezen om het te kunnen volgen.

Het door onderzoeken geadviseerde aantal karakters per regel zit tussen de 45 en 90 karakters per regel. Allen afhankelijk van font-size, schreef, enzovoorts. Een gemiddelde van 60 is goed om aan te houden. Onderzoeken o.a. uit het boek: “Typographie: A Manual of Design” door Emil. Ruder en de webpagina Web typography: http://webtypography.net/2.1.2.

Een goede minimum font-size om aan te houden is 16pt, met regel hoogte van 1.2em.
Kleine (minor) breakpoints (wijzigingen) op zaken als fonts en images.
Hoofdstuk 7 - Responsive design (4)

7.8 Responsive images

Wanneer we ervoor kiezen een afbeelding te tonen op onze webpagina, zullen we rekening moeten houden met de performance van onze site bij het laden ervan.

7 Responsive design

Twee belangrijke dingen zullen we ten aller tijde in ons achterhoofd moeten houden:

- Het laden van een externe file (afbeelding in dit geval) kost altijd een HTTP
- Des te lager de filesize, des te kleiner het resultaat is van het verzoek aan de server.

We kunnen een hoop instellen aan onze afbeelding en de manier van het laden ervan.

Indien je (eventueel basaal opgemaakte) tekst wilt weergeven, gebruik dan gewone tekst die met CSS code is opgemaakt. Ga in dit geval nooit een afbeelding maken van je stuk tekst, want:

- De tekst zal niet goed schaalbaar zijn
- De tekst is slecht vindbaar in de meeste zoekmachines
- Screenreaders kunnen de afbeelding niet als tekst gebruiken

Indien we een eenvoudig pictogram of icoon willen weergeven, kunnen we in veel gevallen ook gebruik maken van een font in plaats van een afbeelding. Hoe gaat dat in zijn werk? Eerst kiezen we een lettertype uit via (bijvoorbeeld) We Love Icon Fonts: http://weloveiconfonts.com/.

- Kies een lettertype uit waar het gewenste icoon in zit.
- Klik op '(hartje) Add' en vervolgens onderin beeld op: 'Use X', waarij 'X' staat voor het aantal lettertypen dat wij hebben geselecteerd.
- Er verschijnt nu een blok met CSS code. De import regel dienen we in onze CSS file te plaatsen om de lettertype(n) te kunnen gebruiken. Het toepassen van deze lettertypen gaat echter iets anders dan dat we gewend zijn.
- Geef de elementen waarbinnen we (direct) de iconen willen gebruiken een class-attribuut met als waarde: 'naam_lettertype-gekozen_icoon'. Hiernaar kunnen wij vervolgens vanuit onze CSS code verwijzen met de code:

```
[class*="naam_lettertype-"]:before {
font-family: 'naam_lettertype, sans-serif;
}
```

Bekijk ook de voorbeeldcode op de We Love Icon Fonts webpagina.

7.8.1 Filetype keuze

Om het juiste bestandstype te kunnen kiezen, moeten we weten welke mogelijkheden ons bestand dient te hebben. Hierbij dienen we (onder andere) te letten op de volgende zes aspecten:

- Compressie
- Zwart wit of kleur
- 8of 16 bits kleur
- CMYK of LAB mogelijkheid
- Transparantie
- Animatie mogelijkheid

Onderstaande link toont een tabel met daarin een aantal bestandstypen met de verschillende eigenschappen:
http://socialcompare.com/en/comparison/image-file-formats

Nog een aantal algemene tips:

- Verkies JPG boven andere formaten indien je iets complexere afbeeldingen zoals foto's wilt weergeven, maar niet te grote files wilt.

- Verkies altijd PNG boven GIF gezien de betere compressie en betere kleurweergave.

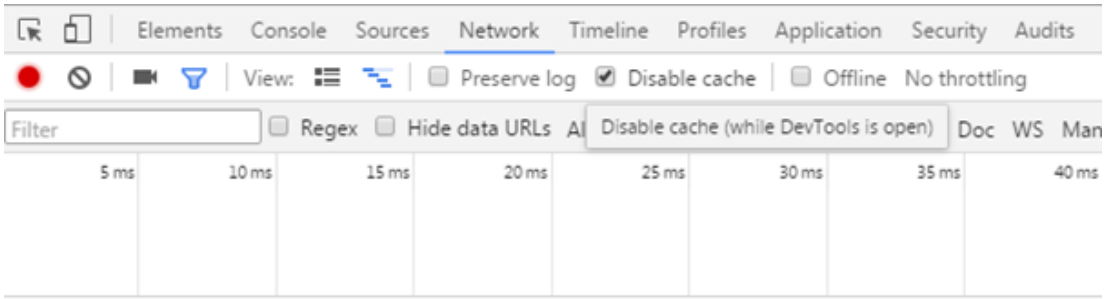
Verkies SVG indien je goed schaalbare (liefst eenvoudige) afbeeldingen wilt gebruiken

7.8.2 Filesize mogelijkheden

Voor het creeren en aanpassen van files met de bijbehorende formaten kunnen we bijvoorbeeld ImageMagick: https://www.imagemagick.org/ goed gebruiken. Om vervolgens de afbeeldingen te optimaliseren kunnen we de tool ImageOptim: https://imageoptim.com/ goed te gebruiken. Imageoptim is een betaalde service, maar biedt ook de mogelijkheid de API een periode gratis te gebruiken.

Hoe kunnen we zien hoe groot onze gebruikte afbeeldingen zijn én waar kunnen we dus straks ook het verschil zien wanneer we andere file typen gebruikt hebben of afbeeldingen vergroot of verkleind hebben?

Ga naar de Chrome Developer Tools en open daar het tabblad ‘Network’. Druk hier op de rode cirkel (record) en ververs de pagina (<ctrl><F5>). We zien nu voor ieder gedownload bestand de omvang en de tijd die het downloaden ervan in beslag heeft genomen. Files kunnen ook gecached worden op het systeem van de eindgebruiker. Dit zorgt voor een performance voordeel, maar kan ook nadelig zijn voor een actuele ervaring indien de afbeelding verouderd is. Indien we de download tijden willen berekenen, is het aan te raden het cachen van files uit te zetten:



7.8.3 Width en height instellingen

We hebben wellicht in de HTML cursus gezien dat het interessant kan zijn om alvast een hoogte en breedte in te stellen van onze afbeelding binnen HTML code. Dat zorgt ervoor dat de browser al rekening kan houden met een bepaald formaat afbeelding. Dit hoeft de CSS parser (verwerker) dan niet voor ons te doen. Welke maatvorm gebruiken we bij het instellen van de afmeting (in HTML of CSS)?

- Image width en height als pixels instellen kan, maar bij venster vergroten en verkleinen moet de user scrollen om de hele image te zien (zonder scroll balken!).
- Image width en height als % instellen kan, maar dan bij aanpassen image zelf in browser krijg je resolutie problemen.
- Image max-width op 100% kan, maar dan kan hij nooit groter worden dan zijn maximum.

7.8.4 Het srcset attribuut

Om een afbeelding zowel op een groot scherm als klein scherm scherp weer te kunnen geven, kunnen we een kleine image vergroten, waardoor hij op de hogere resolutie

waarschijnlijk onscherp wordt. Of we kunnen een grote image laten verkleinen voor de mobiele devices, maar wat ten kostte gaat van downloadtijd voor zo'n grote afbeelding.

Met het srcset attribuut, kunnen wij verschillende afbeelding-URL's opgeven voor één en dezelfde afbeelding. Hierdoor wordt door de browser de juiste afbeelding gekozen (maar voor nu nog allemaal gedownload!) aan de hand van de viewport van het device.

(U kunt dit voorbeeld niet zelf uitvoeren) :

```
<img src="skyline_1x.jpg" srcset="skyline_1x.jpg 1x,
skyline_2x.jpg 2x" alt="Skyline Rotterdam" />
```

We zien dat er gebruik wordt gemaakt van meerdere URL's binnen het srcset attribuut. Daarnaast staat er achter iedere URL een 1x, 2x of 3x waarde. Dit staat voor de pixel dichtheid, ofwel de pixels per inch. De meeste, grotere desktop schermen hebben een 1x dichtheid. Terwijl mobiele devices regelmatig 2x of zelfs 3x dichtheid hebben. Dit zijn natuurlijk lastige waarden om in te schatten, vooral gezien de techniek ontzettend snel gaat. Gelukkig kunnen we met een commando in onze Dev Tools zien welke pixel density ons device heeft.

. Ga binnen de Chrome Developer Tools naar het tabblad 'Console' en voer daar in: window.devicePixelRatio

En voilá! Daar is de ratio van het huidige device. Wat gebeurt er nu met ons huidige src attribuut? Waarom staat deze er nog? Dat is nu de default image URL geworden, in het geval srcset niet wordt ondersteund.

Bovenstaande voorbeelden zorgen ervoor dat de browser de juiste afbeelding toont. Echter wordt nog steeds iedere versie van de afbeelding gedownload. Hoe komt dat? De browser weet niet hoe groot iedere afbeelding is. Dat weet hij pas zodra ze zijn gedownload. Hoe kunnen we dit alsnog duidelijk maken aan de browser vóórdat hij alle images download, zodat hij énkel de juiste image download voor het huidige device?

Met gebruik van de 'w' grootte indicator, vertellen wij tegen de browser hoe breed de te laden afbeelding is (dus niet de weer te geven grootte!). Hierdoor kan hij er al bepalen welke afbeelding hij dient te downloaden uit de srcset. Deze manier van notatie is dus qua resultaat efficiënter dan het vorige <img> voorbeeld. We passen de 'w' indicator als volgt toe:

(U kunt dit voorbeeld niet zelf uitvoeren) :

```
<img src="skyline_1x.jpg" srcset="skyline_1x.jpg 300w,
skyline_2x.jpg 500w" alt="Skyline Rotterdam" />
```

De browser kan nu de juiste afbeelding downloaden op basis van de informatie die hij nu al wel weet: de viewport, de pixel dichtheid en (inmiddels) breedte van de afbeelding.

7.8.5 Sizes attribuut voor viewport gerelateerde download

Er is nog één probleem waar we tegenaan kunnen lopen en dat is de uiteindelijke afmeting van de afbeelding op de webpagina. De browser weet van tevoren niet hoe groot wij onze afbeelding willen tonen op het scherm. Hij zal standaard ervan uitgaan dat de afbeelding de volledige viewport breedte in beslag zal gaan nemen.

Uiteraard kunnen wij de gewenste breedte met CSS regels instellen. Maar dan nog zal de browser zeer waarschijnlijk een té grote afbeelding downloaden. Dit komt doordat de browser altijd éérst HTML verwerkt en erna pas CSS code. Dus de afbeelding wordt gedownload, maar de CSS code is nog niet bekeken.

We kunnen het sizes attribuut gebruiken om aan te geven op welke grootte onze afbeelding wordt weergegeven. Hierdoor kan de browser uit het srcset attribuut de juiste afbeelding kiezen. Het sizes attribuut laten wij matchen met onze CSS image breedte, zodat er geen verschil daarin ontstaat.
Voorbeeld:

Willen wij onze afbeelding altijd tonen op 50vw (50% van de viewport), dan stellen wij dat in in CSS middels:
```
img {
   width: 50vw;
}
```
Én vervolgens ook in HTML middels:
```
<img src="skyline_1x.jpg" srcset="skyline_1x.jpg 300w,
skyline_2x.jpg 500w" alt="Skyline Rotterdam" sizes=”50vw” />
```
Willen we echter een verschillende afbeelding afmeting bij verschillende viewports? Dan kunnen we de media query die wij in CSS gebruiken ook gebruiken (exact overnemen) in ons HTML sizes attribuut.

Stel je voor dat wij onze afbeelding boven de 400px graag op 60vw willen tonen, maar in alle kleinere maten de oude 50% van het viewport willen aanhouden, dan kan dat door in CSS code in te stellen:
Voorbeeld:
```
img {
     width: 50vw;
    @media screen and (min-width: 400px) {
          width: 60vw;
     }
}
```
Met in de HTML code:
```
<img src="skyline_1x.jpg" srcset="skyline_1x.jpg 300w,
skyline_2x.jpg 500w" alt="Skyline Rotterdam"
sizes=”(min-width: 400px) 60vw, 50vw” />
```

7.8.6 Alternatieve bestand typen met het picture element

We kunnen zoals met de HTML 5 video en audio elementen, ook bij een afbeelding gebruik maken van een lijst met bronnen.

De opbouw van een picture element is als volgt:
```
<picture>
     <source srcset=”bron.webp” type=”image/webp” />
     <source srcset=”bron.png” type=”image/png” />
     <source srcset=”bron.jpg”  type=”image/jpeg” />
     <img src=”bron.jpg” alt=”Dit is de bron” />
</picture>
```
De browser zal nu van boven naar beneden het lijstje doorlopen en de eerste afbeelding (via het img element) tonen die werkt. Het src attribuut in het img element is ook hier wederom de default waarde indien de browser de bovenliggende files niet kan gebruiken. In

bovenstaand voorbeeld zou het laatste source element dus ook weggelaten kunnen worden…
Hoe valt dit alles nu samen? Want ook binnen het picture element kunnen we gebruik maken van meerdere files in het srcset attribuut en zelfs een media attribuut toevoegen voor gebruik van media queries.

In het volgende voorbeeld gaan we met media queries de juiste file uitkiezen. Natuurlijk kan dit ook in CSS code, maar als dit zoals hier al in je HTML code kan, scheelt dat weer tijd.

```
<head>
    <title>Picture element responsive</title>
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
</head>
<body>
    <picture>
        <!-- verschillende media queries voor verschillende
viewports -->
        <source media="(min-width: 800px)"
        srcset="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bron_l.png"
type="image/png" />
        <source media="(min-width: 800px)"
        srcset="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bron_l.jpg"
type="image/jpeg" />
        <source media="(min-width: 400px)"
        srcset="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bron_m.png"
type="image/png" />
        <source media="(min-width: 400px)"
        srcset="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bron_m.jpg"
type="image/jpeg" />
       <img src="https://s3-us-west-
2.amazonaws.com/s.cdpn.io/1201815/bron_s.jpg"
        alt="Dit is de bron" />
    </picture>
</body>
</html>
```

We zien dat we hier media queries gebruiken om de juiste afbeelding te laten kiezen op basis van de viewport. Daarna wordt er gekeken of het PNG bestand gebruikt kan worden en indien dat niet het geval is, wordt de jpg variant gebruikt.

7.8.7 Inline images

Wat ook laadtijd kan besparen is het gebruik van een datastream in het src attribuut van een img element. We vullen dan dus geen absoluut of relatief pad in, maar een een verwijzing naar een plaatje middels een zogeheten datastream.

7 Responsive design

Met behulp van tools is een plaatje om te zetten naar een datastream. De image file wordt hierdoor rechtstreeks in de webpagina geladen. Dit kan een voordeel of een nadeel bieden, afhankelijk van de omvang van de file. Doordat het bestand niet meer door de site bezoeker gedownload (en e.v.t gecached) hoeft te worden, scheelt dit een HTTP request. Maar, bij gebruik van een groot plaatje, wordt de bestandsgrootte van de .html file weer erg groot, wat het laden van de algehele webpagina kan vertragen. Uit ervaring moet blijken wat interessanter is.

Onderstaand volgt een voorbeeld van het toevoegen van een plaatje aan uw website, met behulp van een datastream:

```
<head>
    <title>Picture element responsive</title>
      <meta name="viewport" content="width=device-width,
       initial-scale=1.0">
</head>
<body>
<img src="
wgHBgoICAgLCgoLDhgQDg0NDh0VFhEYIx8lJCIfIiEmKzcvJik0KSEiMEExNDk7Pj4+JS5
ESUM8SDc9Pjv/2wBDAQoLCw4NDhwQEBw7KCIoOzs7Ozs7Ozs7Ozs7Ozs7Ozs7Ozs7O
zs7Ozs7Ozs7Ozs7Ozs7Ozs7Ozs7Ozv/wAARCAAgACADASIAAhEBAxEB/8QAFwAAAwE
AAAAAAAAAAAAAAAAAABAUGB//EACgQAAEEAQMEAQUBAQAAAAAAAAECAwQRBQYSIQATMUFRB
yJCgZEjM//EABkBAAIDAQAAAAAAAAAAAAAAAAMFAQIEBv/EACgRAAIABAQEBwAAAAAAAAA
AAAECAAMEEQUSITETQVFhFCIygaHw8f/aAAwDAQACEQMRAD8AY661pPdzDmNxkpyLHhrKV
uMrUhbixwqzwaBsV4NXzxRcSJrvTmWTKkqlZSE2pKXgh8vhaCRe1BIVuHzQ8F3KY2A/i9
cQYMlO11jINIVwQD94oi64I5B+COtP1vqh7TMCMuIllcl90gIeQogoA+48EcglPv30sQ58
0xyRYx3FUop+DR00tXDqdxv3vuOZvy06QNrTWM3Tym2oOPU4dw7kiQ0vsgkWEpIrcr354r
2boHTH1JTkpiIOWjpZffdCGXI6TsJPACgSSDdCxfnmqvoSN9W/wDkmVh/gOuNP/0pSU/wE
/v31GalyMTLaglz4LKmWH1JUEKSEm9o3EgEiybP76mZUWOdGv2itHhAeWaeokZTb1g31v8
AdOkarqqImBJjarjxVSZONSUuNBZTvaIUD6PKSrdfxuv1Spev9IZd9pvI49xSU3tclxUOJ
b458FR5oeB8dXDjbbzS2nUJcbWkpUhQsKB8gj2OsfzOgsu3qGTFxePcein/AFZXYSgIP47
lHyCaomzV10afxEN0FwdxaFmFeFqV4dSxVkHlbNbTp7E6djFbnfp7hsljnJmGb7MlSFOs9
hy2nieQKJoA+BRAF+wK6W6F0LKjzm8vl2uz2qVHjrAKiSOFKH41fA82PVc3GBxicPg4ePA
SFMNAL2kkFZ5UQTzRUSf30w6IKdCwcixjK2L1KSnplfMpJAJ3tt8/kf/Z"
alt="5hart-it logo" />
</body>
</html>
```

Het encoderen van een afbeelding zoals bovenstaand gebeurt niet vaak. Dit kan vooral interessant zijn voor zware websites, die moeten omgaan met vele duizenden bezoekers en pagina-laad acties. Daarom zullen wij dit verder ook niet behandelen in deze cursus.

Mocht uw interesse hierin toch zijn gewekt en u wilt zelf afbeeldingen omzetten naar het base64 datastream formaat, dan is dat niet iets wat wij eenvoudig met de huidige basiskennis kunnen doen. Een website die deze klus bijvoorbeeld voor ons kan klaren is: Base 64 Online: http://base64online.org/.

7.9 Samenvatting

Het ontwerpen van een webpagina die geschikt is voor mobiel gebruik, is tegenwoordig een groot onderdeel van het gehele webdesign traject.

7 Responsive design

Omdat er veel verschillende device formaten zijn, is het aan te raden gebruik te maken van een flexibele layout. Dit is te realiseren door
het flexbox model, het gebruik van relatieve meetvormen en media queries om de nodige breakpoints te verwerken.

Ook kunnen we tabellen geschikt maken voor mobiel gebruik door stukken weg te laten of aan te passen. Voor afbeeldingen kunnen wij de bestandsgrootte, het bestandstype, de compressie, de alternatieven, caching mogelijkheden en nog meer instellen.

7 Responsive design

8 Frameworks

8.1 Inleiding

Veel developers gebruiken zogeheten frameworks om onder andere het ontwikkelproces van bijvoorbeeld een webpagina te versnellen of vereenvoudigen. Omdat er veel redenen zijn om wel of niet voor het gebruik van frameworks te kiezen én gezien er al tal van frameworks beschikbaar zijn, is dit hoofdstuk gewijd aan beide aspecten.

Eerst zullen wij gaan kijken naar redenen om wel of geen framework te gebruiken. Tot slot zullen wij gaan kijken welke frameworks er zoal zijn.

8.2 Wel of geen frameworks gebruiken

Het gebruik van een framework is binnen de IT geen uncommon practice. Een framework is vaak bedoeld om het werken met een bepaalde programmeertaal, of zelfs het werken binnen een project eenvoudiger te maken.

Binnen dit hoofdstuk kijken wij vooral naar front-end frameworks. Dat wilt zeggen; frameworks die de toepassing vinden op het HTML, CSS en JavaScript vlak.

Onderstaand volgen een aantal voor- en nadelen van het gebruik van frameworks.

Voordelen:

- Verkorting ontwikkeltijd
- Browser onafhankelijk ontwikkelen
- Regelmatige toepassing van design patterns en best practices
- Overzichtelijke code

Nadelen:

- Core talen zoals CSS en JavaScript hoeven mogelijk niet bekend te zijn
- Mogelijk hogere leercurve
- Overhead aan ongebruikte code

8.3 Overzicht front-end frameworks

Er zijnontzettend veel frameworks beschikbaar. Met de dag worden er nieuwe frameworks ontwikkeld en toegepast. Dit is ook de reden dat wij in deze paragraaf enkel een klein aantal van de meest gebruikte front-end (HTML, CSS en JavaScript) frameworks gaan bespreken. Van elk van deze frameworks zullen wij de meest voorname voor- en nadelen bespreken.

8.3.1 Normalize.css

Normalize is eigenlijk geen framework zoals wij dat wellicht kennen van andere frameworks, maar dit is wel een goed 'framework' om mee te beginnen.

Normalize bestaat enkel uit één .css bestand, met daarin allerlei instellingen om ervoor te zorgen dat veel browser quirks (eigenaardigheden) opgelost worden, zonder dat wij er zelf naar hoeven te kijken.

Wat doet Normalize.css nog meer?
Reset van CSS eigenschappen om browser specifieke CSS toepassingen weg te halen; dus meer consistentie in opmaak over meerdere browsers
Browser specifieke CSS problemen of bugs aanpakken waar mogelijk
Er zijn uiteraard meer frameworks die een zogeheten 'CSS reset' uitvoeren, maar Normalize verschilt hierin op het volgende vlak (quote):

"Normalize.css preserves useful defaults.

Resets impose a homogenous visual style by flattening the default styles for almost all elements. In contrast, normalize.css retains many useful default browser styles. This means that you don't have to redeclare styles for all the common typographic elements.

When an element has different default styles in different browsers, normalize.css aims to make those styles consistent and in line with modern standards when possible."

Om de Normalize.css file te kunnen toepassen, hoeft deze enkel geïmporteerd te worden in je HTML of CSS document. Een actuele verwijzing naar de meest recente normalize.css file vindt u hier: https://github.com/necolas/normalize.css/blob/master/normalize.css.

8.3.2 Modernizr

Een veel ervaren probleem is het verschil in ondersteuning van HTML, CSS en JavaScript features tussen verschillende browsers. Bij het schrijven van code kunnen wij hopen dat de eindgebruiker een bepaalde browser gebruikt; dat is natuurlijk een slechte manier van programmeren. Een andere manier is om van te voren te weten welke browser eigenaardigheden er zijn en hoe wij die kunnen omzeilen. Modernizr gaat op een snelle manier actief op zoek naar (on) mogelijkheden van de browser en lost deze problemen op.

Over het toepassen van Modernizr op JavaScript, maar ook binnen CSS files, leest u hier: https://modernizr.com/docs/#using-modernizr-with-css.

8.3.3 Bootstrap

In 2011 opgestart door medewerkers van Twitter, inmiddels is Bootstrap uitgegroeid tot één van de meest toegepaste frameworks voor front-end development. Bootstrap bevat onderdelen voor:
* Mobile first design
* Browser onafhankelijke CSS code
* Form opmaak
* Grid lay-outs

Om Bootstrap toe te passen op onze webpagina, dienen we naar de Bootstrap libary te verwijzen, als lokale file op de server, of als link naar een CDN (content delivery network). Vervolgens worden alle classes waar wij een specifieke Bootstrap 'class' eigenschap aan hebben toegevoegd, verwerkt en/of opgemaakt door Bootstrap.

Bekijk het onderstaande voorbeeld voor een demonstratie:

```
<html>
<head>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
```

```
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">

<head>
</body>
<button class="btn btn-success">Testknop!</button>

<div class="form-group has-success">
  <label class="form-control-label" for="inputSuccess1">Input with
success</label>
  <input type="text" class="form-control form-control-success"
id="inputSuccess1">
  <div class="form-control-feedback">Success! You've done it.</div>
  <small class="form-text text-muted">Example help text that remains
unchanged.</small>
</div>
<div class="form-group has-warning">
  <label class="form-control-label" for="inputWarning1">Input with
warning</label>
  <input type="text" class="form-control form-control-warning"
id="inputWarning1">
  <div class="form-control-feedback">Shucks, check the formatting of that
and try again.</div>
  <small class="form-text text-muted">Example help text that remains
unchanged.</small>
</div>
<div class="form-group has-danger">
  <label class="form-control-label" for="inputDanger1">Input with
danger</label>
  <input type="text" class="form-control form-control-danger"
id="inputDanger1">
  <div class="form-control-feedback">Sorry, that username's taken. Try
another?</div>
  <small class="form-text text-muted">Example help text that remains
unchanged.</small>
</div>
</body>
</html>
```

Omdat het niet via een server draait zal het niet helemaal werken, maar het verschil in look en feel met en zonder link naar de css is overduidelijk.

8.3.4 Foundation

Net als Bootstrap voorziet ook Foundation in de behoefte van mobile first en browser onafhankelijke development. Dit is per 2017 nog steeds één van de grootste alternatieven voor Bootstrap.

Foundation promoot zichzelf naast met het bovenstaande, ook dat er een daadwerkelijk bedrijf (ZURB) achter dit framework zit. Hierdoor wordt er ook tijdens de ontwikkeling van het framework erg veel getest in live omgevingen.

Kijk hier voor de website van ZURB Foundation: http://foundation.zurb.com/.

8.3.5 jQuery

Één van de eerste bekende JavaScript libraries is jQuery. Deze library is ondertussen al een aantal jaren op vele duizenden sites in gebruik. Veel nieuwe libraries zijn ook weer op jQuery gebaseerd.

jQuery richt zich vooral op het vereenvoudigen van:

- maken van AJAX calls
- gebruik van JavaScript events
- maken van invoer controles
- maken van lightboxes
- enzovoorts...

Voor een compleet overzicht en een korte introductie, bekijk deze webpagina: https://learn.jquery.com/about-jquery/how-jquery-works/.

8.3.6 Angular

Deze library, ondertussen onder leiding van Google, is voor veel webapplicaties de basis als het gaat om front-end frameworks. De focus ligt op het ontwikkelen van moderne (offline) web applicaties en goede performance. Het is bedoeld om alle krachten van HTML, CSS en JavaScript te bundelen, maar zonder de vervelende eigenaardigheden.

Een korte opsomming van krachtige mogelijkheden van Angular:

- Two way data binding (content sturen in de HTML code)
- Templates (HTML structuren uitgebreid met opdrachten voor betere programma structuur)
- MVC (Model View Controller)
- Dependency injection (afhankelijkheden eenvoudiger aan te vragen)
- Directives (eigen HTML elementen)
- Testing

Een uitgebreid overzicht van de bovengenoemde mogelijkheden van Angular vindt u hier: https://code.tutsplus.com/tutorials/5-awesome-angularjs-features--net-25651.
Overige frameworks

- HTML 5 boilerplate
- LESS & SASS CSS pre-processors
- ReactJS
- NodeJS
- Polymer

Index