

INTERACTIVE MEDIA AND COMMUNICATION_

Workshop 3 - Opdracht 2

CSS & Kleur

Frans van Hofwegen

20 augustus 2021

CSS & KLEUR

In deze oefening gaan we spelen met verschillende manier om kleuren te gebruiken!

1 ID'S VERSUS CLASSES

Inmiddels weet je dat je *HTML-tags* kunt voorzien van *attributes*. De *id*- en *class*- attributes zijn onmisbaar om complexe vormgeving en lay-outs te maken.

1.1 ID

Id staat voor *identifier* en wordt gebruikt om één element aan te spreken. Vergelijk de id met jouw eigen naam. Een ID is **altijd** uniek en mag daarom per HTML-document één keer voorkomen.

```
<!-- HTML code: -->
<article id="studeertips">
  <h1>Handige studeertips</h1>
  <p>In dit artikel bespreken we handige studeertips.</p>
</article>

/* CSS-code: */
#studeertips {
  padding: 20px;
}
```

1.2 Class

De tegenhanger van de *id* is de *class*. Je kunt hiermee een hele groep HTML-elementen aanspreken. Dit is te vergelijken met de HAN, die met een klassencode alle studenten uit een bepaalde klas tegelijk kan bereiken. Een class mag worden toegekend aan meerdere elementen:

```
<!-- HTML code: -->
<article class="tips">
  <h1>Studeertips</h1>
  ...
</article>

<article class="tips">
  <h1>Stagetips</h1>
  ...
</article>

/* CSS-code om alle h1-kopteksten in alle .tips-elementen aan te spreken: */
.tips h1{
  font-size: 20px;
}
```

1.3 Prioriteit

Onthoud dat een *class* het aflegt tegenover een *id*, vanwege de *prioriteit*-regel (zie vorige opdracht).

Kun jij het resultaat van onderstaande code voorspellen zonder het te proberen in de browser?

```
<!-- HTML code: -->
<article id="studeertips" class="tips">
  <h1>Studeertips</h1>
  ...
</article>
<article class="tips">
  <h1>Stagetips</h1>
  ...
</article>

/* CSS-code om alle h1-kopteksten in alle .tips aan te spreken: */
#studeertips h1{
  font-size: 25px;
}

.tips h1{
  font-size: 20px;
}
```

1.3.1 Complexe selectors en *combinators*

Complexe *HTML-code* vereist ook complexe *selectors*. Bestudeer:

- “Eenvoudige” *selectors*
(een verraderlijke naam, want eenvoudige selectors kunnen echt wel heel complex worden)
zie https://www.w3schools.com/Css/css_selectors.asp
- de *CSS-combinator*
(https://www.w3schools.com/Css/css_combinators.asp).

Begrijp jij het verschil tussen onderstaande *selectors*?

```
#studeertips.tips{
  font-size: 30px;
}

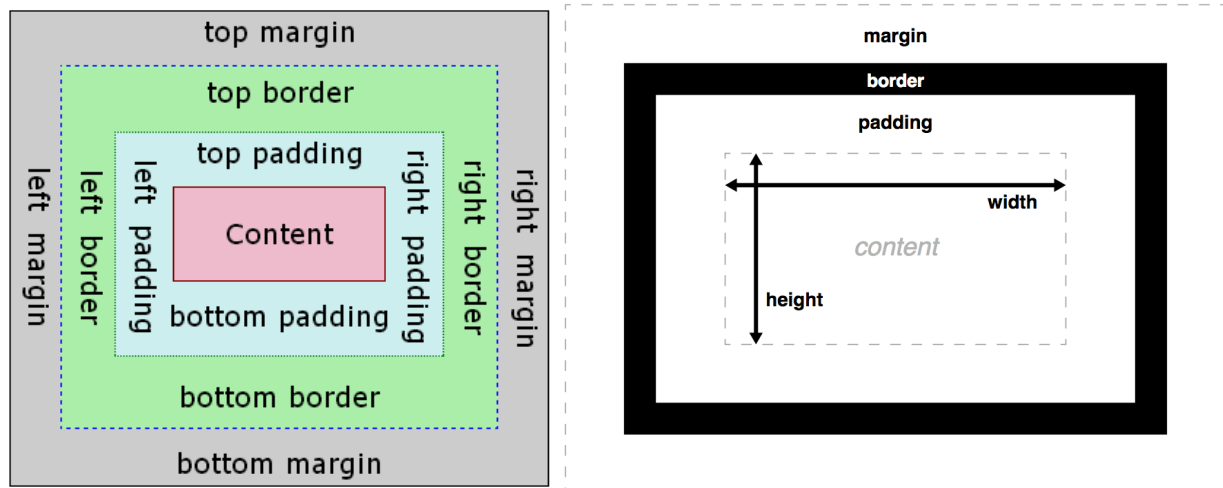
#studeertips .tips{
  font-size: 30px;
}

#studeertips, .tips{
  font-size: 30px;
}
```

Zo ja, dan begrijp je *selectors* en *combinators* voldoende en mag je doorgaan. Anders moet je de oefeningen nogmaals doen, meer lezen op [w3schools.com](https://www.w3schools.com) of je kunt natuurlijk ook hulp vragen aan het MediaLab of één van de docenten.

2 BOX MODEL

Ieder *HTML-element* is als het ware een soort “doosje”. Dit principe wordt de *CSS box model* genoemd. Het begrijpen en kunnen toepassen van de *box model* is een onmisbare vaardigheid.



2.1 Content

De inhoud waar tekst, afbeeldingen of andere *HTML-elementen* instaan. *Content* heeft *width* en *height* eigenschappen.

2.2 Padding

De ruimte om *content* heen. *Padding* krijgt altijd dezelfde kleur als de *background-color* van *content*.

2.3 Border

De rand om de *padding*. *Borders* kunnen een eigen dikte, stijl, kleur en/of afbeelding krijgen.

2.4 Margin

De ruimte aan de buitenkant van de *border*. De *margin* is transparant en krijgt de kleur van het onderliggende (parent) *element*.

2.4.1 Margin Collapse

Wanneer twee marges van twee aangrenzende elementen tegen elkaar komen te liggen, dan zal de kleinste marge opgaan in de grootste. Dit principe noemen we *margin collapse*. De *margin collapse* is soms niet wenselijk en bepaalt in sommige gevallen dus of je een element *padding* of juist *margin* geeft.

2.5 Afmetingen van de box model berekenen

Bereken de ruimte die een *HTML-element* in beslag neemt, door onderstaande onderdelen van de *box model* bij elkaar op te tellen:

- De width van een element = width + padding-left + padding-right + border-left + border-right. Hou daarnaast ook nog rekening met margin-left en margin-right.
- De height van een element = height + padding-top + padding-bottom + border-top + border-bottom. Hou daarnaast ook nog rekening met margin-top en margin-bottom.

2.5.1 Box-sizing: border-box

Bij het maken van complexe *Responsive Web Design* lay-outs is het soms (lees: vaak) wenselijk dat de width van een element wordt berekend **inclusief** padding en border. Gebruik hiervoor de handige *border-box value* van de *box-sizing property*.

In onderstaand voorbeeld blijft de totaalbreedte van het hele *element*, inclusief padding en border, 100% van de *parent-element* (en dus niet 100% + 10px + 10px + 2px + 2px). Dit komt dankzij de waarde van de *box-sizing property*:

```
article{
  width: 100%;
  padding: 10px;
  border: 2px;
  box-sizing: border-box;
}
```

3 OPDRACHT

Maak de volgende afbeelding zo goed mogelijk na (op de volgende pagina staan meer instructies):

Kleuren in CSS

In CSS kun je op verschillende manieren kleur geven aan tekst. Hieronder worden er vier manier besproken.

Kleurnaam

Er zijn 147 kleurnamen gedefinieerd. Veel mensen vinden dit een te beperkt palet, en het is moeilijk om de naam voor elk van de kleuren te onthouden, zodat ze (met uitzondering van 'white' en 'black') niet vaak gebruikt worden.

Voorbeelden:

- `color: red`
- `color: darkgreen`
- `color: mediumaquamarine`

Hexadecimale code

Dit zijn zescijferige codes die de hoeveelheid rood, groen, en blauw (RGB) in een kleur vertegenwoordigen, voorafgegaan door een hekje (#).

Hoeveelheid van een kleur wordt aangegeven in het hexadecimale stelsel. Dit is een 16-tallig stelsel waarmee je van 0 tot F kunt tellen. Hoe hoger het 'getal', hoe meer van een kleur.

Voorbeelden (herken je het patroon?):

- `color: #ff0000;` [rood]
- `color: #00ff00;` [groen]
- `color: #0000ff;` [blauw]
- `color: #000000;` [zwart]
- `color: #ffffff;` [wit]

RGB-waarde

Net als bij de hexadecimale code druk je hiermee de kleur ook uit in hoeveelheden van rood, groen, en blauw. Alleen gebruik je bij RGB-waarden decimale getallen die van 0 tot 255 lopen.

Voorbeelden (probeer de kleur aan de code te herkennen, en vul de kleuren hieronder in):

- `color: rgb(255, 0, 0);` [zet hier de kleur]
- `color: rgb(0, 255, 0);` [zet hier de kleur]
- `color: rgb(150, 0, 255);` [zet hier de kleur]
- `color: rgb(100, 100, 100);` [zet hier de kleur]

Hue, saturation, luminosity (HSL)

Een laatste manier om kleur aan te geven is via HSL-waarden. Lees meer over deze manier op de websites van [W3Schools](#) en [The New Code](#).

Voorbeelden:

- `color: hsl(0, 100%, 51%)` [zet hier de kleur]
- `color: hsl(212, 100%, 50%)` [zet hier de kleur]
- `color: hsl(212, 0%, 100%)` [zet hier de kleur]
- `color: hsl(212, 0%, 0%)` [zet hier de kleur]

1. Gebruik het aangeleverde HTML-document als startpunt.
2. Maak zelf een extern CSS-bestand en koppel dat aan het HTML-document.
3. Vergeet niet om daarnaast ook een CSS-reset toe te passen.
4. De kleuren die in het voorbeeld worden gebruikt zijn: voor de h1-tag “mediumaquamarine”, voor de h2-tags: “rgb(105, 2, 149)” en voor een aantal van de p-tags is dat: “#777777”.
5. De CSS-reset zal natuurlijk de diverse paddings en margins resetten naar 0. Schrijf daarom ook die benodigde CSS-code om het voorbeeld van de vorige pagina zo goed mogelijk te benaderen. Experimenteer en krijg zo de basisprincipes van de *CSS box model* onder de knie!

Gelukt? Check of je het begrepen hebt. Is de HTML-code overzichtelijk geschreven voor een ander? Staan de CSS-bestanden in een eigen mapje? Zijn de links naar de CSS-bestanden juist toegevoegd?

Vergeet niet je code te valideren! Kopieer je code van deze opdracht. Plak de code in het formulier op https://validator.w3.org/#validate_by_input en check of er errors of warnings tevoorschijn komen. Los in ieder geval altijd de *errors* op! Check nu ook de CSS-code met de CSS-validator op: https://jigsaw.w3.org/css-validator/#validate_by_input en los de errors op!

OPEN UP
NEW HAN_ UNIVERSITY
HORIZONS. OF APPLIED SCIENCES