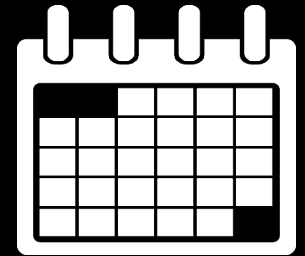


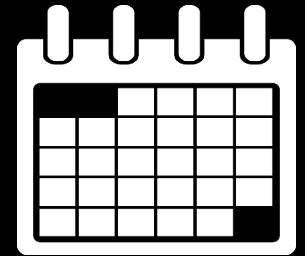
User Interfaces 1

CSS

Basis



1. De taal CSS
2. CSS verwerking
3. Waarden en eenheden
4. Elementen benaderen met selectors



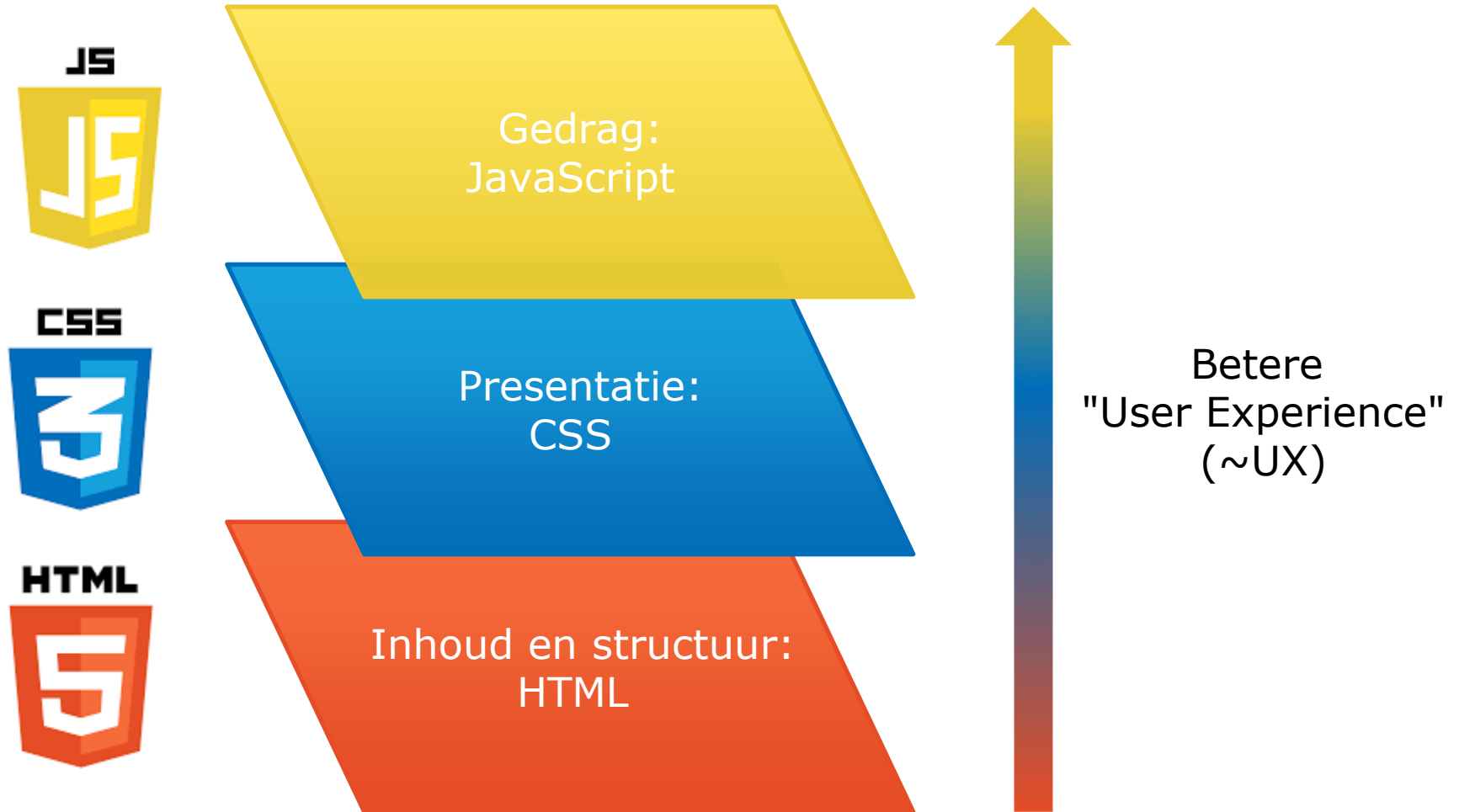
1. De taal CSS
2. CSS verwerking
3. Waarden en eenheden
4. Elementen benaderen met selectors

De taal CSS (cascading style sheets)

- CSS is een taal om aan te geven hoe de styling en de lay-out van je webpagina moet gebeuren
- Een CSS-bestand is een tekstbestand met de extensie .css
- Demonstratie van de mogelijkheden:
<http://www.csszengarden.com/>
 - Surf naar de website
 - Selecteer rechts verschillende designs
 - Inhoud (HTML) is dezelfde,
design (CSS) is verschillend...



3 Lagen



Stijlregel (~rule)

- Een CSS document bestaat uit stijlregels (*rules*)
- Een stijlregel ziet er schematisch als volgt uit:

```
selector { eigenschap: waarde; }
```

In het Engels wordt dit:

```
selector { property: value; }
```

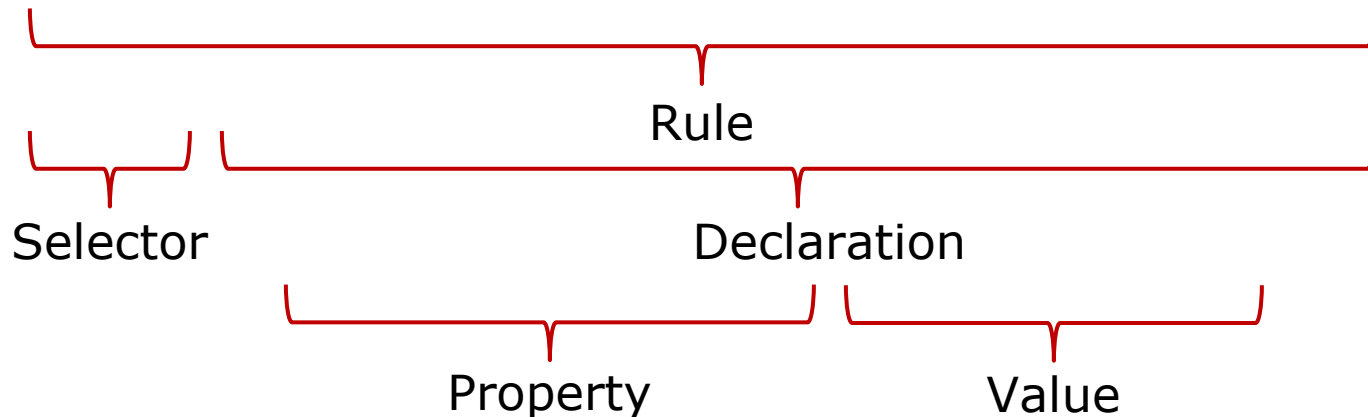
- In een stijlregel kan je meer dan 1 eigenschap/waarde combinatie plaatsen, de schrijfwijze wordt dan meestal:

```
selector {  
    property: value;  
    property: value;  
    ...  
}
```

Stijlregel (~rule)

- Om de inhoud van elke <h1> tag horizontaal te centreren gebruik je de volgende stijlregel:

```
h1 { text-align: center; }
```



- De naam van de eigenschap en de waarde worden gescheiden door een dubbelpunt ':'.
- Elke declaratie (= combinatie van eigenschap en waarde) sluit je af met een puntkomma ';'.

Voorbeeld: H8/01_basis

Commentaar in CSS

`/* Dit is een stukje commentaar */`

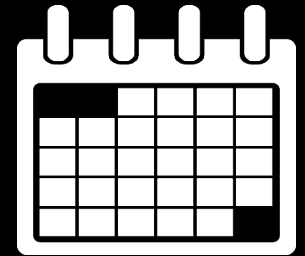


`/*` → start commentaar



`*/` → sluit commentaar af

- Zelfde opmerking als bij HTML:
 - Handig om code uit te leggen of stukjes uit te schakelen
 - Opgelet: iedereen kan het lezen!



1. De taal CSS
2. CSS verwerking
3. Waarden en eenheden
4. Elementen benaderen met selectors

CSS verwerking

Bij CSS zijn er 2 belangrijke concepten:

1. Inheritance
2. Cascading

Het is op basis van deze 2 concepten dat de uiteindelijke waarde voor een css-eigenschap van een element in het html-document bepaald wordt

Inheritance

Dit concept geeft aan hoe de waarde van een css-eigenschappen bepaald wordt indien er geen waarde gedefinieerd is.

Alle css-eigenschappen zijn onderverdeeld in 2 categorieën:

- Inherited
 - erven de waarde over van het ouderelement
- Non-inherited
 - krijgen een standaard/default waarde

Cascading

De waarde van een css-eigenschap kan meerdere malen en op verschillende plaatsen (~bronnen) gedefinieerd worden:

1. Extern
 - bestand:css
 - wordt naar verwezen adhv van het element 'link'
 - in het element 'head' van het html-document
2. Intern
 - element/tag: <style>...</style>
 - in het element 'head' van het html-document
3. Inline
 - attribuut: style="..."
 - is een globaal html-attribuut

Bij 'extern' en 'intern' is de volgorde waarin deze voorkomen in het element 'head' belangrijk voor het samenvoegen.

De **laatst gedefinieerde waarde voor een css-eigenschap**, volgens deze trapsgewijze samenvoeging, wordt de feitelijk toegepaste waarde!

Opgelet: voor de bronnen 'extern' en 'intern' geldt ook nog dat **enkel de selectors met de hoogste 'specificiteit' waarin de css-eigenschap gedefinieerd is** van toepassing zijn (zie verder).

Bronnen: extern

Bij elke html pagina, of er nu een stijlblad aan gelinkt is of niet, wordt standaard eerst het *default stylesheet* van de browser toegepast. Hierin is voor elk element een standaard rule gedefinieerd en zorgt dus voor basis weergave in je browser.

In bijna alle gevallen gebruik je een eigen CSS-bestand (extensie .css). Daarin kan je de standaard rule *overschrijven*. De koppeling met de bijhorende HTML pagina gaat via de `<link>` tag in de `<head>` sectie.

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

VOORBEELD



css bestand (./stijlen/basis.css)

```
h1 {  
  color: navy;  
}  
  
h2 {  
  color: orangered;  
}  
  
article {  
  color: green;  
}
```

html bestand

```
<!DOCTYPE html>  
<html>  
  <head lang="nl">  
    <meta charset="UTF-8">  
    <title>Demo link tag</title>  
    <link href="stijlen/basis.css" rel="stylesheet">  
  </head>  
  <body>  
    <h1>Kop level 1</h1>  
    <h2>Kop level 2</h2>  
    <article>  
      <header>Header</header>  
      <p>Eerste alinea</p>  
    </article>  
  </body>  
</html>
```

Door de CSS in een apart bestand te stoppen hebben we een perfecte scheiding CSS/HTML, dit biedt grote voordelen wat betreft het onderhoud van de code.

Voorbeeld: H8/03_link

Normalize.css


<https://necolas.github.io/normalize.css/>

Nevenverschijnselen
door default stylesheets
van browsers?
→ reset CSS

**waarom plaats je deze
code vóór je eigen CSS?**

source:

<https://github.com/necolas/normalize.css/>



Normalize.css


**A modern, HTML5-ready alternative
to CSS resets**

Normalize.css makes browsers render all elements more consistently and in line with modern standards. It precisely targets only the styles that need normalizing.

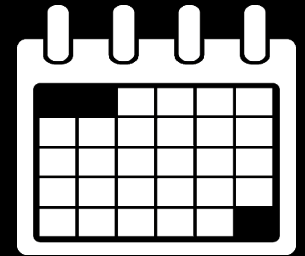
Download v8.0.1

Chrome, Edge, Firefox ESR+, IE 10+, Safari 8+, Opera
[See the CHANGELOG](#)

```
npm install normalize.css
```

 [Tweet](#)

[Read more about normalize.css »](#)



1. De taal CSS
2. CSS verwerking
3. Waarden en eenheden
4. Elementen benaderen met selectors

Waarden en eenheden

Elke eigenschap heeft een aantal mogelijke waarden:

- sleutelwoorden zoals auto, top, left enz
- een **getal**, al dan niet negatief of met een decimaal punt
- een **lengte** (een getal met direct daarna een eenheid)
- een **percentage** (een getal van 0 tot en met 100 direct gevolgd door een %-teken)
- een **URL** (een bestandslocatie)
- een **kleur** (een kleurnaam of een numerieke waarde)

Waarden en eenheden

- Lengte wordt gebruikt voor onder meer een marge, een rand, de breedte van een blok, ...
- Ook de grootte van een tekst is een lengte
- Een lengte bestaat uit een getal direct gevolgd door een eenheid
- Een lengte-eenheid is **relatief** of **absoluut**
- De enige zinvolle *absolute* eenheid is de pixel (komt overeen met 1 pixel op een beeldscherm)
 - in css de afkorting px

```
h1 { font-size: 48px; }
```

Waarden en eenheden

Een andere, voor tekst betere optie, is de *relatieve* waarde em.

```
h1 { font-size: 2em; }
```

De grootte van een em wordt bepaald door de lettergrootte (`~font-size`) van het **(ouder)element**.

Standaard hebben `<html>` en `<body>` in de meeste browsers een font-size van 16 pixels.



<http://webdesign.about.com/od/typemeasurements/qt/how-big-is-an-em.html>

Waarden en eenheden

Een nieuwe eenheid is de relatieve waarde rem.

```
h1 { font-size: 2rem; }
```

De grootte van een rem wordt bepaald door de grootte van de font-size het **root-element** <html>. Dit staat in de meeste browsers ingesteld op een grootte van 16 pixels.

```
article {  
    font-size: 2em;  
}  
  
h1 {  
    font-size: 1.5em;  
}  
  
p {  
    font-size: 1.5rem;  
}
```

Voorbeeld: H8/02_voorbeeld_rem

h1 en p zitten beide in het article element. Dus de lettergrootte in pixels van de kop is $2 \times 1,5 = 3\text{em}$ (of 48px) en de lettergrootte van de alinea (p-tag) is 1,5rem (of 24px)

Waarden en eenheden



Een andere relatieve waarde is het percentage.

```
h1 { font-size: 200%; }
```

De grootte van het element wordt bepaald door de grootte van het **ouderelement**.

```
body {  
    width: 960px;  
}  
  
h1 {  
    width: 50%;  
}
```

Voorbeeld: H8/02_waarden

h1 zit in het body element. Dus de breedte van het h1 element is 50% van 960 pixels, of 480 pixels.

Waarden en eenheden



Voorbeeld:

```
h1 {  
  border: solid blue;  
  height: 2em;  
}  
  
h2 {  
  border: solid blue;  
  width: 300px;  
}  
  
p {  
  border: solid blue;  
  height: 2em;  
  width: 50%;  
}
```



2x standaardhoogte

Helpt breedte browservenster

Voorbeeld: H8/02_voorbeeld

Waarden en eenheden

Andere nieuwe eenheden zijn gebaseerd op de **viewport** (= browservenster). De indeling van de pagina, de grootte van de tekst en andere inhoud, kan worden gekoppeld aan de grootte van het venster. Wordt het venster kleiner, dan schaalt de inhoud mee.

vw relatief tot 1% van de vensterbreedte

vh relatief tot 1% van de vensterhoogte

vmin relatief tot 1% van de kleinste afmeting

vmax relatief tot 1% van de grootste afmeting

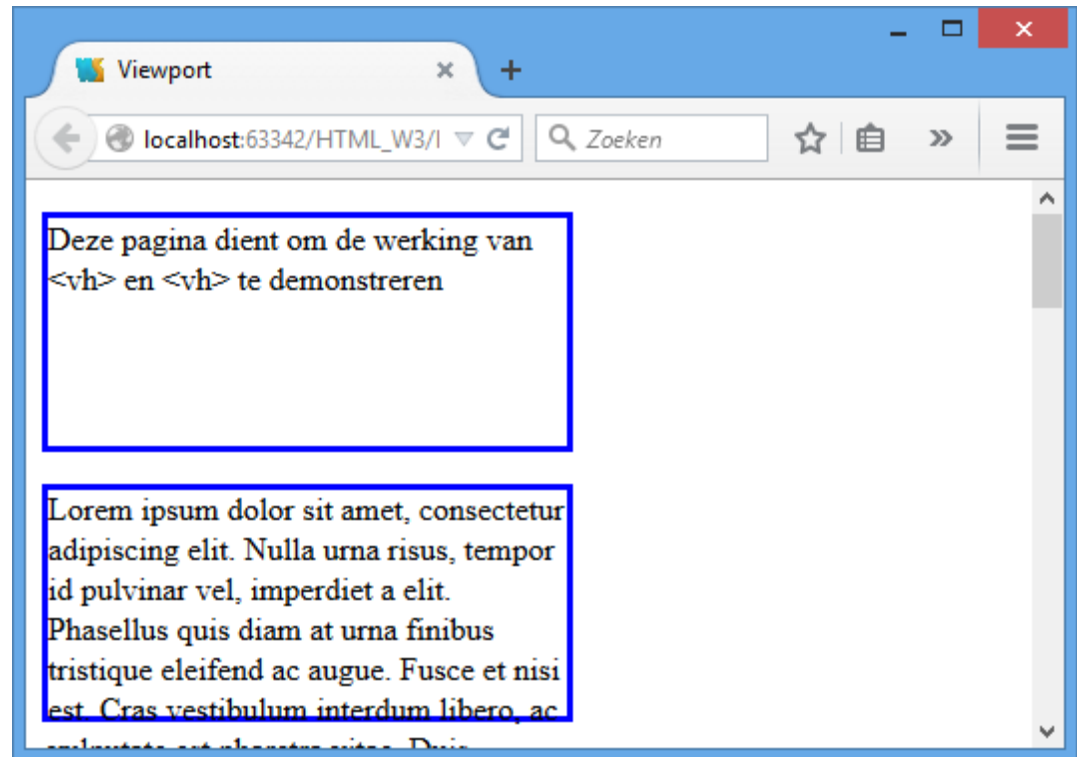
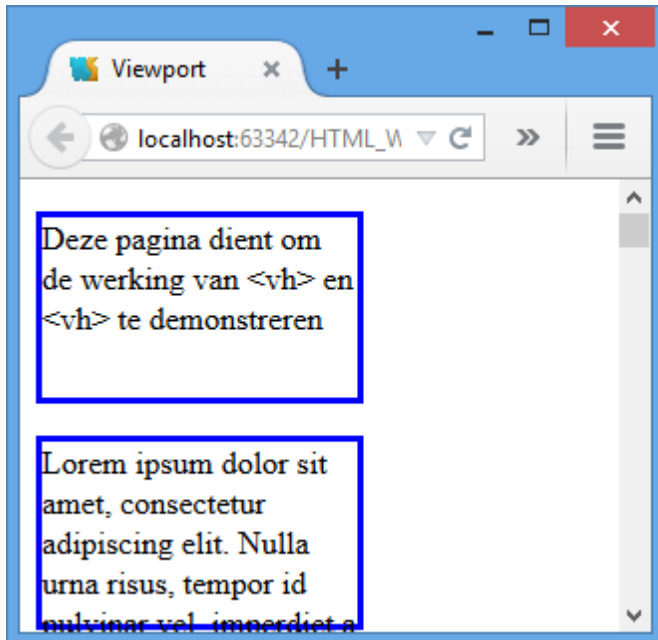
Just to clarify: 1vmax equals 1vh in portrait mode, whilst in landscape mode, 1vmax will equal 1vw

Waarden en eenheden

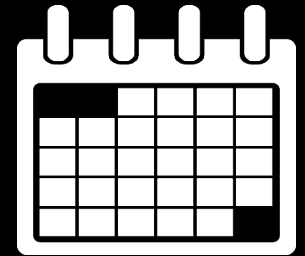


Voorbeeld:

```
p {  
    border: solid blue;  
    width: 50vw;  
    height: 40vh;  
}
```



Voorbeeld: H8/02_voorbeeld_viewport



1. De taal CSS
2. CSS verwerking
3. Waarden en eenheden
4. Elementen benaderen met selectors

Selectors

Er zijn verschillende typen van selectors, verdeeld in de groepen:

- de universele selector
- de typeselector
- de klasse-selector
- de id-selector
- de attribuutselectors
- de pseudoklassen
- de pseudo-elementen

Selectors: 'specificiteit'

Of een css-eigenschap gedefinieerd in een stijlregel in aanmerking komt bij het bepalen van de waarde bij de trapsgewijze css verwerking hangt af van de **specificiteit** van de selector waarin de css-eigenschap is gedefinieerd.

Dit is een waarde die wordt bepaald door het tellen van de onderdelen in de selector volgens volgende categorisering:

A	B	C
id	(pseudo-)klasse, attribuut	(pseudo-)element

Vb.:

```
<nav id="topmenu">
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</nav>
```

```
#topmenu li { /*specificiteit: 1-0-1*/
  color:green; }

ul > li { /*specificiteit: 0-0-2*/
  font-size:1.3em; color:blue; }
```

=> opmaak die op de 'li' elementen toegepast zal worden: **font-size:1.3em** en **color:green**

De universele selector *

In zijn eenvoudigste vorm selecteer je er alle elementen van de pagina mee.

```
* { color: blue; }
```

Alle tekst krijgt een blauwe kleur.

Je kunt er ook alle afstammelingen (*descendants*) mee aanduiden.

```
aside * { border: 1px solid red; }
```

Alles wat in de html pagina onder een `<aside>` ligt krijgt een rode boord.

Voorbeeld: [H8/04_universele_selector](#)

De typeselector

Ook **elementselector** genoemd.

```
h1 { background-color: burlywood; }
```

Alle koppen van het type h1 krijgen de achtergrondkleur burlywood.

```
aside ul li { color: red; }
```

Alle li elementen die onder een ul in een aside liggen worden in het rood weergegeven.

Voorbeeld: [H8/04_type_selector](#)

De klasse- en id-selector

De **klasse-selector** krijgt vóór de naam van elke klasse een punt.

```
.kop { background-color: burlywood; }
```

Elke tag waar `class="kop"` bijstaat krijgt als achtergrondkleur `burlywood`.
Meerdere elementen kunnen dus dezelfde klasse hebben!

De **id-selector** krijgt vóór de naam van de enige id van de webpagina een #-teken.

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

De tekst in de enige tag waar `id="belangrijk"` bijstaat wordt in het rood weergegeven.
Bij valide html kan dus maar één element dat id hebben!

Voorbeeld: [H8/04_class_id_selector](#)

De attribuut-selector

Hiermee selecteer je elementen die een bepaald attribuut hebben of waarvan een attribuut een bepaalde waarde heeft.

```
img[width] { border: double 6px; }
```

Elke img tag waarbij een width attribuut staat krijgt een dubbele zwarte boord met een breedte van 6 pixels

```
img[width="200px"] { border: solid 6px red; }
```

Elke img tag waarbij een width attribuut staat met een exacte waarde van 200 pixels krijgt een rode boord met een breedte van 6 pixels

Combinaties

```
1 article
2     section
3         section
4         section
5     section
6 section
7 section
```

Mogelijkheden:

- **Komma** (,): scheidingsteken tussen selectors = opsomming
`article, section` → alle articles en alle sections = lijn 1 tot 7
- **Spatie** (): alle onderliggende elementen
`article section` → alle sections met article ergens als voorouder = 2 tot 5
- **Punthaak** (>): descendant combinator, moet kind zijn
`article > section` → alle sections met article als parent = 2 en 5
- **Plusteken** (+): zelfde parent, **eerste match** onder zelfde ouder van/én na
`article + section` → eerste section onder zelfde ouder van/én na article = 6
- **Tilde** (~): zelfde parent, **alle matches** onder zelfde ouder van/én na
`article ~ section` → alle sections onder zelfde ouder van/én na article = 6 en 7

subtiel verschil...

```
aside li, footer li { background-color: beige; }
```

Alle li elementen die of onder een aside of onder een footer tag liggen krijgen een beige achtergrond.

```
aside > ul { background-color: aquamarine; }
```

Alleen de ul elementen die **direct** onder een aside liggen krijgen een aquamarine achtergrond.

Combinaties

http://www.w3schools.com/cssref/css_selectors.asp

CSS1

V A Elk element A dat afstamt van een element V
(A is kind, kleinkind, achterkleinkind... van V)
→ *een afstammeling van een voorouder*

CSS2

O > K Elk element K als rechtstreeks kind van element O
→ *een kind van een ouder*

B:first-child Elk element B als eerste element van zijn ouder

Z + B Elk element B als eerstvolgende broer/zus (sibling) van
element Z (eerstvolgend kind van dezelfde ouder)
→ *de eerstvolgende broer van een zus*

CSS3

Z ~ B Alle elementen B als nakomende broer/zus (sibling) van
element Z (alle volgende kinderen van dezelfde ouder)
→ *alle volgende broers van een zus*

Pseudoklassen

Pseudoklassen enkel voor hyperlinks (a-element):

:link	Een niet bezochte link
:visited	Een bezochte link
:active	Een actief element (moment van klikken = mousedown)

Overige pseudoklassen:

:hover	Een element waar de muis over zweeft
:focus	Een element dat de focus heeft

:nth-child()
:nth-last-child()
:nth-of-type()
:nth-last-of-type()
:first-child
:last-child
: ...

<u>Andere</u> :checked :enabled :disabled :target :root :lang()

Pseudoklassen: :first-child :last-child

```
li:first-child { color: red; }
```

Geldig voor elk **eerste** element in een block

```
li:last-child { color: blue; }
```

Geldig voor elk **laatste** element in een block

Extra CSS toegevoegd, zie voorbeeld

Voorbeeld: H8/04_first_child_last_child

Zeeslag
Rush Hour
Mad Virus
Wippen
Dammen

Olifant
Neushoorn
Leeuw
Luipaard
Buffel

```
<ul>
  <li>Zeeslag</li>
  <li>Rush Hour</li>
  <li>Mad Virus</li>
  <li>Wippen</li>
  <li>Dammen</li>
</ul>
<ul>
  <li>Olifant</li>
  <li>Neushoorn</li>
  <li>Leeuw</li>
  <li>Luipaard</li>
  <li>Buffel</li>
</ul>
```

Pseudoklassen: :nth-child(value)

```
li:nth-child(odd) {  
  background-color: yellow;  
}
```

Geldig voor elk **oneven** element in een block

```
li:nth-child(even) {  
  color: blue;  
}
```

Geldig voor elk **even** element in een block

Zeeslag
Rush Hour
Mad Virus
Wippen
Dammen

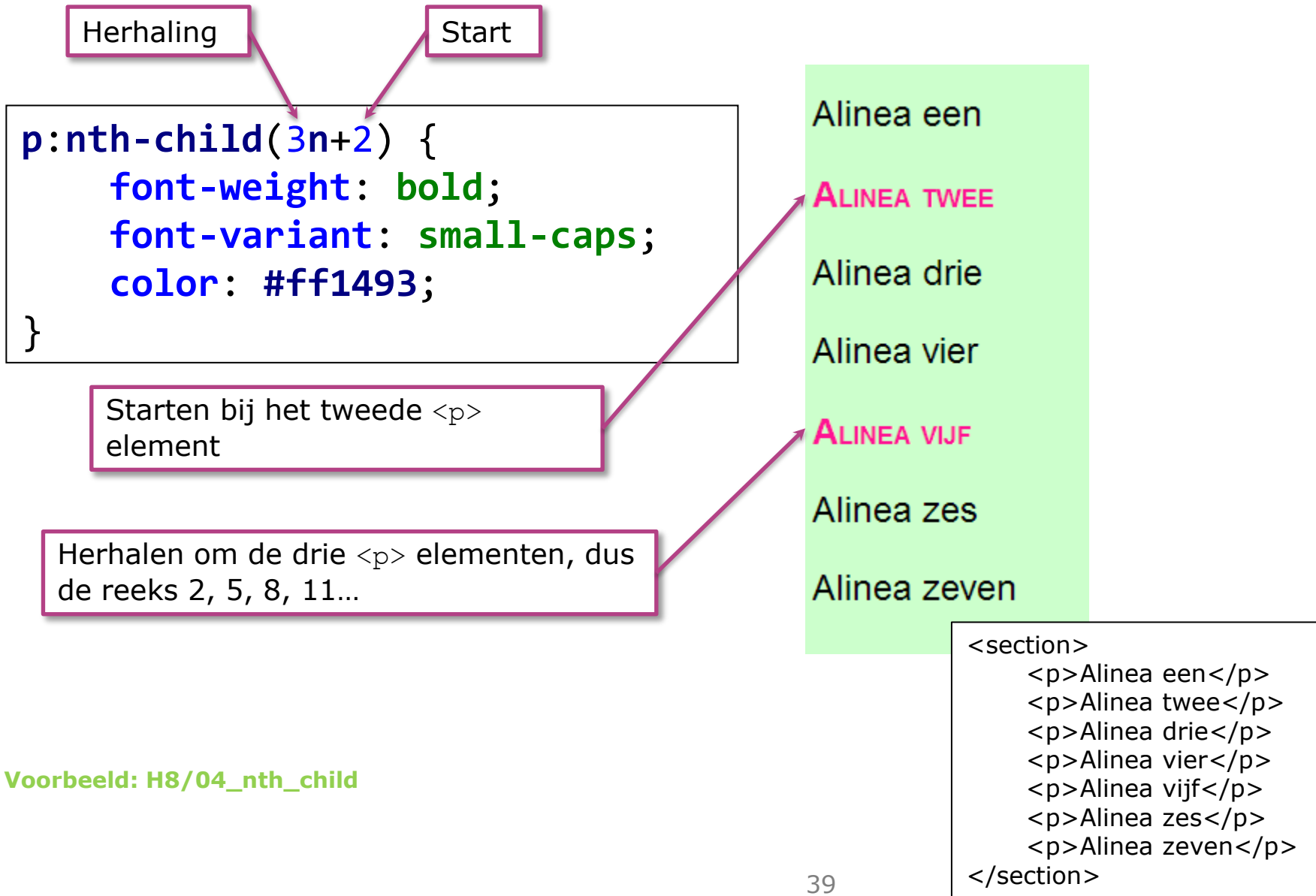
Olifant
Neushoorn
Leeuw
Luipaard
Buffel

```
<ul>  
  <li>Zeeslag</li>  
  <li>Rush Hour</li>  
  <li>Mad Virus</li>  
  <li>Wippen</li>  
  <li>Dammen</li>  
</ul>  
<ul>  
  <li>Olifant</li>  
  <li>Neushoorn</li>  
  <li>Leeuw</li>  
  <li>Luipaard</li>  
  <li>Buffel</li>  
</ul>
```

Voorbeeld: H8/04_odd_even



Pseudoklassen: :nth-child(value)



Pseudo-elementen

Pseudo-elementen:

<code>::first-line</code>	Opmaak voor eerste regel in de browser
<code>::first-letter</code>	Opmaak voor eerste letter
<code>::before</code>	Tekst invoegen vóór de inhoud van het element
<code>::after</code>	Tekst invoegen ná de inhoud van het element

Pseudo-elementen

Voorbeelden

```
p.regel::first-line {  
    font-variant: small-caps;  
}
```

Zorgt er voor dat de eerste regel van elke alinea (in de browser) met de class regel in kleine hoofdletters wordt getoond.

```
p::first-letter {  
    font-family: cursive;  
    font-size: 3rem;  
}
```

Zorgt er voor dat de eerste letter van elke alinea cursief en met 3x de normale grootte getoond wordt.

```
p.inleiding::first-letter {  
    color: red;  
}
```

Zorgt er voor dat de eerste letter van elke alinea met de class inleiding in het rood getoond wordt. Tevens worden de eigenschappen van de vorige rule overgenomen (overerving)

Voorbeeld: [H8/04_pseudo_elementen/pseudo_elementen](#)

Pseudo-elementen

Voorbeelden:

```
h1::before {  
    content: url(../images/smiley.png);  
}
```

Zorgt er voor dat elk h1 element vooraan een smiley krijgt.

```
p::after {  
    content: "...";  
}
```

Zorgt er voor dat elke alinea achteraan 3 punten krijgt.

Voorbeeld: [H8/04_pseudo_elementen/before_after_selection](#)