CSS

* Governed by W3C
* Language used to specify the presentation aspects (e.g. layout and formatting) of structurally marked up documents
* Developed by Hakon Wium Lie (CHSS) and Bert Bos (SSP)

Versions:

* CSS 1 1996
* CSS 2.1 2011 more functionalities were added (e.g. azimuth)
* CSS 3 modular specifications (e.g. color module, media queries)

\*\*Modules with the tag level 3 means in was available in CSS 2.1

CSS Preprocessors

* Sass, Less

CSS Frameworks – provide default formatting

* Bootstrap, Materialize, etc.

HTML/XHTML Stylesheets

3 possible styles:

* Author Style – CSS the authors of the site created
* external stylesheets (recommended)

- can be linked to multiple HTML pages

- placed on the head element

e.g. <head>

<link rel= “stylesheet” type = “text/css”>

- alternate styles and media queries can be added

* embedded styles - used for 1 page websites

- no separate file, embedded within HTML

* inline styles - styles are placed in the elements itself

- used for quick changes, NOT for normal practice

* User Style – used by the viewers of the page
* User Agent Styles – comes from the browser itself (e.g. default CSS 2.1 stylesheet for HTML 4)

CSS Statements

1. At-Rules – starts with @

e.g. @charset - to specify a character set

@import - to include other stylesheets

@media - for media queries

@font-face - for downloadable web fonts

@keyframes - for animations

@page - for printed page

@namespace - targetting SVG

@documents - targetting a specific document for CSS

2. CSS Rule Sets (aka CSS Rules, Style Rules)

selector

|

body {

display: block !important; property important declaration

color: green; declarations normal declaration

} | |

property property

name value

* Block Comment Syntax – for CSS comments

**CSS Selectors** – used as a condition in CSS rule

Subjects - matched selectors

Selector Syntax - chain of one or more sequences of simple selectors separated by combinators with one

pseudo-element

e.g.

combinators pseudo-element

| | |

d i v # a b c > p . x y z + s p a n [ t i t l e ] : : a f t e r

simple selectors

a. **Simple Selector**

* type selector – type of elements you are selecting
* universal selector
* attribute selector – elements are targeted based on a value

e.g. abbr[title] 🡨 [attr]

abbr[title = xyz] 🡨 [attr = value]

p[class = xyz] 🡨 [attr ~= value]

[lang l = en] 🡨 [attr l= value]

for CSS3:

[attr ^= value] starts with this

[attr $= value] ends with this

[attr \*= value] found anywhere

* class selector – depends on class value

e.g. p.xyz 🡨 class

* ID selector – for the ID attribute (which should be unique for valid HTML)
* pseudo-class

- dynamic pseudo-class

> link pseudo-classes (for hyperlinks)

: link

: visited (link was already seen)

> user action pseudo-class (for user interaction)

: hover (when pointer is hovered on something)

: active (when you click on something)

: focus (interaction with the element)

- target pseudo-class (exists in CSS 2.1)

> target (CSS 3) – allows to format a section that is referenced by a bookmark

e.g. :target {

color:red;

}

- language pseudo-class

> lang()

e.g. :lang(en){

}

- UI element states pseudo-class (interactive user elements)

> enabled

> disabled

> checked

> indeterminate

- structural pseudo-class

> root – targets root element of the document

> first-child – formatting ALL first-child elements

> last-child

> only-child – element with no siblings

> nth-child – value should be indicated

e.g. :nth-child(2n+1), nth-child(odd), nth-child(even)

> nth-last-child – count from the end

> first-of-type

> last-of-type

> only-of-type

> nth-of-type()

> nth-last-of-type()

> empty – targets empty selectors

- negation pseudo-class

> not() – negates, reverses the logic

**DOM Tree**

html

head body

title p div

p p p

em p

**CSS Combinators**

- descendant combinator (whitespace, i.e., space, tab, line feed, carriage return, form feed)

- child combinator ( > )

- sibling combinatory

> adjacent sibling combinator ( + )

e.g. h1+p (same level)

> general sibling combinatory ( ~ )

e.g. h1~p (any sibling)

**CSS Pseudo-Elements** (can only appear once)

* :: first-letter : first-letter
* :: first-line : first-line
* :: before : before
* :: after : after

e.g. h1::before{

content:”Topic”;

}

body{

counter-reset: tn 0;

}

h1{

counter-increment:tn 1;

}

h1::before{

content “Topic” counter(tn) “:”;

}

<div>

<p> \_\_\_\_\_\_\_ <em> \_\_\_\_\_\_\_\_\_\_\_\_\_\_ </em> </p>

</div>

for CSS: div p { color: blue; }

both allowed

div>p { color: red; }

**CSS Rule Precedence**

1. By **origin** (author, user, user-agent) and **importance** (if the declaratory has an important tag)

a. user agent important declarations

b. user important declarations

c. author important declarations

d. author normal declarations

e. user normal declarations

f. user agent normal declarations

for CSS 3: highest level is the transition declarations

2. by specificity (how specific is the selector)

a. inline style – always more specific than non-inline styles

b. number of ID (unique) selectors

c. number of class selectors, attribute selectors, and pseudo-classes

d. number of type selectors and pseudo-elements

e.g. <p id = x class = y>

*allowed*  p

*allowed* p.y

*allowed* p#x MOST SPECIFIC

3. by order – the LAST one WINS

**CSS Rule Precedence**

1. properties

* **transparent** - default value of background color
* **initial** - default value of color
* **span** has default inline elements that’s why height and width cannot be specified UNLESS display will be set to **block**
* if a **background-color** is set on the body element, the children will NOT inherit it because it’s default is **transparent**
* **media groups** are superceeded by media queries

a. shorthand properties longhand properties

e.g. font, background e.g. font color

b. vendor specific extensions (a.k.a. vendor prefix)

e.g. webkit – Chrome

moz – Mozilla

ms – Microsoft browsers

o – Opera

2. values

a. keywords – not enclosed in quotes

e.g. initial

b. numbers (integers and reals in decimal notation)

e.g. font-weight:100

c. dimension – number with a unit

e.g. 2px, 3em

* length, angle, duration, frequency, resolution
* length units:
* font-relative: em (font size)

ex (height of lowercase)

ch (zero character)

rem (root-element)

* viewport-percentage : vw (1% of viewport width)

vh (1% of height)

vmin

vmax

* absolute-lengths: cm, mm, q, in, pt, pc, px
* angle units: deg, grad, rad, turn (1 full 360)
* duration units: s, ms
* frequency units: hz, khz (formatting and stylesheets)
* resolution units: dpi, dem, dppx

d. percentages

e. URLs and URIs – specifying resources

f. colors

* keyword based {e.g. red}
* #fff -- hexadecimal digits
* rgb ( \_\_,\_\_,\_\_ ) {e.g. rgb (255,255,255) }
* rgbg ( \_\_,\_\_,\_\_ ) -- alphachannel-transparency
* hsl -- hue, saturation, lightness {e.g. hsl (0, 150%, 50%) }

**CSS Preprocessors**

* advanced syntax and compiles into valid CSS
* SASS (Syntactically Awesome Stylesheet; Ruby)
* SCSS (Sassy CSS)
* LESS (Node.js)

e.g. @function twice($v) {

return 2\*$v;

}

@mixin pm ($v) {

padding: $v;

margin: $v;

}

@ for $i from 1 through 6 {

h# {$i} {

font-size: 22px - $i#2;

}

}

*For* HTML Preprocessor (Haml)

%body

OUTPUT: proper HTML code

%div

%p Hello World!

**CSS Frameworks**

* makes use of matching styling (e.g. bootstrap buttons will look good with bootstrap tables)
* keywords: default success danger

info warning

-getbootstrap.com/css------

-getbootstrap.com/components------