The three Amigos

What’s in a webpage?

HTML: HyperText Markup Language

Gives structure and meaning to content and links to other web pages

CSS: Cascading Style Sheets

Defines the presentation (look and feel) of your site

JS: JavaScript

Adds interactivity to your site

CSS

Cascading Style Sheets

- ✓ CSS fundamentals
- ✓ Types of CSS implementation
- ✓ Using selectors
- ✓ Using the box model

F27WD – Web Design and Databases
School of Mathematical & Computer Sciences

S.Chumbe@hw.ac.uk
EMB G41
HERIOT WATT UNIVERSITY

CSS fundamentals

Definitions

CSS is a language to design and create a great-looking web pages. With **CSS** you can define how HTML elements are displayed. **CSS** is implemented using ***styles***.

A ***style*** is a rule that describes how to format a specific part of an HTML document. A *style sheet* is a set of style rules.

You can create a style and apply it to many elements based on a **selector**. You use a *selector* to locate and select elements based on tag name, class name, ID, and more. You can create a style that works with images, and you can create a style that works only with hyperlinks. You can also create a named style that you can apply to any element. The reusability of **CSS** is powerful.

CSS fundamentals

Applying Style Sheets

1. **Inline** style sheet within a tag. Applies only to that particular occurrence of that tag.
2. **Embedded** (also called Internal) style sheet is defined within the head section of a page. Applies to that page only.
3. **External** style sheet defined in a separate, hence **external**, file.

CSS fundamentals

inline styling

```
<p style="font-size:14px;">…</p>
```

Pros

- ✓ Highly specific to the element on which it is defined.
- ✓ You don’t need a selector.
- ✓ It is handy to override styles that are defined elsewhere.

Cons

- Hard to maintain: it is bad for reusability because you will need to copy this style to each HTML document you want to style.
- HTML/CSS coupled: it violates the primary goal of separation between structure and presentation.

CSS fundamentals

embedded (block) styling

```
<style>
  p { font-size: 14px; }
</style>
```

Pros

- ✓ Affects all matched elements
- ✓ Useful when you want to have a single, stand-alone webpage that contains everything needed to render.
- ✓ It can be located within the <head> or the <body> elements. Better in <head>.

Cons

- HTML/CSS coupled: Still, it does not provide file separation. It only provides reuse within the files.
- You need to use SELECTORS

CSS fundamentals

external style

separate CSS file

```
p {
   font-size: 14px;
}
```

mystyle.css

Pros

- ✓ Easy to maintain: Write once for whole site
- ✓ HTML & CSS decoupled: **Themes !!!**

Cons

- It can become harder to manage
- You need to use SELECTORS

CSS fundamentals

external style

separate CSS

```
font-size: 14px;
```

my style.css

external style

Pros

- ✓ Easy to maintain: Write once for whole site
- ✓ HTML & CSS decoupled: **Themes !!!**

Cons

- It can become harder to manage
- You need to use SELECTORS

What’s in a webpage?

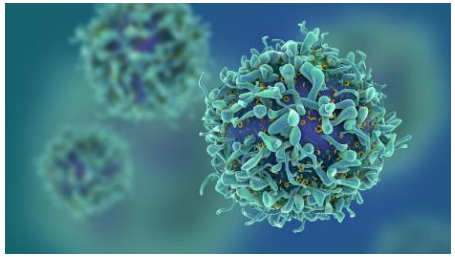
What's in a webpage?

index.html

1 external JS:
main.min.20181002.js

3 embedded JS

3 images



2 external CSS:
non-critical.min.20180912.css
legacy.min.20180604.css

1 embedded CSS

5 inline CSS

(2 embedded JSON)

Creating themes with style

- http://www.wordpress.com
- https://www.w3schools.com/w3css/w3css_color_themes.asp

"There is only one way to avoid criticism: do nothing, say nothing, and be nothing."

– Aristotle

Take the CSS challenge

- http://www2.macs.hw.ac.uk/~santiago/F27WD/no_dejavu.html

CSS fundamentals

selectors

A selector connects the style rule to your HTML

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

CSS fundamentals

selectors

- ✓ element-type: (div)
- ✓ id (#name)
- ✓ class (.name)

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

CSS fundamentals

selectors

Element-type selectors

An *element type selector* is based on the name of the tag. In this example, the tag name (**button**) is the selector and the style will be applied to every **button** in your HTML document.

```
button {
        background-color: white;
        color: red;
}
```

If your HTML document contains 50 buttons, the style of all 50 buttons would be set. This is desirable in some scenarios, but if you want to set the style on a single button or a subset of buttons, you should use the **class** or the **id** selectors.

CSS fundamentals

selectors

id selectors

An *id selector* is based on the id of the element. For example, to set the style on a single button, you can assign an id to the button and then specify the id as the selector, prefixed with the hash (#) symbol.

HTML

```
<button id='btnSave'>Save</button>
```

CSS

```
#btnSave {
        background-color: white;
        color: red;
}
```

In this example, it doesn't matter which type of element is being accessed; all that matters is that the **id** is btnSave. The **id** must be unique across an HTML document. **You cannot have two elements with the same id.**

CSS fundamentals

selectors

class selectors

A *class selector* is a style with a class name of your choice, prefixed with the period (.) symbol.

HTML

```
<button class='myStyle'>OK</button>
<button class='myStyle'>Cancel</button>
```

CSS

```
.myStyle {
    background-color: black;
    color: orange;
}
```

Class styles promote reuse because they can be used on any element as needed.

CSS fundamentals

using an external style

default.css

```
body {
        background-color: gray;
        color: red;
}
```

test.html

```
...
<link rel='stylesheet' type='text/css' href='default.css' />
...
```

http://www2.macs.hw.ac.uk/~santiago/F27WD/html/test.html

```
@import url('header.css');
```

CSS fundamentals

Selector chain

Style inheritance

HTML

```
<ul>
   <li>
        <a href="#">Home</a>
        <p>
           <a href="#">Help</a>
        </p>
   </li>
   <li>
         <a href="#">About</a>
    </li>
</ul>
<a href="#">Shop</a>
```

CSS

```
li a {
      text-decoration: none;
}
```

This example removes the underline from **<u>every</u>** hyperlink that is a descendant of a list item, regardless of whether the hyperlink is a child, grandchild, or distant descendant.

This is different to grouping selectors!

```
li , a {
      text-decoration: none;
}
```

CSS fundamentals

Selector chain

Style inheritance

HTML

```
<ul>
   <li>
        <a href="#">Home</a>
        <p>
           <a href="#">Help</a>
        </p>
   </li>
   <li>
         <a href="#">About</a>
    </li>
</ul>
<a href="#">Shop</a>
```

CSS

```
li > a {
      text-decoration: none;
}
```

This example removes the underline from hyperlinks that are direct children of a list item only.

CSS fundamentals

pseudo-class and pseudo-element selectors

- How do you assign a style to the first line of a paragraph?
- How do you assign a style to a hyperlink that has been visited?

Pseudo classes select elements based on something other than name, attributes, or content and, usually, something that cannot be deduced from the HTML document.

:root
:link
:visited
:active
:hover
:focus

:checked
:lang
:not Example:
div:not("#mainContainer")
:first-of-type
:only-of-type

:only-child
:nth-child(formula) : For example, li:nth-child(3) selects the third list item.
:nth-last-child(n) : For example, li:nth-last-child(3) selects the third list item from the end of the list.

CSS fundamentals

pseudo-class and pseudo-element selectors

Pseudo elements are abstractions of the HTML document that provide access to information that is not directly available via element-type, class or id styles.

- **::first-line** Selects the first line of each paragraph. Allow you apply a different style to the first line of a paragraph.
- **::first-letter** Selects the first letter of each paragraph. Useful when you want to create a large first letter.
- **::before** Inserts generated textual content into each paragraph directly before the existing content. You can provide a style for *content* too. For example:
  p::before{ content: “Note: “; color: red;} sets the color of “Note: ” to red.
- **::after** Inserts generated textual content into each paragraph directly after the existing content. For example:
  p::after{ content: “Done!”; color: red;} sets the color of “Done!” to red.

CSS fundamentals

pseudo-class and pseudo-element selectors

Condition of action

Condition of existence

Location where content will be displayed

```
a[href]:hover::after {
        content: " ("attr(href) ")";
        background-color: yellow;
}
```

http://www2.macs.hw.ac.uk/~santiago/F27WD/html/test.html

CSS fundamentals

pseudo-class and pseudo-element selectors

CSS variables

```
:root {
    --red: #ff6f69;
    --beige: #ffeead;
    --yellow: #ffcc5c;
}

html, body {
    background: var(--beige);
    color: var(--red);
}
```

http://www2.macs.hw.ac.uk/~santiago/F27WD/html/test.html

CSS fundamentals

properties

Typography

font-size
font-weight
font-family
line-height
text-align

Colors

color
background-color
background-image
border-color

Positioning

position
width, height
margin
padding
border

view all

http://htmldog.com/reference/cssproperties/
http://www.w3schools.com/cssref/

CSS fundamentals

selectors

HTML element	CSS rule
`<p>`	`p { font-size: 14px; }`
`<p class="bp">`	`.bp { color: gray; }`
`<p id="headline">`	`#headline { font-size:20px; }`

CSS fundamentals

cascading selectors

(inheritance)

HTML element

```
<div class="intro">
   <h1>…</h1>
   <p>…</p>
</div>
```

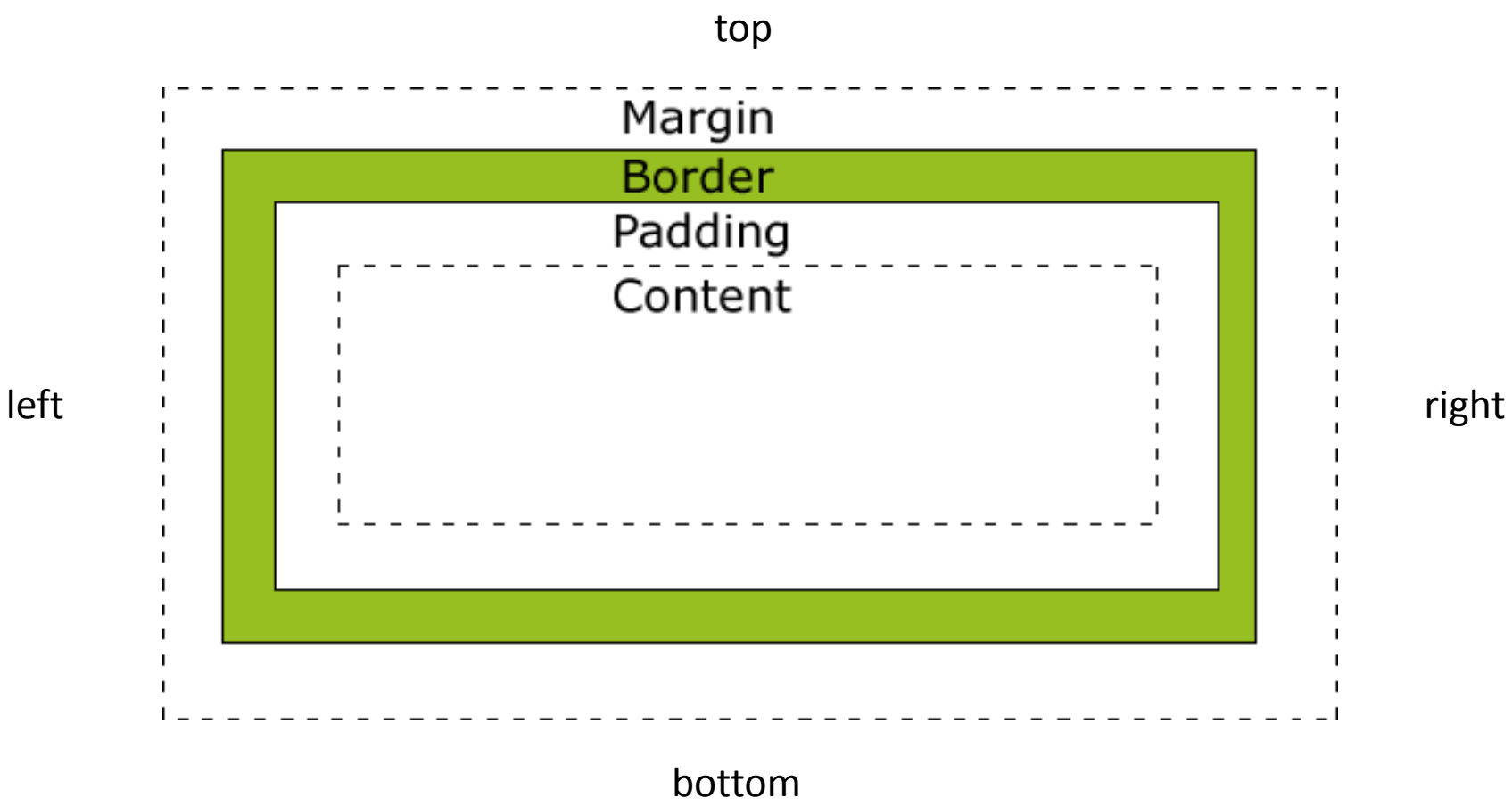
CSS rule

```
.intro h1 {
  font-size:18px;
}
```

It styles `h1` and every element inside div element with `class="intro"`

CSS fundamentals

box model



CSS fundamentals

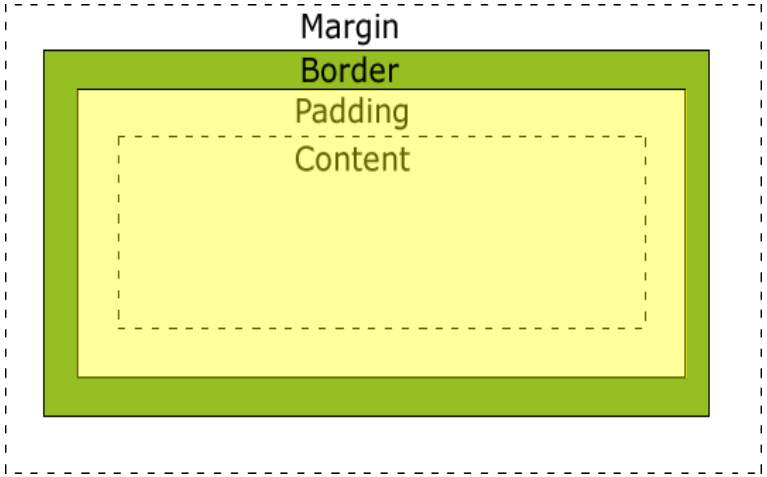
box model

The **margin** is the space outside the **border**, between the border and the next element.
The **padding** is the space inside the border, between the border and the **content**.

If the border is being displayed, the margin and padding settings will have distinct effects. If the border is not being displayed, it can be difficult to differentiate margin and padding settings.

CSS fundamentals

box model




```
main {
    margin: 15px;
    border: 10px;
    padding: 25px;
    background-color: yellow;
    border-style: solid;
    border-color: green;
}
```

http://www2.macs.hw.ac.uk/~santiago/F27WD/html/test.html

CSS fundamentals

Size Units

```
main {
  margin: 15px;
  border: 10px;
  padding: 25px;
  background-color: yellow;
  border-style: solid;
  border-color: green;
}
```

px: (*pixel*) One pixel is equal to one dot on the computer screen.

✓ **em:** an *em* is equal to the current or default size set by the browser. For example if the current font size is 12px, 2em will be equal to 24px.

✓✓✓ **Percent (%):** It is much like the "em" unit, but using percentages. Thus, the current size is equal to 100% (i.e. 12px = 100%).

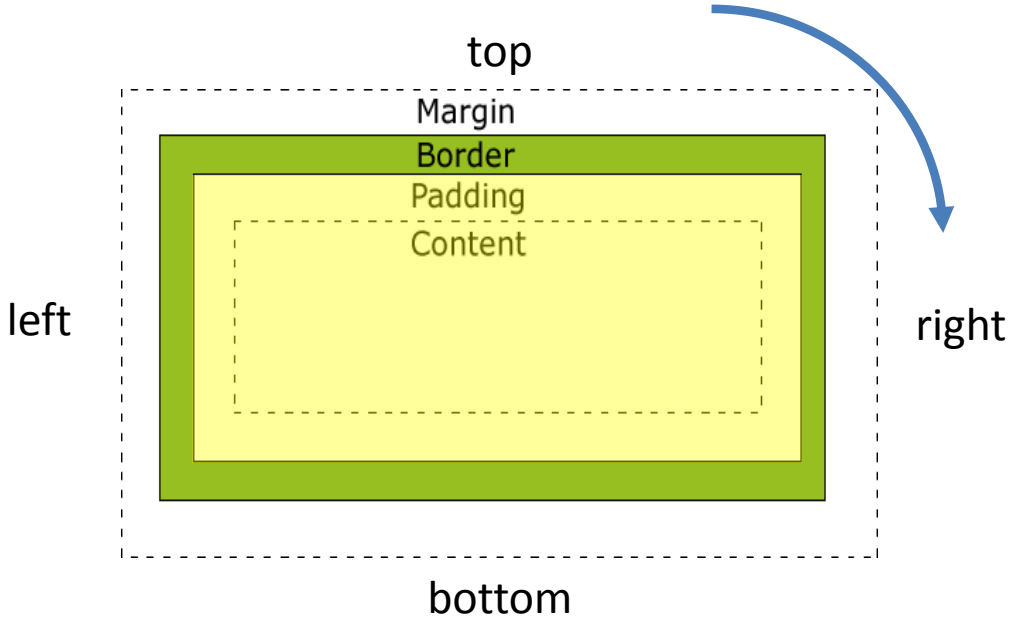
✓✓ **rem:** It is similar to *em* but the size always refers back to the root element (<html>).

✗ **pt:** (*points*) They are traditionally used in print media and are are much like pixels, in that they are fixed-size units and cannot scale in size.

CSS fundamentals

box model

```
main {
  margin-top: 0px;
  margin-right: 5px;
  margin-bottom: 10px;
  margin-left: 1px;
  padding: 1px 2px 3px 4px;
  border: 15px;
  background-color: yellow;
  border-style: solid;
  border-color: green;
}
```



http://www2.macs.hw.ac.uk/~santiago/F27WD/html/test.html