CSS
Part Two

Website Development 1

Agenda

- Classes
- IDs
- Descendant selectors
- Styling lists

Class selector

- We previously looked at elements selectors, which targets all elements (which translates to HTML tags) on a page. If we want to be more specific, class selectors is the next step. Instead of targeting all elements with a specific name, they target all elements that has a specific class name specified.
- A class selector looks just like an element selector, but instead of using names that are tied to the names of HTML elements, you make up the name and then you prefix it with a dot (**.**). For instance:

```
.red { }

.myElements { }

.navigation { }
```

Class selector

```
<style>
 p { font-size:20px;}

.pchange { color: red;}

.beautiful { font-weight: bold;
             font-style: italic;
}
</style>
```

Class selector

```
<p> Here is some text.</p>
<p class="pchange">
Here's more text –
<span class="beautiful">this part is
especially pretty!</span>
</p>
```

Class selector

- Element specific classes
  - In our previous example, all element types could use our classes but in some situations, you may want to limit the use to a specific element type. This is usually done to indicate how the class is supposed to be used, to allow for more than one class with the same name and avoid conflicts. Element specific classes are used simply by appending the class name to the element name in your selector.

```
p.pchange { color: red;}
```

  - With this rule, this specific class selector will only apply to any paragraph (p) tag.

Class selector

- Multiple classes
  - Classes are not unique and the class property of an HTML tag allows you to specify more than one class. The cool thing about this is that it allows you to combine the rules for several selectors and use them for the same tag however you want to.
  - This also means that instead of writing selectors with many rules and then only targeting few elements, you can write less specific selectors and simply combine them when it is appropriate. This allows for greater re-usability, which is really what CSS is all about.

Class selector

```
<style>
 .status { background-color:yellow;
           font-weight:bold;
           font-size:20px;
 }

  .error {color: red;}

  .information {color: blue;}
</style>
```

Class selector

```
<body>
 <p class="status error">
        This is an error!
 </p>

 <p class="status information">
        This is information!
 </p>
</body>
```

Class selector

- Here we use CSS to show status information. We have a common .status selector and then we have a selector for error messages and one for informational messages. Error and information messages obviously share stuff, since they are both a type of messages, but they also have distinct looks. So, we put the shared stuff in a class called .status, and then put the rest in different classes called .error and .information, and then we use them on the p tags, simply by separating their names with a space.

Agenda

- Classes
- IDs
- Descendant selectors
- Styling lists

ID selector

- Now we will look at the most specific selector type: The *id selector*. The id selector is actually so specific that it only targets a single element on the page.
- An id selector looks just like a class selector, but instead of having a dot as the prefix, it uses the hash sign (#).
- It works just like classes, but instead of using a dot, we use a hash character, and instead of using the class attribute, we use the id attribute - the difference lies in the fact that the id should be unique. The value of the id attribute should be unique, according to the HTML specification, meaning that it can only be used on a single element per page.

ID selector

- Element specific ID selectors
  - Just like the class selector, you may limit an id selector to a specific element type by putting the name in front of the selector name, like this:

```
h1#main-header {
        color: greenyellow;
}
```

  - With this rule, this specific id selector will only apply to a header (h1) tag.

Agenda

- Classes
- IDs
- Descendant selectors
- Styling lists

Descendant Selector

- So far, we have only used selectors which directly targeted a specific element or element(s) with a specific id or class. Especially targeting elements of a specific type, e.g. all links or images, is very powerful, but what if you want to limit this, for instance to elements found only in a specific part of the page? This is where combinators come into the picture, a range of selector types which uses the hierarchy of elements on your page to limit the elements targeted.
- Here we will look into the **Descendant selector**, which allows you to limit the targeted elements to the ones who are descendants of another element. The syntax is very simple - you simply write the parent(s), separate with a space, and then the actual element you want to target.

Descendant Selector

```
<style>
 p b {
        color: blue;
 }
</style>
```

Descendant Selector

```
<p>
 To <b>Err</b> is human. To
 <b>forgive</b> divine!
</p>

<blockquote>
 To <b>Err</b> is human. To
 <b>forgive</b> divine!
</blockquote>
```

Descendant Selector

- In this example, I want all bold elements to be blue, but only if they are inside a paragraph tag. If you try the example, you will see that while the bold tag is used four times, only the first two of them are blue - that's because they are inside a paragraph, which our descendant selector requires!
- This is obviously a very basic example, but think of the bigger picture - for instance, this would easily allow you to change the colour of all links inside your main menu, without having to tag them all with a specific class!

Descendant Selector

A descendant doesn't need to be the direct child

- With this selector type, you should be aware that not only direct children are targeted - also children of the child (grandchildren) and so on will be targeted, all the way down through the hierarchy.

```
div.highlighted b {
        color: blue;
}
```

Descendant Selector

```
<div class="highlighted">
        <b>Level 0...</b>
        <div>
                <b>Level 1...</b>
                <div>
                        <b>Level 2...</b>
                </div>
        </div>
</div>
```

Descendant Selector

- Here, we target bold elements which are descendants of a div tag with the class "highlighted". If you try the example, you will notice that even though we wrap the last set of bold tags in several layers of div tags, it is still affected by the rule about blue bold tags.

Child Selector

- We have seen just how powerful the descendant selector can be, because it allows you to target ALL children and grandchildren (and so on) of one or several elements. However, sometimes this can be TOO powerful - sometimes you only want to target the direct children of an element. Fortunately for us, CSS has a selector for this as well!
- The syntax for using the direct child selector looks like this:

```
parent > child
```

Child Selector

```
<style>
 div.highlighted > b {
        color: blue;
 }
</style>
```

- Now, only direct children of the parent are now affected.

Sibling Selector

- What if you want to target siblings instead? CSS has a couple of selector types for that as well, here we'll check out the general sibling selector.
- With the general sibling CSS selector, which takes a selector, followed by a tilde character (~) and then the selector you wish to target, you can target elements by requiring the presence of another element within the same parent element. Another requirement is that the first part of the selector needs to be present in the markup BEFORE the targeted element, even though they are all children of the same parent.

Sibling Selector

```
<style>
 h2 ~ p {
        font-style: italic;
 }
</style>
```

Sibling Selector

```
<h1>Hello, world!</h1>
<p>Some text here</p>
<h2>Hello, world!</h2>
<p>Some text here</p>
<p>More text here</p>
```

- So, all of the text elements are children of the same element (body). We then specify that paragraph elements which are siblings to an H2 element should be in the *italic* style. As you can see, this means that the last two paragraph tags will be in italic, but not the first, because it comes before the H2 element in the markup.

Adjacent Sibling Selector

- We have just looked at the sibling selector, which allows us to select all elements which follows another element within the same parent. However, using the Adjacent sibling selector, you can limit this to only include the first element which comes directly after the first element in the markup.

```
<style>
h2 + p {
        font-style: italic;
}
</style>
```

Adjacent Sibling Selector

```
<h1>Hello, world!</h1>
<p>Some text here</p>
<h2>Hello, world!</h2>
<p>Some text here</p>
<p>More text here</p>
<h2>Hello, world!</h2>
<p>Text here as well...</p>
<p>But no more!</p>
```

Adjacent Sibling Selector

- With the adjacent sibling selector, we have just specified that the first paragraph element after all H2 elements should use italic text.
- The syntax for the adjacent sibling selector is - the two selector parts are simply joined by a plus character (+).

Agenda

- Classes
- IDs
- Descendant selectors
- Styling lists

Styling lists

The list-style-type Property

- The list-style-type property defines the kind of marker that is to be associated with each item in the list.

- By default, an unordered list displays with an item marker of a bullet (disc). In nested unordered lists, the item marker changes to an open circle for the first level of indentation, and a square for the second level. What if you prefer to have the item marker be a square for the outermost list, a bullet for the next one, and an open circle for the third?

Styling lists

```
ul {
  list-style-type:square;
}
ul ul {
  list-style-type:disc;
}
ul ul ul {
  list-style-type:circle;
}
```

Styling lists

- For an ordered list you can change from the default numbering system to alphabetic characters or roman numerals, for example:

```
ol {
 list-style-type:upper-roman;
}
ol ol {
 list-style-type:lower-roman;
}
```

Styling lists

- If you want no bullets or numbers:

```
ul {
  list-style-type:none;
}
```

Styling lists

- Two other properties available to lists are:
  - list-style-position
  - list-style-image