IK WIL

Cascading Style Sheets

CSS 3

vijfhart IT-OPLEIDINGEN

Introductie

1. Voorstelrondje
2. Agenda
3. Dagindeling
4. Cursusmateriaal
5. Doel van deze cursus

Agenda dag 1 en 2

Dag 1

- H1 Inleiding
- H2 Opbouw van een CSS document
- H3 Lettertypen en tekstopmaak
- H4 Kleurgebruik, afbeeldingen en animaties

Dag 2

- H5 Padding, margin en borders
- H6 Positionering
- H7 Responsive design
- H8 Frameworks overview (optioneel)

Dagindeling

- 8:45 - ..... Begin cursusdag
  - Introductie
  - H1 Algemeen
  - H2 Hoofdstukinhoud
    - Theorie
    - opdrachten maken
    - opdrachten bespreken
- 10:30 - 10:45 Koffiepauze
- 12:00 - 12:45 Lunchpauze
- 14:30 - 14:45 Koffiepauze
- ..... - 16:00 Einde cursusdag

Cursusmateriaal

- Online leeromgeving
  - PDF met sheets
  - Content uitgeschreven
- CodePen.io oefen omgeving
- Bonus materiaal

H1 - Inleiding

H1 - Inleiding

1 Doelstelling

2 Handige websites

3 Tools

4 Introductie CSS

5 Introductie W3C

6 Debugging

Doelstelling

Na deze cursus ben je in staat zelfstandig HTML webpagina's op te maken met Cascading Style Sheets.

Dit houdt onder andere in dat je:

- Elementen kunt positioneren
- Elementen kunt opmaken qua kleur en omvang
- Formulieren kunt opmaken
- Teksten kunt vormgeven
- ..... Nog véél meer ;)

Handige websites

Een kleine greep uit een selectie met praktische websites om tijdens en na deze cursus bij de hand te houden:

- Browser ondersteuning: Can I use it?
- Browser ondersteuning CSS en Javascript: Quirksmode
- Web development oefeningen: W3 schools
- HTML en JavaScript naslag: MDN
- Browser prefixes: Peter Beverloo
- Chrome DevTools: DevTools
- En de volgende webpagina verdient een losse vermelding:
- CodePen.IO; een online editting omgeving

Tools

Tijdens deze training gebruiken wij:

- Google Chrome, met Developer Tools

Qua editor(s) kunnen we gebruik maken van:

- NotePad++
- https://CodePen.io
- Iedere andere tekst editor zal (in principe) voldoen

Introductie CSS

- CSS staat voor Cascading Style Sheets
- Taal om webpagina's vorm te geven
- Voorheen opmaak met HTML zelf maar onpraktisch, why? https://codepen.io/5hart/pen/jmZLdr
- BETER, met CSS: https://codepen.io/5hart/pen/zwRdeP

Opdracht

Gebruik onderstaande voorbeeld:
https://codepen.io/5hart/pen/zwRdeP

Pas de code zodanig aan dat de paragraaf niet groen, maar in een andere kleur wordt weergegeven.

Bekijk hier eventueel alle mogelijke CSS kleurnamen:
http://www.colors.commutercreative.com/grid/

Introductie W3C

Het World Wide Web Consortium, ook wel W3C, is een bedrijf dat zich richt op het opstellen van regels en richtlijnen bij het gebruik van webtalen zoals HTML, CSS en JavaScript.

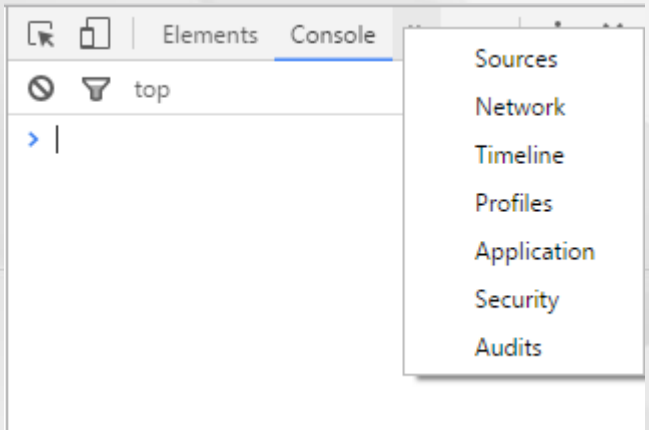
Gebruik in de cursus de pagina:
https://www.w3.org/
voor een beschrijving van alle standaarden omtrent 'front-end' webtalen

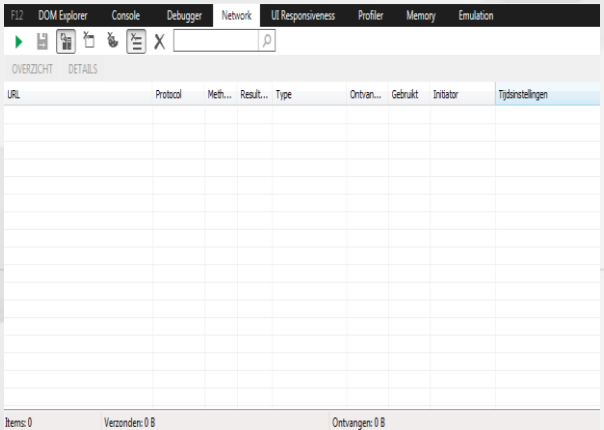
Debugging

- Doorzoeken (‘inspecteren’) van elementen en stijlen (HTML & CSS)
- Foutopsporing voor met name JavaScript code

Middels: 'Hulpprogramma's voor ontwikkelaars‘:

- Google Chrome: <ctrl><shift><i>, óf <f12>
- Internet Explorwer: <f12>
- Opera: <ctrl><shift><i>
- FireFox: gebruik bijvoorbeeld de extensie





Vragen?

H2 – Opbouw van een CSS document

Leerdoelen

- 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

H2 – Opbouw van een CSS document

1 Selectors

2 HTML element typen

3 Opbouw van een HTML document (voorkennis)

4 HTML element typen (voorkennis)

5 CSS code toepassen op een HTML pagina

6 CSS overerving

Selectors


```
<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>
```

Selectors - prefixes

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:

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

Let op: In deze cursus laten wij ze achterwege i.v.m. vele regels extra code. In de praktijk echter wel aan te raden te gebruiken.

Overzicht met prefixes:

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

Opdracht

Gebruik de volgende HTML code:

https://codepen.io/5hart/pen/EmQwPR

Voeg nu de volgende CSS regels toe aan het style element in het boven of onderstaande codeblok:

p {color: grey}

Het p element heeft nu alle paragrafen geselecteerd: de tekst wordt grijs weergegeven. Voeg nu toe:

div {color: purple}

De regel voor het div element geldt voor **alle** subelementen waar nog geen opmaak aan was toegekend.

Selectors

We kunnen verschillende typen selectors ('selecteerders') onderscheiden in CSS:

- Type selectors (input, p, div, span, article, etc.)
- Class selectors (div class="....")
- Id selectors (div id="....")
- Descendant selectors (div>span)
- Overige selectors (input[type="tekst"])
- Pseudo classes (:hover, :active)

We gaan een aantal demo's hiermee doen:

Selectors

DEMO's:

https://codepen.io/5hart/pen/ZKrrWa

Opbouw van een HTML document

Opbouw basale webpagina volgens W3C HTML 5 standaarden:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Pagina titel</title>
    </head>
    <body>
    </body>
</html>
```

Opbouw van een HTML document

- Elementen zijn in HTML beschreven met behulp van tags.
- Een tag wordt geopend met een < en afgesloten met een >. Veel elementen kunnen ook inhoud bevatten.
- Voorbeelden van tags (en daarmee elementen) zijn:

  ```
  <body>
  <div>
  <p>
  ```

- Iedere tag dient ook afgesloten te worden. Dit is bij de meeste tags middels een gelijknamige tag, met een backslash erin. Respectievelijk: `</body>`, `</p>` en `</div>`.
- Elementen zonder inhoud worden in de openingstag ook afgesloten. Zoals: `<input />` en `<br/>`.

HTML Element typen

Grofweg zijn HTML elementen te onderscheiden in block- en inline elementen. Dit is een standaard ingestelde 'display' CSS eigenschap.

De verschillen tussen deze twee typen elementen zijn:

- Block elementen hebben impliciet een 'newline' achter zich, inline elementen niet.
- Block elementen mogen block en/of inline elementen bevatten. Behalve een in een <p>.
- Inline elementen mogen gegevens en enkel andere inline elementen bevatten.
- Block elementen vullen de volledige breedte op wat mogelijk is, inline elementen gebruiken alleen de horizontale ruimte die ze nodig hebben.

HTML Element typen

Een element is ook altijd **replaced** of **non-replaced**

- Replaced: wordt runtime vervangen door de grafische component en hebben geen tekst tussen de tags (<img>, <input>, <hr>)
- Non-replaced: worden niet vervangen, dus bevat bijvoorbeeld de ingevoerde tekst (<p>, ,)
- Replaced elementen hebben intrinsieke dimensies, zoals breedte en hoogte: we hoeven deze dimensies daarom niet te bepalen met een stylesheet, mag wel.

Webpagina's opmaken met CSS

- De norm is om pagina's op te maken met CSS
- Met CSS verwijzen wij naar één of meer elementen en passen daar opmaak regels op toe.
- Met relatief weinig code kun je véél verschillende elementen van een pagina kunt stijlen.
- De code is eenvoudig toepasbaar op meerdere webpagina's

Wat zou de volgende code doen?

```
div {
  background-color: yellow;
}
```

Webpagina's opmaken met CSS

- Gangbaar is dat CSS code in een aparte .css file wordt opgeslagen.
- Vanuit de HTML pagina wordt verwezen naar een .css document.
- Het mogelijk meerdere .css files aan een document te koppelen (twee voor weergave op PC, één voor tablet weergave en één voor print opmaak).

Webpagina's opmaken met CSS

Manieren om CSS code op je HTML document toe te passen:

- **Embedded**: in de HTML pagina verwerkt, maar wél buiten de beschrijving van de elementen. Vaak wordt bovenaan de pagina een HTML codeblok gemaakt, waarin wij onze CSS code kwijt kunnen.

  Voorbeeld:

```
<style type="text/css">
   p { color: green; }
</style>
```

Webpagina's opmaken met CSS

Manieren om CSS code op je HTML document toe te passen:

- **Inline**: zoals de naam al zegt: in de regel. Dus via deze weg kun je CSS code als zogeheten style attribuut toekennen aan een willekeurig HTML element. Kort en snel, maar niet flexibel of beheerbaar.

  Voorbeeld:

```
<p style="color: green">
```

Webpagina's opmaken met CSS

Manieren om CSS code op je HTML document toe te passen:

- **Linked:** is de meest gebruikte manier voor het toekennen van CSS code aan een HTML pagina. Er wordt middels een <link> tag met een src attribuut, verwezen naar een .css bestand dat op de betreffende pagina moet worden toegepast.

```
<link rel="stylesheet" type="text/css"
href="naam.css" media="all">
```

Webpagina's opmaken met CSS

Manieren om CSS code op je HTML document toe te passen:

- **Imported**: is een variant op de bovengenoemde linked toepassing. Het importeren van een stylesheet gebeurt hier met het @import commando, binnen <style> HTML element tags.

```
<style type="text/css">
  @import url(bestandsnaam) mediatype;
</style>
```

Overerving binnen CSS

- 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*.

DEMO: https://codepen.io/5hart/pen/XRqrLB

```
(wijzigen van de background-color van het
invoerveld naar 'inherit')
```

Meerdere stijl definities

- Wanneer (bijv.) een externe (linked/imported) en interne (embedded) stylesheet zijn gedefinieerd en er vervolgens ook inline styles worden toegepast.
- Er zal één algehele virtuele stylesheet aangemaakt worden.
- 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.

Meerdere stijl definities

Een CSS regel kan met een andere regel overlappen of in conflict zijn

- De meest specifieke regel wint
- 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

Meerdere stijl definities

Demo prioriteiten: https://codepen.io/5hart/NjyzEv

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

#vb1{color: black;}

body div p{color: green;}

div.voorbeelden {color: red;}

Meerdere stijl definities

In het 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  */
```

Meerdere stijl definities

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.
- Om een regel toch een hogere prioriteit te geven in geval van gelijke specificiteit kan dit worden aangegeven met !important

DEMO: https://codepen.io/5hart/rmEpEN

Commentaar in CSS

Commentaar notatie

/ * *hier staat code* */

Toelichting

Deze combinatie van een slash en ster (en vise versa), wordt gebruikt om commentaar over meerdere regels in te voegen.

Vragen?

H3 – Lettertypen en tekstopmaak

Leerdoelen

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

H3 - Opmaak

1. Lettertype keuze
2. Lettertype karakteristieken
3. Tekst uitlijning
4. Meer mogelijkheden met web fonts, @font-face en icon fonts

Lettertype keuze

Fonts worden onderverdeeld in vijf groepen:

- Serif fonts: schreef (kleine decoraties op letters) en verschillende letterbreedte. Voorbeeld fonts: Georgia, Times.
- Sans-serif fonts: verschillende letterbreedte, zonder schreef. Voorbeeld fonts: Arial, Helvetica, Verdana
- Monospace fonts: vaste letterbreedte, met en zonder serifs. Voorbeeld fonts: Andala Mono, Courier, Courier New
- Cursive fonts: extremere ronde vormen en decoraties dan bij serif fonts, vaak gelijkenissen met handschriften. Voorbeeld fonts: Author, Comic Sans
- Fantasy fonts: verzamel categorie, vaak gerelateerd aan verhalen, films, enzovoorts. Voorbeeld fonts: Klingon, Western, Woodblock.

Lettertype keuze

- Om een element een gewenst lettertype te kunnen geven, maken wij gebruik van de **font-family** property.

DEMO's:

https://codepen.io/5hart/pen/JNZmZM/

https://codepen.io/5hart/pen/mmKzxa/

Karakteristieken – font-size

- Voor het instellen van tekst afmeting

Voor het instellen van deze eigenschap kunnen we een onderscheid maken tussen beschrijvende en niet beschrijvende meetvormen en tussen absolute en relatieve meetvormen.

- Absolute beschrijvende meetvormen: xx-small, x-small, small, medium, large, x-large en xx-large
- Relatieve beschrijvende meetvormen: smaller of larger
- Niet beschrijvende absolute maatvormen: cm, in, mm, pc, pt, px
- Niet beschrijvende relatieve maatvormen: em, ex, %, vw

DEMO: https://codepen.io/5hart/pen/BRVqEa (beschrijvend)
DEMO: https://codepen.io/5hart/pen/jmKQbV (niet beschrijvend)

Karakteristieken – font-weight

Font-weight geeft aan of tekst vet gedrukt is en in welke mate.

Men kan als waarde 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.*

***Opdracht**: maak in CodePen een paragraaf en maak deze vervolgens dik gedrukt:*

https://codepen.io/5hart/pen/BRgYmM

Karakteristieken – font-style

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

Qua waarde kan gekozen worden voor:

- normal
- Italic
- Oblique

Het 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.

Tekstuitlijning

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.*

Tekstuitlijning

Voorbeelden van:

text-indent: https://codepen.io/5hart/pen/eWKQBy

tekst-align:

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.

Tekstuitlijning

- **vertical-align**: Uitlijning van stukken tekst in een inline element; ten opzichte van het midden van de regel.

- **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.

Tekstuitlijning

Voorbeelden van:

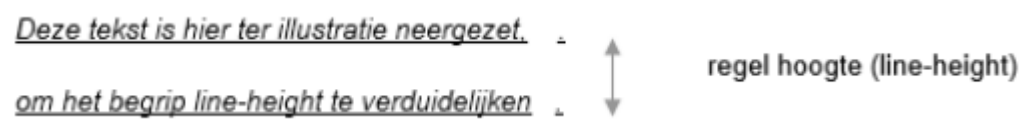
word-spacing: De eigenschap word-spacing doet in principe hetzelfde als letter-spacing, alleen dan toegepast op woorden. We kunnen ook hier een negatieve waarde opgeven om ervoor te zorgen dat de woorden juist dichter bij elkaar komen te staan.

Voorbeeld: https://codepen.io/5hart/pen/jmKQYL

Tekstuitlijning

line-height: hoeveel ruimte willen we tussen regels willen zien?

Hoe de regelhoogte precies wordt gemeten kunt u zien in het volgende plaatje:



Tekst decoratie

Met de text-decoration eigenschappen kunnen we een lijn trekken over, onder of door een tekst.

Qua waarden kan gekozen worden voor:

- none
- underline
- overline
- line-through
- blink

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.

Voorbeeld: https://codepen.io/5hart/pen/KmeroX

Meer mogelijkheden - 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.

Meer mogelijkheden - Web fonts

Helaas verwachten verschillende browsers verschillende soorten lettertypen:

Browser	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)

We moeten vaak aan alle versies van een font zien te komen. Op het internet zijn zeer veel gratis lettertypen te vinden. Op sites, zoals Font Squirrel, kunnen we daarnaast m.b.v. een font bestand, de andere bestandstypen (+CSS code!) genereren.

Meer mogelijkheden - @font-face

We gebruiken de @font-face selector om zelf een nieuw type te declareren:

https://codepen.io/5hart/pen/XRLEgZ

(we hoeven hiervan niet alle code te begrijpen)

Verderop in de stylesheet kunnen we naar dit lettertype verwijzen:

```
p {
  font-family: 'Hobbiton Brush', serif;
  font-size:   25pt;
}
```

Zorg er wel voor dat je alternatieve lettertypen opgeeft voor oudere browsers die @font-face niet ondersteunen. In dit geval is de groep ‘serif’ gebruikt.

Meer mogelijkheden - @font-face

DEMO https://fonts.google.com

Meer mogelijkheden - icon fonts

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.

1) Kies een font op bijvoorbeeld: We Love Icon Fonts

2) Kilk op 'add', de CSS code die wij kunnen gebruiken verschijnt dan. Deze code is anders dan wij zijn gewend, bijvoorbeeld:

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

Let op: het gebruik van icon fonts in SVG formaat wordt al even *niet* meer ondersteund door Chrome.

H4 – Kleurgebruik, afbeeldingen en animaties

Leerdoelen

- 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

H4 - Kleurgebruik, afbeeldingen en animaties

1 Display eigenschap

2 Kleurgebruik in CSS

3 Achtergrondplaatjes

4 Background instellingen

5 List items

6 Schaduw-, transparantie- en gradienteffecten

7 Transformaties en animaties in 2D en 3D

Display eigenschap

- Hoe HTML elementen binnen een tekst worden weergegeven hangt af van de *display* eigenschap.
- 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.
- Ook kunnen we block elementen inline weergeven of andersom.

Display eigenschap

De meest voorkomende waarden zijn:

block, inline, inline-blocklist-item, flex, flow, grid en none

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.

DEMO: https://codepen.io/5hart/pen/gWKZwR

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.

Display eigenschap

Opdracht:

https://codepen.io/5hart/pen/NjBgGG

Wijzig hierin van via pull down menu de display eigenschap van een div. De beschikbare waarden *zijn block, inline, inline-block* en *none*.

Kijk wat er gebeurt.

Kleurgebruik

De CSS eigenschappen die ons in staat stellen iets met kleur te doen zullen we hieronder bespreken:

- background-color
- border-color
- color

De waarden (de kleur) voor de genoemde eigenschappen kunnen we invullen met rgb waarden (numeriek of procentueel), met hexadecimale waarden of met kleurnamen.

Kleurgebruik

Zoals u kunt raden, zijn er veel meer mogelijke kleuren te maken dan we in onderstaande tabel tonen. We gaan in de onderstaande tabel echter uit van de beschikbare kleurnamen:

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	
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

Handige kleur selectie tool! Hier te vinden:
https://www.w3schools.com/colors/colors_picker.asp

Kleurgebruik

Een overzichtelijke weergave van de verschillende kleuren vind je op de volgende webpagina:

http://illinoisdouble.com/html-color-wheel/the-hexagon-colors-chart-web-safe-color-codes-web-safe-color-codes-html-color-wheel/

- 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.

Achtergrond plaatjes

background-image
We kunnen m.b.v. de background-image property een achtergrond plaatje instellen op ieder willekeurig block en inline element.

Let op: de background-image eigenschap wordt niet automatisch overgeërft.

De inhoud van de eigenschap mag zijn:

- url(pad van de file)
- none

DEMO: https://codepen.io/5hart/pen/vmaZzB

Achtergrond plaatjes

background-repeat

- Een achtergrondplaatje wordt by default herhaald weergegeven, zowel verticaal als horizontaal.

- Met de background-repeat property kunnen we aangeven of - en zo ja, in welke richting - het achtergrondplaatje herhaald moet worden.

De mogelijke waarden voor deze eigenschap zijn:

- repeat
- repeat-x
- repeat-y
- no-repeat

Opdracht: zorg dat in onderstaande code het plaatje verticaal wordt herhaald: https://codepen.io/5hart/pen/vmaZzB

Achtergrond plaatjes

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.
- 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.

Achtergrond plaatjes

background-position

- Hoe staat de afbeelding zichtbaar binnen de grenzen van de afbeelding url.
- Er kunnen twee waarden ingevuld worden. Namelijk een waarde voor de horizontale positie en voor de verticale positie.
- We kunnen 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.

Achtergrond plaatjes

background-position

(vervolg)

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.

List items

Voor het wijzigen van de stijl van lijsten (<ul> en <ol>), zijn er verschillende CSS properties beschikbaar, waaronder:

- list-style-type
- list-style-image
- list-style-position

Niet alle mogelijke waarden worden hier genoemd. Deze zijn in onze online leeromgeving terug te vinden.

DEMO van alle eigenschappen:
https://codepen.io/5hart/pen/KmBvKQ

Schaduw-, transparantie- en gradienteffecten

box-shadow en tekst-shadow

Syntax voor beide properties:

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

Bij een negatieve waarde voor positie, zal de schaduw respectievelijk links of boven het element uitkomen in plaats van normaliter rechts en onder het element.

Voorbeeld: https://codepen.io/5hart/pen/aWNvmw

Schaduw-, transparantie- en gradienteffecten

Opacity

- Op alle elementen in CSS kunnen wij transparantie toepassen.
- Een opacity van waarde 1 betekent dat het element geheel niet transparant is. 0 is volledig transparant is.
- Vaak zoeken naar de juiste waarde tussen 0 en 1.

DEMO: https://codepen.io/5hart/pen/qmZbdQ

DEMO: https://codepen.io/5hart/pen/MmyKVG

Schaduw-, transparantie- en gradienteffecten

Gradient effecten

- Veel gebruikt effect in foto bewerkingsprogramma's
- Bij een gradient, ook wel 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).

Naslag: https://www.w3schools.com/css/css3_gradients.asp

DEMO: https://codepen.io/5hart/pen/pPZrbK

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'

Elementen in 2D transformeren

transform

Deze property heeft de volgende mogelijke waarden:

- translate(): verplaatsen van een element ten op zichte van de huidige positie met x en y waarde.
- 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. 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!

DEMO: https://codepen.io/5hart/pen/qmrqoM/

Elementen in 3D transformeren

transform

Voor het transformeren in drie dimensies is dezelfde property te gebruiken als voor twee dimensies. Er komt enkel een z-as bij.

De lijst met mogelijke 3d transformatie eigenschappen is nu:

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

En twee nieuwe waarden:

- translate3D(x, y, z)
- scale3D (x, y, z)

DEMO: https://codepen.io/5hart/pen/ZKeBjX

Animaties

animation

- 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.

Animaties

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

DEMO: https://codepen.io/5hart/pen/wdJgvz

Het nu volgende voorbeeld heeft iedere 'animation' eigenschap uitgewerkt:

DEMO: https://codepen.io/5hart/pen/qmmYgq

Animaties

Tot slot nog een overzicht van de mogelijke animatie gerelateerde properties:

- 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?

Animaties

(vervolg):

- 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.

H5 – Padding, margin en borders

Leerdoelen

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

H5 – Padding, margin en borders

1 Ruimtelijke opbouw van elementen

2 Borders

3 Margin en padding

4 Box model

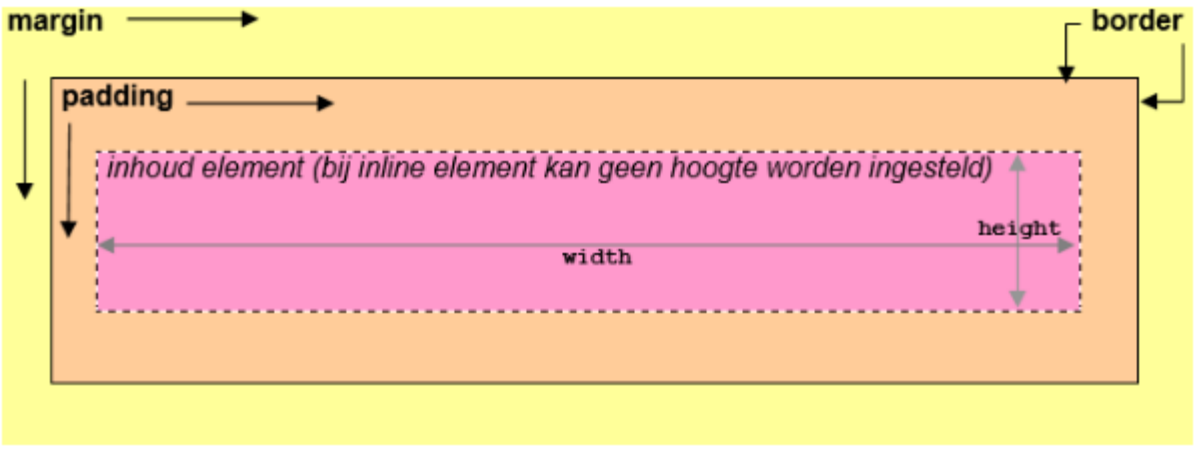
5 Width en height

6 Resize property

Ruimtelijke opbouw van elementen

- Elementen zijn de basis van een document. Met behulp van CSS kunnen we verschillende ruimtelijke eigenschappen van een element beïnvloeden.

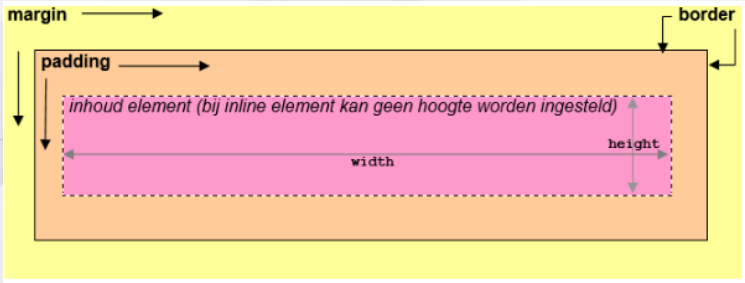
- Deze ruimtelijk eigenschappen worden in onderstaande figuur weergegeven.



Ruimtelijke opbouw van elementen

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. 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. Geen border? Dan is er een onzichtbare van 0px.
- *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 v/h element.



Ruimtelijke opbouw van elementen

- Belangrijk is om te weten dat veel browsers bepaalde CSS eigenschappen al voor ons instellen.

- 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*

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

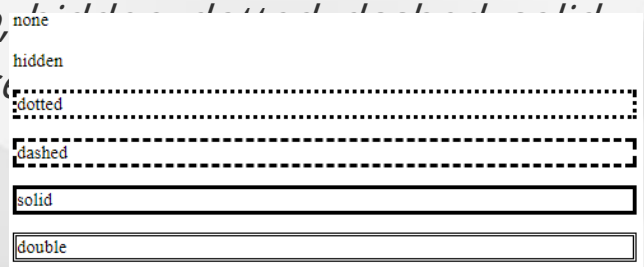
Borders

Voor borders kunnen we drie individuele eigenschappen instellen:

- de vormgeving (style)
- de breedte (width)
- de kleur (color)

border-style

- We kunnen hiermee bijvoorbeeld aangeven of het kader een doorlopend kader moet zijn, of een stippellijn.
- Mogelijke waarden: *none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset*



Opdracht: pas zelf enkele waarden aan:

https://codepen.io/5hart/pen/rmXMvR

Borders

(vervolg)

- We kunnen ook specifiek naar één border verwijzen: *border-top-style, border-right-style, border-bottom-style, of border-left-style.* Dit geldt ook voor alle nog te bespreken border properties.

- *Shorthand* notatie: we kunnen één, twee, drie of vier waarden meegeven voor border-style, border-width of border-color.

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)

Borders

border-width

Wanneer we een kader willen, is het ook van belang dat we aangeven hoe ‘dik’ dit kader is. Je mag een letterlijke dikte in maatvorm naar keuze opgeven, maar ook: ‘thin’, ‘medium’ of ‘thick’

border-color

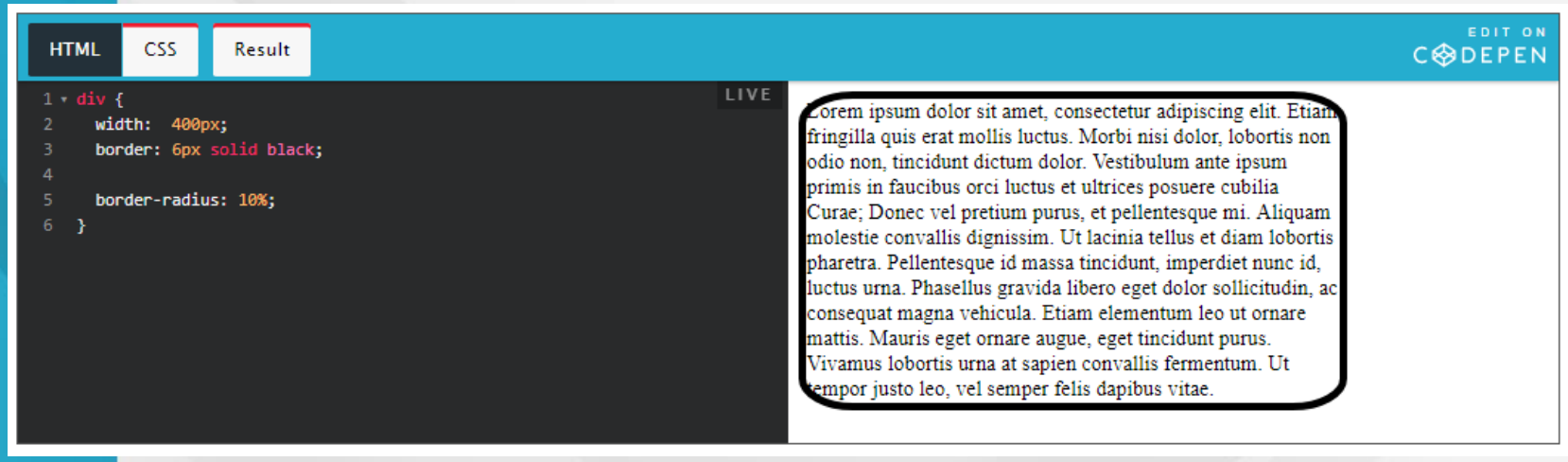
Met *border-color* kunnen we borders een kleur geven. We kunnen ook hier weer onderscheid maken tussen de afzonderlijke zijden van het element. Welke kleuren we kunnen invullen als waarde kunt u teruglezen in hoofdstuk 4. Maar ook ‘transparent’ is mogelijk.

Een veel voorkomend gebruik van border is het maken van een scheidingsteken tussen menu items. We gaan nu een voorbeeld aanpassen:

Borders

border-radius

- 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).

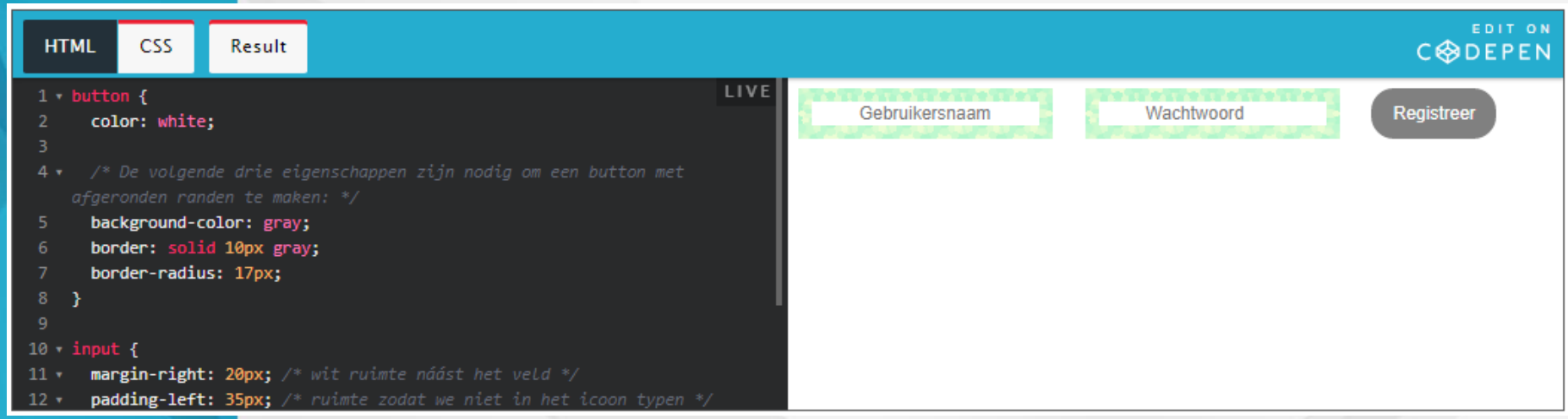


Bekijk zelf ook dit voorbeeld:
https://codepen.io/5hart/pen/mmMGPJ
En dit praktische voorbeeld:

Borders

border-image

We kunnen naast een border met bepaald patroon te vullen, er ook voor kiezen een achtergrond afbeelding er in te plaatsen.



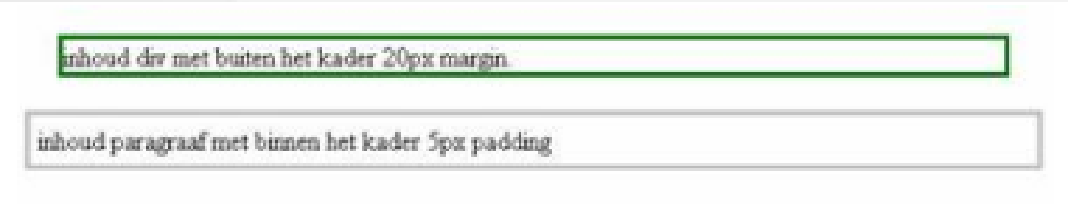
De border-image eigenschap is de korte weergave van: border-image-outset, border-image-repeat, border-image-slice, border-image-source en border-image-width

Padding en margin

- De visuele afstand tussen elementen wordt hiermee ingesteld.
- Met margin geven we de afstand aan van een element tot een volgend element.
- Padding is de (evt) ruimte tussen de inhoud en de border van het element.
- 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*.

Padding en margin

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.

margin: default waarde afhankelijk v/d browser

padding: default waarde 1px

Waarden zijn middels de bekende maatvormen op te geven

Padding en margin

We kunnen zowel padding als margin ook voor individuele zijden van een element opgeven:

https://codepen.io/5hart/pen/NjQbGL

Box model

- Wanneer we elementen naar eigen wens willen plaatsen op een webpagina, kunnen we o.a. gebruik maken van de *position* en de *float* eigenschap (ander hoofdstuk)
- 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 betrekking op deze hele *element box*.

Box model

Containing block

- De positie en grootte van een element box wordt meestal bepaald ten opzichte van het omliggende (containing) block element:

```
<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, ook wel *initial containing* block.

Width en height

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

- 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

Width en height

Onderstaand voorbeeld toont het effect van width en height op verschillende soorten elementen:

Link naar de code: https://codepen.io/5hart/pen/xdoPLa

Width en height

Min-width, min-height, max-width en max-height

- Voor begrenzen van waarden indien geen vaste breedte wordt opgegeven, maar auto of een percentage

- Een kolom met een breedte van 20% is op een telefoon al vrij snel te smal; stel dan bijvoorbeeld een min-width in van 300px.

Resize eigenschap

- Om elementen qua afmetingen variabel te maken naar wens van de gebruiker, kunnen wij de CSS eigenschap 'resize' gebruiken

- Dan kan de gebruiker afhankelijk van de instelling ervoor kiezen de lengte, breedte of beide van een element te wijzigen.

- Mogelijke waarden: *none*, *horizontal*, *vertical* en *both*.

- Het is noodzakelijk *overflow* anders dan 'visible' te zetten; de content zou buiten een element terecht kunnen komen: Het instellen van de resize eigenschap zou dan geen effect hebben.

DEMO: https://codepen.io/5hart/pen/JNNaVb

H6 – Positionering

Leerdoelen

- 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

H6 – Positionering

1 Position property

2 Float

3 Clearing

4 Flexbox

5 Extra aandachtspunten voor positionering

6 Grid

Position property

- Waar een element wordt geplaatst wordt standaard bepaald door de positie en marges van de overige elementen, en van de *document flow* (p verschijnt onder div).

- 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.

- Met *position* kunnen we deze manier van positioneren van elementen aanpassen.

- Mogelijke waarden: static, relative, absolute en fixed

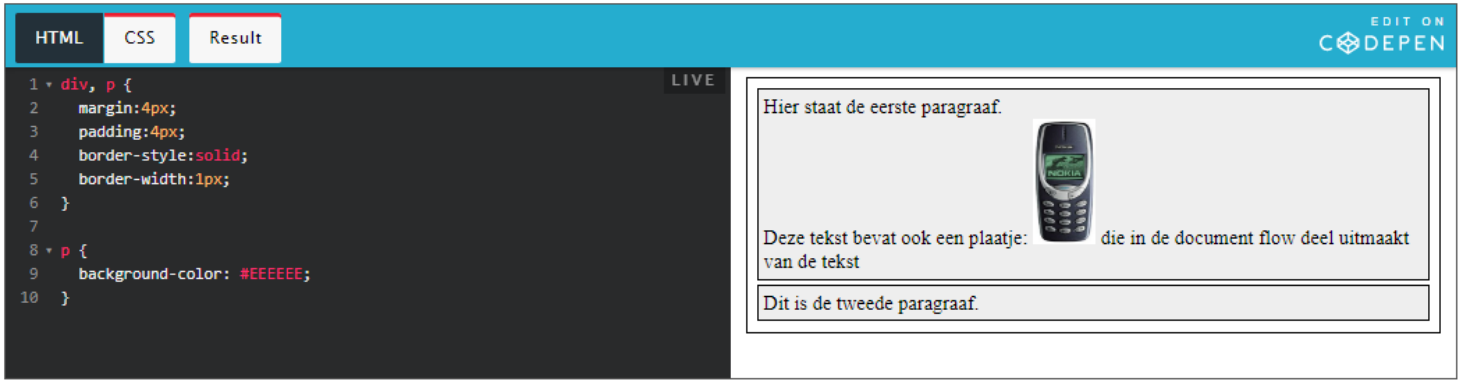
Position property - static

- 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.

Position property - Relative

- Ook hier gaan we uit van de 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.

Position property - Relative



https://codepen.io/5hart/pen/EmqNmK

De bovenstaande 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.

Position property - Absolute

- Het element heeft 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.

Opdracht:

Bekijk het volgende voorbeeld en test de werking van absolute versus relative positioning:

https://codepen.io/5hart/pen/EmqNmK

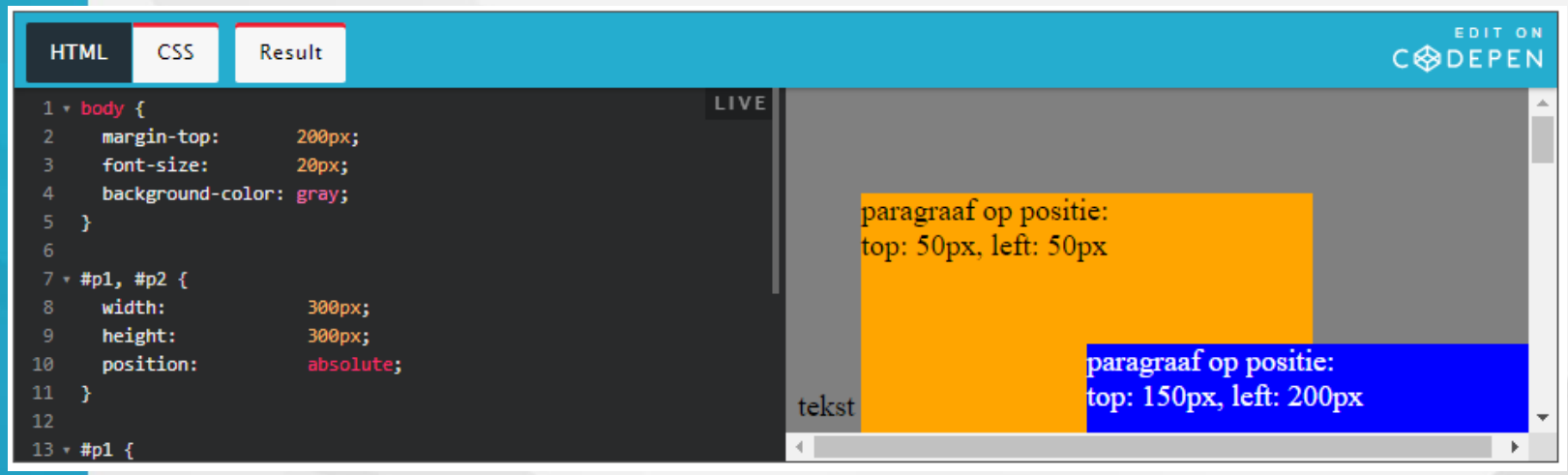
Position property – Fixed

Waarde van position: *fixed*

- 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.
- Een fixed positie wordt onder meer voor kop en voetteksten, reclameblokken en "hulp bij uw aankoop" wizard knoppen.

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.

Position property - Fixed



Opdracht:

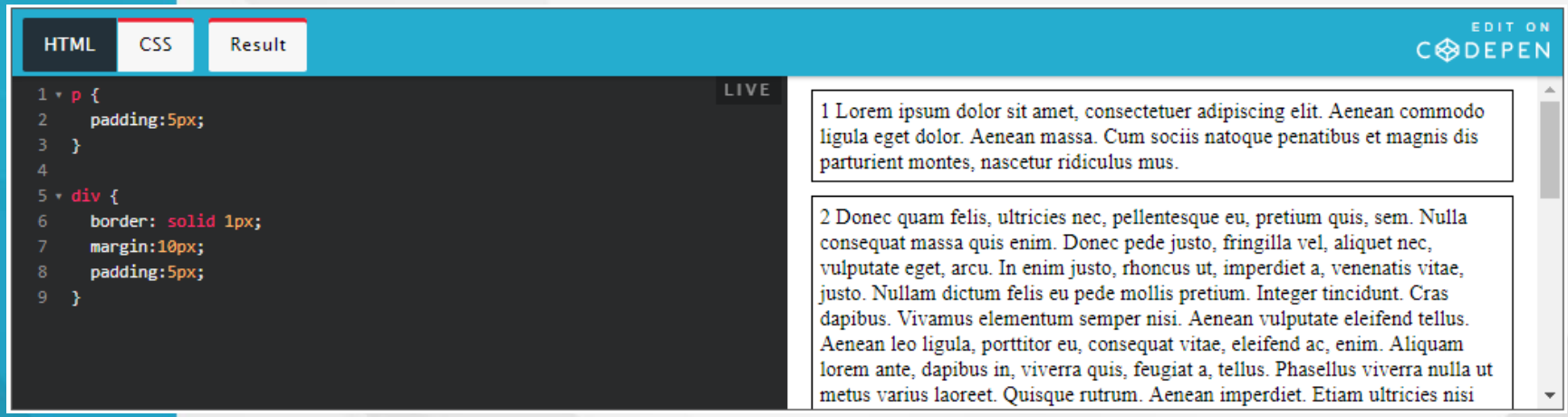
Wijzig position:absolute in position:fixed. Kijk wat er gebeurt.

Float

- Manier om elementen te positioneren buiten de normale flow
- Genereert een block box , ongeacht het type element
- Het float element wordt uit de document flow naar links of rechts verplaatst, tot de grens van het containing block of de buitengrens van het volgende float element
- Geadviseerd wordt float elementen een breedte (width) mee te geven, anders is het effect in de browser (zelfs –per-browser), onvoorspelbaar.

Float

Bekijk deze code op: https://codepen.io/5hart/pen/mmNOvG



Opdracht: Voeg de volgende CSS code toe 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.

Float

Bekijk deze code op: https://codepen.io/5hart/pen/mmNOvG

Opdracht:
Zorg er nu voor dat zowel #d1 als #d2 de waarde float: left krijgt en bekijk het resultaat. (NB zorg ervoor dat zowel de HTML code als de CSS code "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.

Float

Stelregels:

- Een float element kan *horizontaal* niet buiten de kaders van het containing block komen
- 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.
- Bij twee floats met een totale breedte groter dan de scherm breedte: 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.

Clearing

clear property

- Wanneer we niet willen dat bepaalde inhoud van een pagina om een float element heen loopt
- Wanneer we deze eigenschap op een float 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.
- Keywords: 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.

Opdracht: probeer het clear attribuut toe te passen:
https://codepen.io/5hart/pen/QveGXW

Extra aandachtspunten

Overflow

- Meer tekst dan element kan bevatten? Tekst zal over de randen van het element 'vloeien'.
- De overflow eigenschap kan de volgende waarden bevatten: *visible, hidden, scroll, auto*
- Vaak ingesteld op 'auto', zodat er een scrollbalk verschijnt in het element om zo toch, en netjes, de gehele inhoud te kunnen lezen.

Verticale margins

- Let op het inklappen van verticaal aangrenzende margins.
- Inklappen is alleen van toepassing op margins en niet op borders of padding.
- Gebeurt wanneer 2 elementen boven elkaar een margin hebben (één 'bottom', ander 'top'. De hoogste waarde wordt dan aangenomen!

Extra aandachtspunten

Breedte van een element

- Width = breedte + padding + breedte border van het element
- Wijzigen borderbreedte => de totale element breedte groter

Element horizontaal op pagina uitlijnen

- Breedte instellen op zowel het element als op het parent element.
- Margin instellen op de het element, zodat deze daadwerkelijk gecentreerd wordt weergegeven.
- Indien je de breedte van in dit geval de div op een percentage instelt (dus ten opzichte van de parent), zal hij mooi meeschalen bij andere beeld formaten.

DEMO: https://codepen.io/5hart/pen/NjGYaN

Flexbox

- Alternatieve manier van webpagina's opmaken
- Voorheen bedoeld voor netjes uitlijnen van o.a. formulieren
- Bestaat uit flexbox container met daarin flex elementen
- Flexbox element met weer andere flex elementen bevatten
- Flex element mag in iedere richting worden geplaatst en verkleind qua formaat indien nodig
- Zeer veelzijdig
- Vrij complex

Flexbox

- Om een flex container te maken, moet de **display** property van het element op ‘flex’ of ‘inline-flex’ worden gezet
- Ieder element in een flex container wordt als flex element behandeld
- Pas zodra een flex element de property **flex** een waarde heeft gegeven, kan deze zich ook pas gedragen als flex element
- De container bepaalt de richting van de elementen in het geheel; horizontale of verticale positionering t.o.v. elkaar

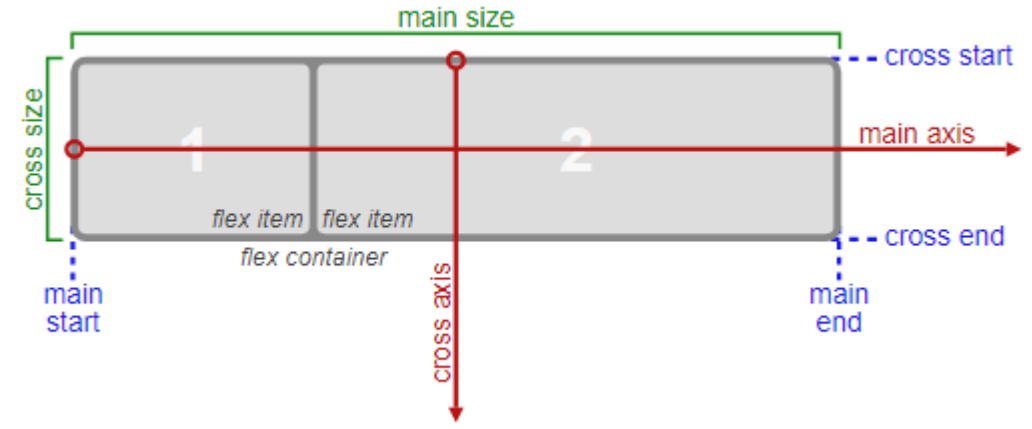
Flex elementen zullen de beschikbare ruimte binnen de container zo optimaal mogelijk benutten

Flexbox - container

Eigenschappen van een flex container:

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

Onderstaand tonen wij hoe een container met virtuele lijnen is opgebouwd. Handig om de onderstaande termen te herkennen:



Flexbox - container

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

Bekijk een DEMO op:

https://codepen.io/5hart/pen/MmYEJb

Flexbox - container

flex-wrap

- De flex container probeert de verschillende child elementen (horizontaal) op dezelfde regel te tonen.
- Hij probeert de betreffende elementen gelijkmatig te verkleinen totdat ze wél op de regel passen.

Wil je niet dat elementen verkleind worden óf juist buiten het beeld vallen? Gebruik dan de eigenschap **flex-wrap.**

Flexbox - container

flex-wrap

De mogelijke waarden die deze property kan hebben zijn:

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.

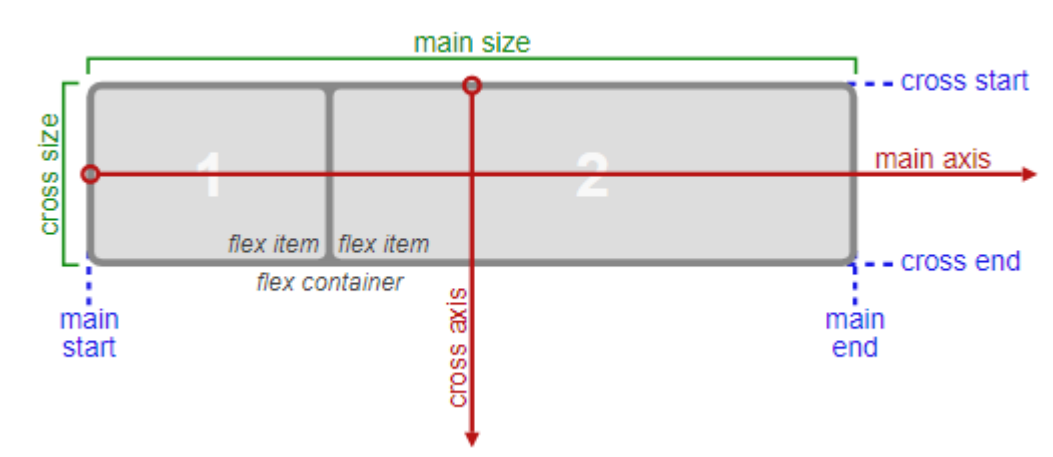
Let op dat de wrap waarden een gelijk effect kunnen hebben indien je je code op een device met een kleine viewport (klein zichtbaar scherm) bekijkt.

DEMO: https://codepen.io/5hart/pen/QvwaLB

Flexbox - container

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.

Denk aan de onderstaande afbeelding:



Flexbox - container

justify-content

Dit is de property die ons helpt met het uitlijnen van elementen binnen een 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.

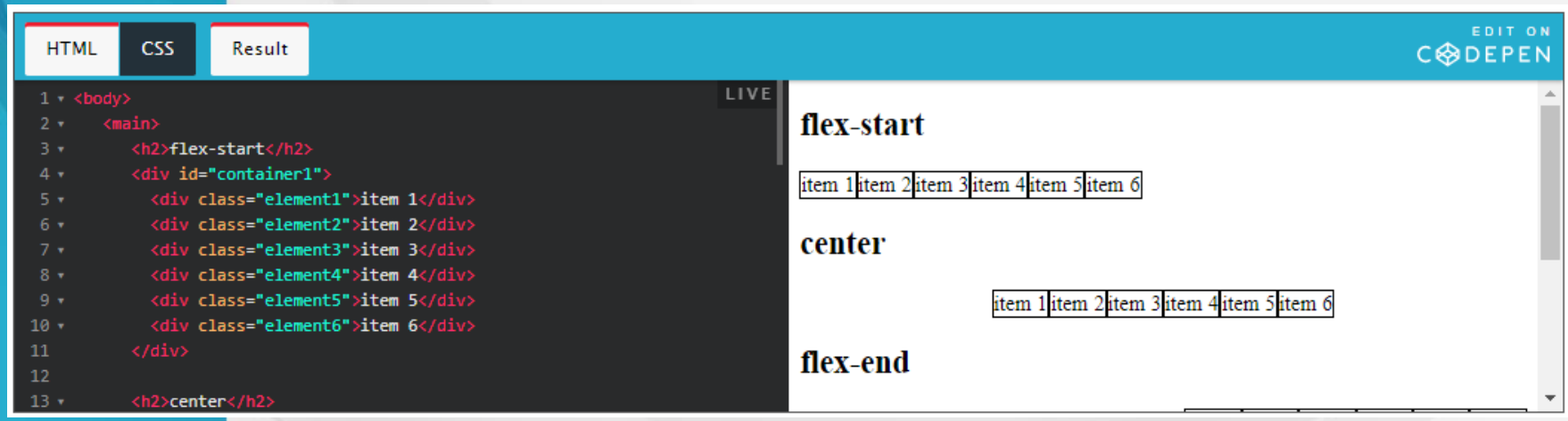
Flexbox - container

justify-content (vervolg)

Mogelijke waarden van de 'justify-content' eigenschap zijn:

flex-start, *center*, *flex-end*, *space-between* en *space-around*

https://codepen.io/5hart/pen/eWmrWO



Flexbox - container

align-items

- Deze property wordt gebruikt voor het verticaal uitlijnen van flex elementen

- 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.

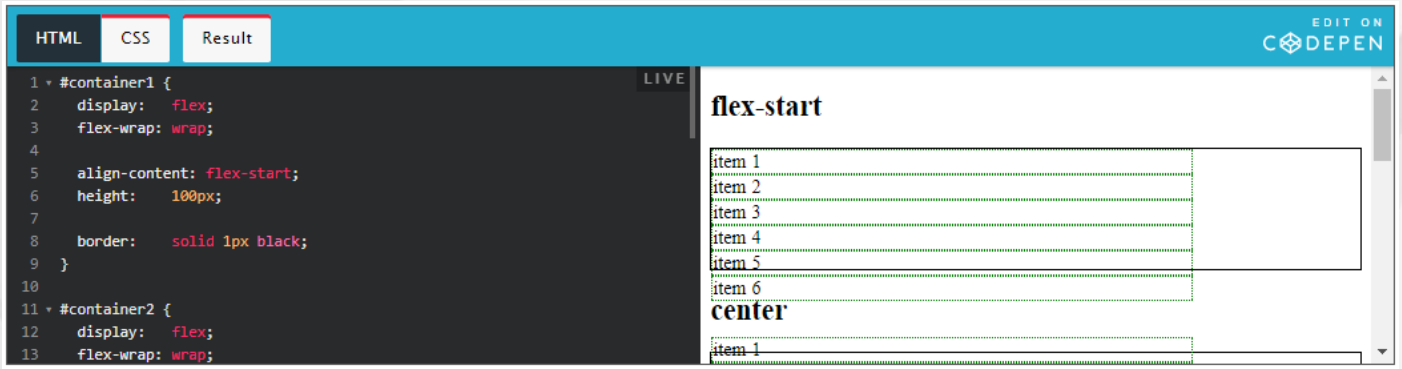
https://codepen.io/5hart/pen/EmjGZq

Flexbox - container

align-content

Voor het uitlijnen van de regels flex elementen indien ze op meer dan één regel staan, gebruiken wij de 'align-content' eigenschap

- De regels worden (verticaal gezien) binnen een container over de Y-as verspreid.
- Geldt voor elementen die meer dan één regel in beslag nemen
- Container dient een hoogte ingesteld te hebben, anders is er



https://codepen.io/5hart/pen/bWdzry

Flexbox - elementen

- 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.

Qua flex element properties gaan aan bod komen:

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

Flexbox - elementen

flex-basis

- Heeft als waarde een getal om aan te geven hoe hoog een element is in geval de flex-direction is ingesteld op 'column’
- Geeft aan hoe breed een element is in geval de flex-direction is ingesteld op 'row’.
- Kan ook de waarden 'auto’ of '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.

Flexbox - elementen

flex-shrink

- Is een deel of vermenigvuldig waarde van de verkleining van een de afmeting van een element in verhouding tot de andere elementen binnen de container.

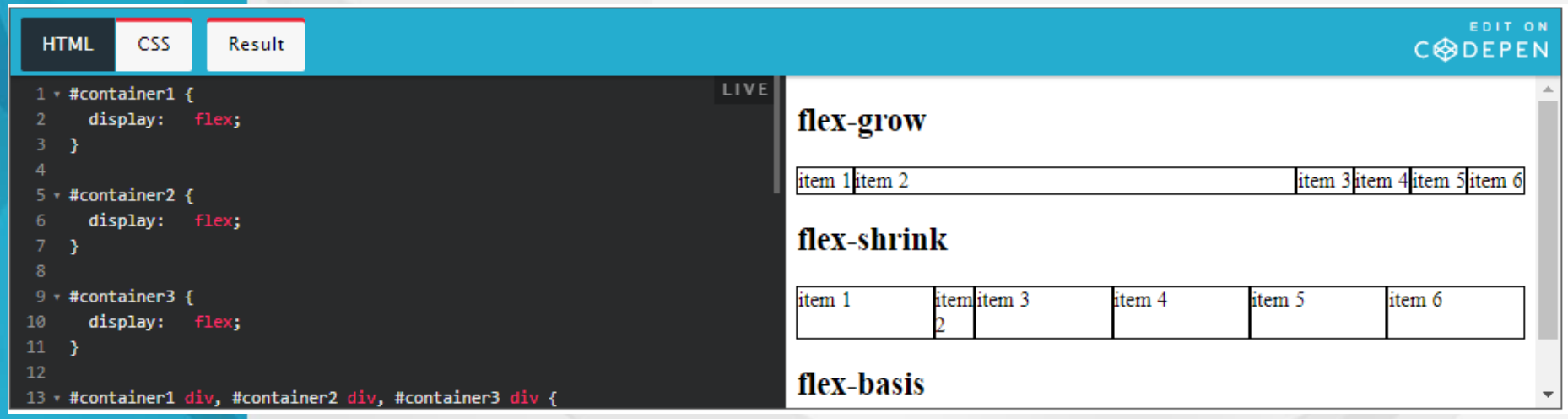
flex-grow

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

Flexbox - elementen

DEMO:

Bekijk de code op: https://codepen.io/5hart/pen/mmyXMo



Flexbox - elementen

flex

- 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.

align-self

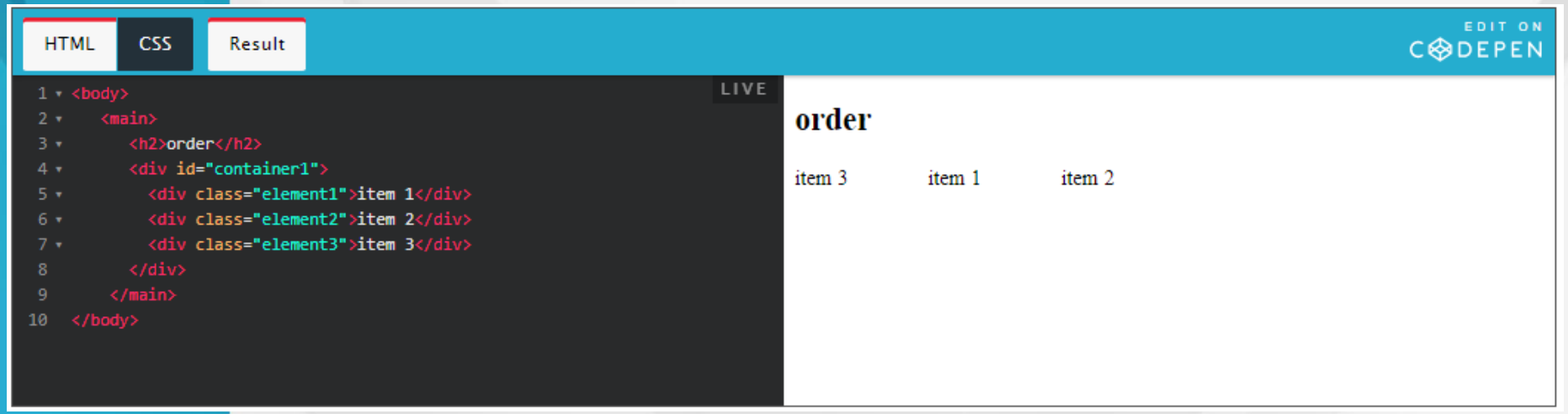
- Overrulen van de 'align-items' eigenschap van een container, enkel voor dit specifieke element.

Flexbox - elementen

order

- 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:



Grid

- De vijfde opmaak mogelijkheid naast tabellen, positionering met de 'position' eigenschap, floats en het flexbox systeem
- Goed toepasbaar voor verticale uitlijning en het creëren van tabel-achtige layouts
- Er zijn eigenschappen voor een container element en eigenschappen voor de child elementen (de zogeheten grid items)
- '**display**' property moet worden ingesteld op '**grid**'. Dit vertelt het systeem dat dit element de container wordt voor grid items. De waarden 'subgrid' en 'inline-grid' zijn ook mogelijk, maar vallen buiten de scope van de cursus.

Grid – container eigenschappen

- Een grid is opgebouwd uit kolommen en rijen.
- Wij geven aan hoeveel kolommen en rijen wij willen gebruiken, ieder met de door ons in te vullen breedte (of hoogte, voor rijen). Dit doen we middels de eigenschappen:
    - grid-template-columns
    - grid-template-rows
- Daarnaast kunnen wij iedere kolom en rij een naam geven om later naar terug te kunnen verwijzen.

- https://codepen.io/5hart/VboWQg/

Grid – element eigenschappen

- De individuele grid elementen kunnen wij een plaats geven binnen het grid, die lijkt op het werken met tabellen.

- Met behulp van de eigenschappen:
    - grid-column-start
    - grid-column-end
    - grid-row-start
    - grid-row-end

  Kunnen we onze elementen plaatsen (en e.d. spreiden) over onze grid.
- https://codepen.io/5hart/WjVEaz/

Bekijk de online cursus content voor meer informatie

H7 – Responsive Design

Leerdoelen

- Het mobile first concept kunnen toelichten
- Weten wat de viewport is
- Met touch events kunnen omgaan
- Breakpoints met behulp van media queries kunnen afhandelen
- Afbeeldingen responsive weergeven

H7 – Responsive Design

1 Inleiding

2 Viewport, pixel density en relatieve afmetingen

3 Touch events

4 Breakpoints en media queries

5 Responsive tables en fonts

6 Responsive images

Inleiding

Mobile first ontwerp principe

- Weinig ruimte op een mobiel device (soms zelfs >= 200 pixels br)
- Zorgt voor filteren van minder belangrijke content.
- Zorgt weer voor beter semantische organisatie van pagina of form
- Vanuit performance oogpunt: loopt het goed op een mobiel device, dan ook op een desktop omgeving.

DEMO ‘Device modus’ @ developer tools

Viewport

- 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!
- Het instellen van de viewport voor op je webpagina kan middels een <meta> tag:

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

- 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 dus 1.0 (geen zoom).

Pixel density

- Hoe breed is (bijv) 200 pixels op het scherm daadwerkelijk?
- Heeft alles te maken met Density Independant Pixels (DIPS).
- Een device heeft altijd een fysiek aantal pixels (Hardware Pixels).
- Om content op verschillende devices mooi weer te geven, redeneert de browser in DIPS. Dat is vaak een veelvoud of deling van de Hardware Pixels.
- **Als developer kun je niets doen met DIPS!**
- DIPS = Hardware Pixels / Device Pixel Ratio
- Device Pixel Ratio = uit device te lezen
- **Les: Houd relatieve maatvormen aan, niet pixels. Dus o.a. vw en vh.**

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.

Touch events

- Al lange tijd wordt niet meer enkel met keyboard en/of muis door webpagina's genavigeerd.
- Touch events, ofwel aanraak momenten, nemen een niet te onderschatten grote rol in bij het navigeren van, naar en binnen webpagina's.
- Een klik met de muis niet hetzelfde is als een druk met de vinger.

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. Een marge van 5px (dus 45x45) is ook prima.

Relatieve afmetingen

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.

Touch events

Opdracht:

- 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.

Break points

- Om een pagina op zo veel mogelijk devices goed weer te geven, kunnen we gebruik maken van break points en media queries.
- Op goede responsive webpagina's wordt de opmaak vaaj aangepast aan de hand van het scherm formaat. Ieder omslagpunt van scherm formaat is voor de webpagina als break point in te stellen.
- Bij een breakpoint wordt vaak gekeken naar de pagina breedte.
- Twee of drie breakpoints is het meest gangbaar te gebruiken.
- Media queries zijn veelal het middel om breakpoints mee te configureren.

Break points

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!

Media queries

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; minst snelle variant; moet de eerste regel in <style> blok zijn.
- 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

```
@media screen and (min-width: 800px)
/* browser breedte van minimaal 800px */
@media screen and (min-width: 400px) and (max-width: 600px)
/* browser breedte van minimaal 400px en maximaal 600px */
@media screen and (max-width: 400px), (min-width: 1000px)
/* browser breedte van maximaal 400px OF minimaal 1000px */
```

DEMO:
https://codepen.io/5hart/WjQdyy

Responsive tables

Het weergeven van de meest nuttige overzichtelijke informatie met behulp van tabellen, doen we vooral door:

- Duidelijke kolom koppen. Goed uitdenken en stijlen met CSS
- Niet teveel kolommen. Kolommen kunnen zelfs buiten de viewport en het zicht vallen
- Niet teveel rijen. Rijen beperken doen we vaak met JavaScript code of een server programmeertaal. Denk aan het pagineren van het resultaat.

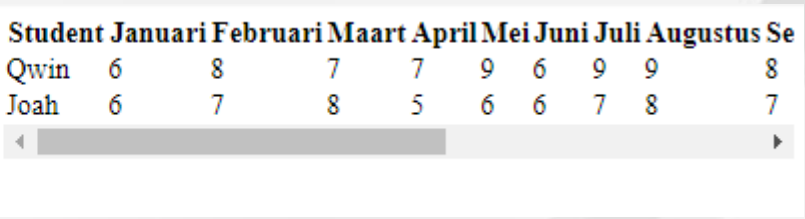
Responsive tables

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.

DEMO:
https://codepen.io/5hart/jmbYgX (verklein het scherm...)

```
/* Géén media query, resultaat middels overflow is scrollbalk */
#tabel-wrapper {
  overflow-x: auto;
  width:      100%;
}
```



... niet een erg mooie oplossing, wél een eenvoudige.

Responsive tables

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. Dit kost wel duidelijk méér werk:

- 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);

DEMO:
https://codepen.io/5hart/EmVQxV (verklein het scherm...)

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.

- Geadviseerde aantal karakters per regel: tussen de 45 en 90.
- Gemiddelde van ongeveer 60 is goed aan te houden.
- 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.

Responsive images

Bij het verbeteren van de performance van onze website zijn deze twee zaken minimaal van belang bij afbeeldingen:

- 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.

Gebruik voor teksten waar mogelijk altijd HTML + CSS; geen afbeeldingen, 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

Alternatief: Icon fonts (matige ondersteuning)

Responsive images

Filetype keuze

Let op / overweeg de volgende zes aspecten:

- Compressie
- Zwart wit of kleur
- 8 of 16 bits kleur
- CMYK of LAB mogelijkheid
- Transparantie
- Animatie mogelijkheid

http://socialcompare.com/en/comparison/image-file-formats

Responsive images

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.

Responsive images

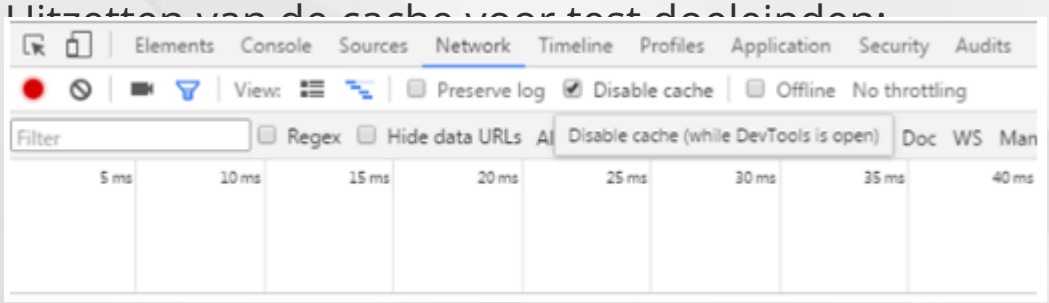
Handige tool voor het veranderen van bestand formaat: ImageMagick

Handige tool voor het veranderen van de file omvang: ImageOptim

Caching

- Soms worden afbeeldingen in de lokale cache opgeslagen.
- Dit kan een performance voordeel opleveren
- Kan er ook voor zorgen dat de data (bijv. de afbeelding) niet actueel is.

Uitzetten van de cache voor test doeleinden:



Responsive images - scrset

Srcset attribuut

Omdat het altijd stoeien is met de afmetingen van een afbeelding kunnen wij middels het *srcset* attribuut:

- Verschillende afbeelding-URL's opgeven voor één en dezelfde afbeelding.
- De browser kiest de juiste afbeelding aan de hand van de viewport van het device.
- Deze (voor nu nog even –alle-) afbeelding(en) worden gedownload.

Responsive images - srcset

Voorbeeld:

- De meeste, grotere desktop schermen hebben een **1x** dichtheid.
- Mobiele devices hebben regelmatig **2x** of zelfs 3x dichtheid.
- We kunnen met een commando in onze Dev Tools zien welke pixel density ons device heeft:

Opdracht:

Ga binnen de Chrome Developer Tools naar het tabblad ‘Console’ en voer daar in: window.devicePixelRatio

Responsive images - srcset

- Het vorige voorbeeld zorgt ervoor dat de browser de juiste afbeelding toont.
- Echter wordt nog steeds iedere versie van de afbeelding gedownload.
- 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?

Responsive images - srcset

Met gebruik van de ‘w’ grootte indicator!

- Wij vertellen 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.
- We gebruiken de ‘w’ breedte indicator in plaats van 1x / 2x / 3x.

Voorbeeld:

Responsive images – Sizes attribuut

Probleem:

De browser weet nu welke afbeelding hij moet downloaden op basis van de ‘w’ indicator, maar nog niet hoe groot de afbeelding op de pagina getoond zal worden met CSS gezien HTML code eerst wordt verwerkt.

Met het **sizes attribuut** geven wij aan:

- Hoe groot de afbeelding wordt weergegeven met CSS
- Sterk aanbevolen deze afmeting te laten matchen met je CSS maat

Responsive images – Sized attribuut

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” />
```

Responsive images – Picture element

We kunnen alternatieve bestandformaten opgeven in HTML middels het picture element:

```
<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>
```

- Wordt van boven naar beneden doorlopen.
- De eerst bruikbare afbeelding wordt gekozen.
- Wordt ingevuld in het IMG element.
- De default afbeelding staat bij ‘src’ in het img element.

Responsive images – Picture element

In het volgende voorbeeld gaan we met media queries de juiste file uitkiezen, vervolgens het filetype. HTML = sneller dan CSS.

HTML Result EDIT ON CODEPEN

```
<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" />
```

https://codepen.io/5hart/qmerQd

Responsive images – Inline images

Wat ook laadtijd kan besparen is het gebruik van een datastream in het src attribuut van een img element.

- We vullen als URL geen absoluut of relatief pad in, maar een verwijzing naar een plaatje middels een zogeheten datastream.
- Image file wordt hierdoor rechtstreeks in de webpagina geladen
- Met behulp van tools is een plaatje om te zetten naar een datastream.
- Laadtijd verschil hangt af van omvang afbeelding

DEMO: https://codepen.io/5hart/aWvpNY

H8 – Frameworks overview

Leerdoelen

- Kennisnemen van verschillende CSS (en JavaScript) frameworks

H8 – Frameworks overview

Inleiding

- Het gebruik van een framework is binnen de IT is common 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.

Inleiding

- 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.
-
- Wij kijken vooral naar front-end frameworks. Dat wilt zeggen; frameworks die de toepassing vinden op het HTML, CSS en JavaScript vlak.

Inleiding

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

Frameworks

Er is een continue ontwikkeling van nieuwe frameworks

Onderstaand een aantal veel gebruikte:

- Normalize.css
- Modernizer
- Bootstrap
- Bulma
- Semantic UI
- jQuery
- Foundation
- Angular
- LESS & SASS CSS pre-processors
- ReactJS
- NodeJS
- ……
  ……

Einde