

# CodeSoc: Web development

...

Class 6: External style

# Last time

- Collected all the sites into a *root* folder
- Added local images into an *assets* folder
- Added navigation to our sites
- Briefly looked at the *cascading* natures of CSS
  - This will be a main focus of today

# Outline

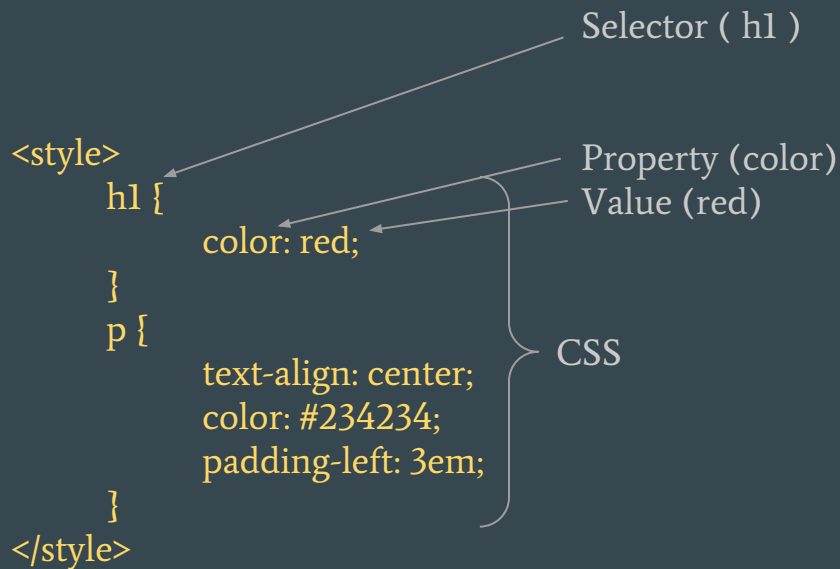
1. Introduction and creation of simple example website
2. (Today) The HTML document - Page structure, content types & best practises
3. More HTML elements & adding style (part i)
4. More HTML elements & adding style (part ii) - consolidating what we know
5. Multi-page sites & navigation
6. External style
7. Advanced CSS
8. Introduction to Javascript

# Today

- How we can manage the styling of our page
  - We seen that just using inline-style attributes can get messy
- How we can *cascade* style to other elements
- Using the *class* attribute to style different types of elements in the same way

# CSS

- Recall from the first class that CSS looks quite different to HTML



The diagram illustrates the components of a CSS rule. It shows a code snippet within a `<style>` block. The first rule is for the `h1` selector, followed by a declaration block in curly braces. The declaration contains two lines: `color: red;` and `text-align: center;`. A bracket groups these two lines under the label 'CSS'. Arrows point from labels to specific parts: 'Selector ( h1 )' points to `h1`, 'Property (color)' points to `color`, and 'Value (red)' points to `red`. The closing tag `</style>` is at the bottom.

```
<style>
  h1 {
    color: red;
    text-align: center;
  }
</style>
```

Selector ( h1 )

Property (color)

Value (red)

CSS

Everything inside the { }  
is called the declaration

# Selector

- The selector is HTML element we want to apply style to
- We have seen `h1` and `h1: hover` previously
- If we use these then they apply to every `h1` element, this isn't always what we want to do
- To solve this we can give the heading we want to style a new attribute called *id*
- To select only this heading our selector is “`#mainHeading`”
- Each *id* should be unique

```
<h1 id= “mainHeading”>  
  Title  
</h1>
```

```
#mainHeading {  
  color: blue;  
}
```

# Selector

- We can also identify elements we want to style by using the attribute **class**
- This allows us to give the same style to two different elements (say a heading and a paragraph)
- The selector is “.blueThings”

```
<h1 class= “blueThings”>
  Title
</h1>
.blueThings {
  color: blue;
}
```

```
<p class= “blueThings”>
  Content
</p>
```

# Selector

- An element can have multiple classes, by separating them with a space
- In this example the paragraph will be blue and indented

```
<h1 class= "blueThings">  
  Title  
</h1>
```

```
<p class= "blueThings indentedThings">  
  Content  
</p>
```

```
.blueThings {  
  color: blue;  
}  
.indentedThings {  
  padding-left: 30px  
}
```

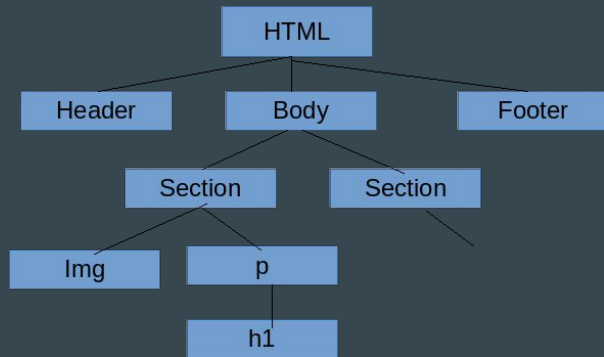


# Refactoring

- Pick one of your websites to move all of the styling into a style element
  - Remember this has to be **outside** of the **html** tags
- Now *refactor* the website, using each of the different selectors
  - `<h1>` = `h1 { }`
  - `<h1 id= "myId">` = `#myId { }`
  - `<h1 class= "myClass">` = `.myClass { }`

# The Cascading effect

- You may have noticed this effect - where you style one element and it changes others
- When you think about the nesting of the elements (elements within elements), any style applied to the containing element is propagated to those elements inside



# Overriding

- Consider this example:

<code>&lt;p&gt;</code>	<code>p {</code>
<code>  &lt;h1&gt;</code>	<code>  color: red</code>
<code>    Title</code>	<code>}</code>
<code>  &lt;/h1&gt;</code>	<code>h1 {</code>
<code>&lt;/p&gt;</code>	<code>  color: blue</code>
	<code>}</code>

- Which color should our heading be?

# Overriding

- Here we see that the h1 style has overridden the p style
- The order for styling a particular element is:
  - Any inline style (Most overriding)
  - Any id attribute style
  - Any class attribute style
  - Any element type style (Least overriding)
- Secondly, any styling that cascaded to the element is overridden by all of the above
-

# External style

- Rather than using a style element, we can create a new file called main.css
  - Note the .css rather than .html
- First create a new file
- Copy and paste all of the code between the style tags, into the new file
- Save as main.css in your assets folder
- Now back in the html add the following lines at the top, but inside the html tags:

```
<head>
```

```
  <link rel="stylesheet" type="text/css" href="assets/main.css">
```

```
</head>
```

- Note how the href is identical to how we link local pictures!

# External style

- The style in this CSS file can be linked to any of your other websites using the same code
- This mean creating one style for the whole site is much easier than having to rewrite the CSS for every new page!

# Wrap up

- Using the class and id attributes to group and individually select elements to style
- Cascading
- Overriding

# Next time

- Flex-boxes
  - Much better layout than using tables
  - Easy (responsive(!)) design
- Drop down boxes
- More styling