CSS

Cascading Style Sheets

CSS Setup

Cascading Style Sheets

The basic structure of every web page, HTML, is very plain on its own. The beautiful websites that you see across the internet are styled with a variety of tools, including CSS.

CSS, or Cascading Style Sheets, is a language that web developers use to style the HTML content on a web page. If you're interested in modifying colors, font types, font sizes, shadows, images, element positioning, and more, CSS is the tool for the job!

Example 1

Download the source code for example 1 on the online syllabus.

Open the .html file with Sublime, then preview it in Chrome or Firefox.

We should see all the html elements unstyled as plain html.

Add the following code to line 5 of the .html file:

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

Save your file, and now refresh the example.html page

We should see the entire styling of the website changed. Let's take a minute to look at the accompanying style.css file. The code we wrote linked the .css file to the .html page and applied the styling.

Inline styling

Although CSS is a different language than HTML, it's possible to write CSS code directly within HTML code using *inline styles*.
To style an HTML element, you can add the `style` attribute directly to the opening tag. After you add the attribute, you can set it equal to the CSS style(s) you'd like applied to that element.

Remember, we've done this before by using the style attribute to change the position of an <img> with float.

Let's try it again with example2.html. Download this file and open it in your text editor and browser.

Let's change the font of the first paragraph by applying a style attribute into the right <p> tag.

```
style="font-family: Arial;"
```

The <style> Tag

HTML also allows you to write CSS code in its own dedicated section with the `<style>` element. CSS can be written between opening and closing `<style>` tags. To use the `<style>` element, it must be placed inside of the `<head>` element.

Let's delete the style attribute we added to the <p> tag in example2.html and move it into a style tag.

Like this:

```
4   <head>
5     <title>Vacation World</title>
6     <style type="text/css">
7       p{
8         font-family: "Arial";
9       }
10    </style>
11  </head>
```

Your turn!

Change the font-color of the paragraph to red.

Change the font-style of the paragraph to italic.

Change the font-size of the paragraph to 30px.

Target the <img>. Change it’s float position to right.

.

And finally!

Convert this html file with inline CSS to an html file that links to an external stylesheet like demonstrated in example 1.

You'll need to copy the code you wrote between the style tags and paste it into a new file. Save that file as a .css file and make sure to delete the <style> </style> tags from both files.

Make sure the .css file is saved in a place that is accessible to your .html file.

In your .html file, link to the .css file you created within the <head> tags:

<link href="your_file_name.css" type="text/css" rel="stylesheet">

CSS Selectors

Tag Names

The basic CSS selector targets html tag names.

We've seen how we are able to target html elements like the <p> and <img> tags in the code you wrote previously.

For example, let's target the elements in our source code and change the color to maroon.

Any element with the tag of will be affected by the CSS styling we specify.

Class Names

CSS is not limited to selecting elements by tag name. HTML elements can have more than just a tag name; they can also have *attributes*. One common attribute is the `class` attribute. It's also possible to select an element by its `class` attribute.

For example, consider the following HTML:

```
<p class="brand">Sole Shoe Company</p>
```

The paragraph element in the example above has a `class` attribute within the `<p>` tag. The `class` attribute is set to `"brand"`. To select this element using CSS, we could use the following CSS selector:

```
.brand {

}
```

To select an HTML element by its class using CSS, a period (`.`) must be prepended to the class's name. In the example above case, the class is `brand`, so the CSS selector for it is `.brand`.

In your stylesheet for example2, target the class “title” and change the color to teal.

Multiple classes

It's possible to add more than one class name to an HTML element's `class` attribute.

For instance, perhaps there's a heading element that needs to be green and bold. You could write two CSS rules like so:

```
.green {
  color: green;
}

.bold {
  font-weight: bold;
}
```

And structure the html accordingly by including both classes one the html element:

```
<h1 class="green bold"> ... </h1>
```

We can add multiple classes to an HTML element's `class` attribute by separating them with a space. This enables us to mix and match CSS classes to create many unique styles without writing a custom class for every style combination needed.

Let's practice with our example2 files. In the .html file, find the h1 tag with a class of "title". Let's add a second class to this called uppercase so the html looks like this:

```
11 ▾   <h1 class="title uppercase">
```

And in our .css file, let's add the following styling:

```
10 ▾ .uppercase{
11       text-transform: uppercase;
12   }
```

CSS classes are meant to be reused over many elements.

The convention is to use classes to target multiple html elements at once, i.e. if multiple elements all need the same styling

ID Name

If an HTML element needs to be styled uniquely (no matter what classes are applied to the element), we can add an ID to the element. To add an ID to an element, the element needs an `id` attribute:

```
<h1 id="large-title"> ... </h1>
```

Then, CSS can select HTML elements by their `id` attribute. To select an `id` element, CSS prepends the `id` name with a hashtag (`#`). For instance, if we wanted to select the HTML element in the example above, it would look like this:

```
#large-title {

}
```

Let’s try using an #id as a selector

Let’s add an id called “article-title” to our h1 element in example2.html:

```
11 ▾    <h1 class="title uppercase" id="article-title">
```

And now we’ll style it in our .css file:

```
17 ▾ #article-title {
18       font-family: cursive;
19       text-transform: capitalize;
20   }
```

While classes are meant to be used many times, an ID is meant to style only one element. IDs override the styles of tags and classes. Since IDs override class and tag styles, they should be used sparingly and only on elements that need to always appear the same.

Pseudo-classes

CSS pseudo classes

A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s). For example, :hover can be used to change a link’s color when the user's pointer hovers over it.

```
/* Any link over which the user's pointer is hovering */

a:hover {

  color: blue;

 }
```

Class files

class7/class_selectors/selectors.html

class7/class_selectors/styles/selectors.css

Your turn!

So far, all the CSS selectors we've seen map directly to a piece of HTML markup that we wrote. However, there's more going on in a rendered web page than just our HTML content. There's "stateful" information about what the user is doing (opposed to the content we've authored).

The classic example is a link. As a web developer, you create an <a href>element. After the browser renders it, the user can interact with that link. They can hover over it, click it, and visit the URL.

Save a new copy of the class file, and assign unique ID's to the 4 links that don't have them.

Style each link's link, active, and hover states so they are all different.

CSS Buttons

Pseudo-classes aren’t just for styling text links—they can be applied to any kind of selector (not just type selectors).

Let’s look at pseudo-buttons (not actually html button elements) styled with CSS:

class7/class_selectors/selectors2.html

class7/class_selectors/styles/selectors2.css