

CSS

MPRI 2.26.2: Web Data Management

Antoine Amarilli

Friday, December 7th

Overview

- **Cascading Style Sheets**
- **W3C** standard:
 - CSS 1** 1996
 - CSS 2** 1998
 - CSS 2.1** 2011, 487 pages
 - CSS 3.0** Ongoing (various **modules**), snapshots:
<https://www.w3.org/Style/CSS/>
→ Cf caniuse.com and cssdb.org
- HTML describes **content/structure**, CSS describes **presentation**
 - **Several designs** for the same website
 - In particular for different **media**
(screen, phone, printing, screen readers, bots...)
 - **Reuse a design** across pages of a website or across websites

Table of Contents

Introduction

CSS and HTML

Selectors

Styling

Layout

Other features

Modern tools

Integrating to HTML (solution 1)

- **Property/value** pairs, e.g.:

```
color: red;
```

```
font-weight: bold;
```

- Can apply **directly** to HTML elements with the `style` attribute:

```
This is <em style="color: red;">red</em>
```

- If you need a dummy element, use `` or `<div>` (inline/block)

```
This text is <span style="font-weight: bold;">bold</span>
```

→ **Problem:**

- Mixes style and content
- Only works on a single element

Integrating to HTML (solution 2)

- **Selectors** to indicate to which elements a property applies
- `<style>` tag in `<head>`:

```
<!DOCTYPE html>
<html>
  <head>
    <style type="text/css">
      em { color: red; }
    </style>
  </head>
  <body>
    This is <em>red</em>
  </body>
</html>
```

Integrating to HTML (solution 3)

- Put the rules in a **separate** .css file
- Point to the file with the `<link>` **tag**
- Can be **filtered** with the `media` attribute:

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css"
          href="style.css" media="screen">
  </head>
  <body>
    This is <em>red</em>
  </body>
</html>
```

→ **Modular** and **reusable** styles

HTML and selectors

- How can we select **elements** to apply the style?

→ id attributes (**unique**) and class (**non-unique**)

```
<p id="thanks">
```

```
  Hello <span class="person">Pierre</span>!
```

```
</p>
```

- An element can have **multiple classes** separated by spaces

→ We can use **id** and **class** in CSS to select the right elements

General syntax

```
/* Comments */
```

```
@charset "UTF-8";
```

```
@import url(other.css);
```

```
@media screen {
```

```
    /* ... specific rules ... */
```

```
}
```

```
selector1, selector2 {
```

```
    property: value;
```

```
    property: value;
```

```
}
```


Conflicts and defaults

- If **no selector** applies, **default style** (browser-dependent)
 - **CSS reset** to align the defaults across browsers
- If **multiple selectors** apply to an element, complicated **conflict resolution** rules to choose the value of each property
 - Depends on counter-intuitive **specificity** heuristics: #a vs #b c
 - Can use **!important** to give more weight to a rule

Best practices

- Some **style guides** describe choices of how to use CSS
 - e.g., <https://github.com/airbnb/css>
- Various **philosophies** for large projects: OOCSS, BEM, etc.

Table of Contents

Introduction

CSS and HTML

Selectors

Styling

Layout

Other features

Modern tools

Elements, classes, IDs

```
strong {  
    /* Put important stuff in red */  
    color: red;  
}  
  
.person {  
    /* Underline names */  
    text-decoration: underline;  
}  
  
p.important {  
    /* Put important paragraphs in bold */  
    font-weight: bold;  
}
```

Combinations

```
#menu li {  
    /* All li elements in the element with id="menu" */  
}  
header > img {  
    /* All img which are immediate children of a header */  
}  
h2 + p {  
    /* The first paragraph after a h2 */  
    /* Also: ~ for successor elements */  
}  
img[src*=".svg"] {  
    /* All .svg images */  
}  
ul > li:first-child {  
    /* The first li which is an immediate child of a ul */  
}
```

Pseudo-elements and pseudo-classes

```
h2::first-letter { /* First letter of an h2 title */ }
```

```
p::before { /* Before a paragraph */ }
```

```
p::after { /* After a paragraph */ }
```

```
a:link { /* a elements which are links */ }
```

```
a:visited { /* a links that have been visited */ }
```

```
a:hover { /* a links which are hovered */ }
```

```
* { /* Universal selector */ }
```

Table of Contents

Introduction

CSS and HTML

Selectors

Styling

Layout

Other features

Modern tools

Measurements

% Percentage of the current value

em **Line height** of the current **font**

px Absolute size in **pixels**

→ Depends on the **screen resolution**


```
font-size: 70%;  
line-height: 150%;  
  
font-weight: bold;  
font-style: italic;  
font-variant: small-caps;  
  
text-align: justify;
```






Fonts

- Not all users have the same **fonts** installed
 - give a **list** with font names and fallback families
- With **CSS3**, fonts can be **downloaded** on-demand:

```
@font-face {  
  font-family: myFont;  
  src: url('myFont.woff');  
}  
  
body {  
  font-family: myFont;  
}
```

- **Web Open Font Format**, W3C, 2012. (Other formats.)

Colors and fill

Color	Css name	RGB	Hexa
	black	rgb(0, 0, 0)	#000000
	blue	rgb(0, 0, 255)	#0000FF
	red	rgb(255, 0, 0)	#FF0000
	teal	rgb(0, 128, 128)	#008080
	-	rgb(32, 64, 96)	#204060

Other possible fills:

Images `url("image.jpg")`

Gradients `linear-gradient(to right,
red,orange,yellow,green,blue,indigo,violet);`

Transitions

With **CSS3** we can **transition** between attribute values:

```
div {  
  background: green;  
  /* background changes should be a transition over 2 seconds */  
  -webkit-transition: background 2s;  
  transition: background 2s;  
}  
  
div:hover {  
  /* Change the background when hovered */  
  background: red;  
}
```

→ **animate.css**: library of animations

Table of Contents

Introduction

CSS and HTML

Selectors

Styling

Layout

Other features

Modern tools

Inline and blocks

- Two **element types** in CSS:
 - **Blocks** : <div>, <p>, <h1>, ...
 - **Inline** : , <a>, , ...
- The **display mode** in CSS can be changed with `display`:
 - `none` Invisible element
 - Still available to robots
 - Compare to `visibility`: `hidden`
 - `block` Displayed as block
 - `inline` Display as inline
 - `inline-block` Displayed as inline, behaves as block
(`inline block` in CSS3)
 - `contents` Only display the **children**

Borders

```
/* red border of 1 pixel */
```

```
border: 1px solid red;
```

```
/* 3d outset border of 2 pixels */
```

```
border: 2px outset black;
```

```
/* round corners by 10 pixels */
```

```
border-radius: 10px;
```

```
/* Add some shading */
```

```
box-shadow: 5px 5px 1px black;
```

```
/* Required for compatibility */
```

```
-webkit-box-shadow: 5px 5px 1px black;
```

Width and height

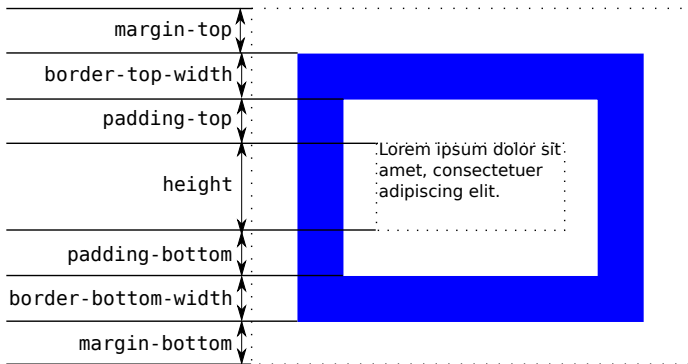
- `height` and `width`
- By default, `auto`: elements are as wide as **possible** and as high as **necessary**
- `min-width` and `max-width` to specify **limits**
 - e.g., ensure that text is not too **wide** (legibility)

Overflow

- If the size is limited, the content may **overflow**
- **overflow** property:
 - visible** The overflowing content is **visible**
(outside of the element)
 - hidden** The overflowing content is **hidden**
 - auto** **Scrollbars** are added if necessary
- **word-wrap**: **break-word** to ensure that long words can be **cut**

Margins and spacing

- Properties `margin` and `padding`



- **box-sizing:** `border-box` makes width and height include **borders** and **padding**
- Special case: `margin: auto` to **center** an element

Positioning

- `position` property to position the element...
 - `static` ... in the **flow** (default)
 - `relative` ... **relatively** to its default position (offset)
 - `absolute` ... relative to the **origin** of the closest non-static **parent** (by default, `<html>`)
 - `fixed` ... relative to the **window**
- For non-static, use the `top` `bottom` `left` `right` properties
- `z-index` describes the **vertical order** in case of overlap

Floats and clearing

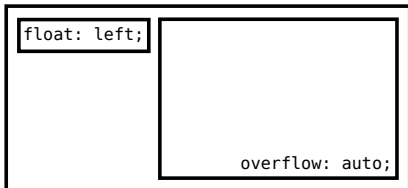
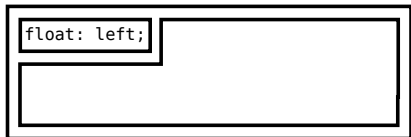
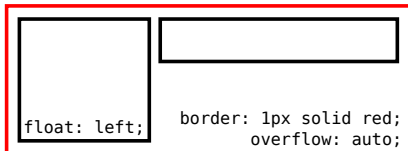
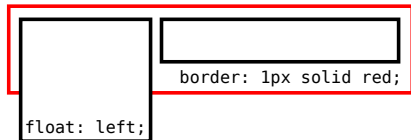
- float to position elements **outside the flow**

```
div#menu {  
    float: right; /* also: left */  
}
```

- clear to **clear** all floating elements

```
footer {  
    clear: both; /* also: left, right */  
}
```

Float subtleties



Columns

- CSS3 supports **columns** for text:

```
p {  
    /* Minimal width and maximal number of columns */  
    columns: 300px 3;  
    /* Separation between columns */  
    column-gap: 20px;  
    /* Rule between columns */  
    column-rule: 1px solid black;  
}
```

→ Vendor-specific prefixes:

-webkit- for Safari and Chrome

-moz- for Firefox

→ Tools to **automatically** put the required prefixes: **autoprefixer**

Grids and flexbox

- **CSS3** makes it possible to position elements in **two dimensions** using **CSS Grid**
 - Use `display: grid` on the element which will serve as a grid
 - Use `grid-template-columns` and `grid-template-rows` to describe the grid
 - Use `grid-column: span 4` to describe how many columns an element should fill

→ Very similar to **table-based design!**
- Also possible to position elements in **one dimension** using `display: flex`

Table of Contents

Introduction

CSS and HTML

Selectors

Styling

Layout

Other features

Modern tools

- Insert text **before** or **after** an element

```
p.reminder::before {  
  content: "Do not forget:";  
}
```

- **Subtle:** is this text **presentation** or **content**?
- Can be used to **insert** an attribute value:
content: attr(src url)

Counters

Examples to **number** sections and subsections

```
article {
  counter-reset: section subsection;
}
article h2 {
  counter-increment: section;
  counter-reset: subsection;
}
article h2::before {
  content: counter(section) ". ";
}
article h3 {
  counter-increment: subsection;
}
article h3::before {
  content: counter(section) "." counter(subsection) ". ";
}
```

Variables and calc

- With **CSS3** we can do **calculations** for attribute values:

```
width: calc(100% - 80px);
```

- We can also define **variables** to be reused:

```
:root {  
  --my-color: red;  
}  
  
h1 {  
  background-color: var(--my-color);  
}
```

- These variables can be **changed** with JavaScript

Feature queries (not in IE)

We can test with `@supports` if the browser supports a feature

```
@supports (display: grid) {  
  div {  
    display: grid;  
  }  
}
```

```
@supports not (display: grid) {  
  div {  
    float: right;  
  }  
}
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/@supports>

Media queries

- Specify **different stylesheets** depending on the **media**
- **media** attribute for <link> (avoids loading useless stylesheets) or @media in CSS
- Testing the **rendering size**: max-width: 1024px (displays less than 1024 pixels wide), etc.
- **Orientation** (for smartphones): orientation: portrait

```
/* Printable version */
```

```
@media print
```

```
/* Vision-impaired */
```

```
@media aural
```

```
/* Screens of less than 1024px wide */
```

```
@media screen and (max-width: 1024px)
```

Mobile refinements

- Some mobiles have **HiDPI** displays with a **scaling factor** between the small **physical pixels** and **CSS pixels**
- By **default**, mobile browsers pretend to have a large **viewport**
 - This allows them to display websites from before the mobile era
 - The user can **zoom in** and **scroll**
- **Mobile-aware** browsers should ask mobile browsers to render them in their actual width by adding to **head**:

```
<meta name="viewport" content="width=device-width">
```
- Check mobile **best practices**, e.g.,
search.google.com/test/mobile-friendly

Table of Contents

Introduction

CSS and HTML

Selectors

Styling

Layout

Other features

Modern tools

Bootstrap is a **toolkit** of HTML, CSS and JavaScript templates



Checkout form

Below is an example form built entirely with Bootstrap's form controls. Each required form group has a validation state that can be triggered by attempting to submit the form without completing it.

Billing address

First name

Last name

Username

Email (Optional)

Address

Address 2 (Optional)

Your cart

3

Product name Brief description	\$12
Second product Brief description	\$8
Third item Brief description	\$5
Promo code EXAMPLECODE	-\$5
Total (USD)	\$20

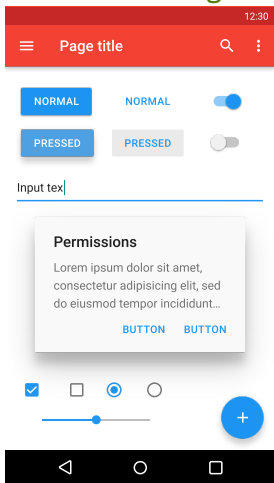
Promo code

Redeem

Design philosophies

A topic for an **entire class!** Common philosophies:

Material design:



Flat design:



Skeuomorphism: imitating the real world in Web design

CSS preprocessors and other tools

- **Preprocessors:** Add features to CSS. Main ones: **Less** and **Sass**
 - E.g., **variable** declarations, **loops**, reusable **mixins**, selector **nesting**...
 - **Sass** is more commonly used
 - **Less** is just JavaScript, **Sass** uses Ruby
 - **Less** has better error reporting
 - **Sass** is more powerful, **Less** is more minimalistic
 - **Bootstrap** uses Sass (v4) and used Less (v3)
 - Other options: **PostCSS**, **Stylus**, etc.
- **linters** for CSS: **stylelint**, also `validator.w3.org`
- Also **CSS minifiers**

- Matériel de cours inspiré de notes par Pierre Senellart et de code par Pablo Rauzy
- Merci à Pierre Senellart et Pablo Rauzy pour leur relecture