Lecture #11: Exploring HTML and CSS

CS106E Spring 2018, Young

In this lecture we take a look at Cascading Style Sheets (CSS). We learn some of the benefits to using CSS and study the actual mechanics of writing CSS rules. We look at different selectors, which allow us to determine which HTML elements a rule will affect. We also take a quick tour of the different CSS properties that can be used to format a webpage.

As CSS allows us to change colors, we examine different techniques available to us for specifying colors in CSS. We take a closer look at how to link from one webpage to another, learning when to create an absolute link vs. a relative link. We also learn how to style links using pseudo-classes. We end with a look at how to place images on a webpage and how to use CSS box properties on them.

Cascading Style Sheets

As we saw in the last lecture, modern webpages use HTML to provide semantic information about the meaning of text, such as what is a major heading vs. what is a paragraph, whereas they use CSS to provide presentation information. This will provide us with several very important advantages:

- We can write a single CSS rule that effects all HTML elements of a particular type. This means that if we want to switch all <h3> headers from Red to Blue, we don't have to go through the HTML looking for all the <h3> headers. We change it once in a single location and they all are instantly updated.
- In fact, as we'll discover, we can place our CSS information in a separate file from our HTML. This means we can write a single set of CSS rules and use it for every HTML file on our website. If we want to change the look of our website, we change that single file and instantly everything is updated.
- In addition separating our presentation from our semantic information will make it easier to create different presentation types, such as one set of rules for laptop-sized screens and a different set for mobile phones.

We'll break our discussion into three parts. First, we'll discuss how to include CSS information in your website. Next, we'll discuss how to select which elements on a webpage will be affected by specific CSS rules. Finally, we'll take a look at what sorts of appearance changes we can actually specify with CSS.

The <link /> and <style> Tags

- There are two different ways to specify CSS for a particular HTML file. The first method is to refer to a separate, external CSS file using the <link /> tag. The second is to include a <style> tag in your HTML file's <head> section and then to enter the CSS directly into the HTML file.
- If you're providing presentation information that you want to use across all the pages of your website (or even across a small subsection of your website) you should use the <link /> tag. With this tag, the CSS is placed in a separate file, which can be shared by many other HTML files.

- You can use the <style> tag if your CSS formatting only makes sense for a specific HTML file and you're sure you won't use the same styling rules elsewhere on your website.

Linking External Files with <link />

- The <link /> tag is a standalone tag with no end tag (which is why I've been writing it as <link /> instead of <link>).
- This tag creates a relationship between two different files.
  - By far its most common use is specifying a relationship between an HTML file and a CSS file that will provide presentation information.
  - Other relationships have appeared in the official HTML specification including:[1]
    - Specifying a help file for a webpage.
    - Specifying a glossary file to associate with a webpage.
    - Specifying which webpage should come before and which should come after the webpage, if the user is following the "standard" progression for reading the website.
- To use the <link /> tag, provide the URL of the associated file and specify how it is related to the original file. For example, here we specify that the file "example.css" (which is in the same location as our HTML file) should be used as a style sheet for our HTML file.

```
<link rel="stylesheet" href="example.css" />
```

- You may link more than one stylesheet to a webpage using <link /> tags. For example, if you have one stylesheet that applies to all webpages on your website and another more specific stylesheet that applies to webpages on your website that are about sports, you can load both these stylesheets by using two separate <link /> tags.

Specifying Styles Internally with the <style> tag

- Using the <style> tag is very easy. Simply place a <style> start and </style> end tag in your <head> section of your HTML file.
- Write your CSS rules between the <style> start tag and the </style> end tag.

CSS Rules

Whether we're using an external style sheet or an internal <style> tag, our styling information consists of a series of rules. Each rule has 1) a selector that specifies which HTML elements the rule applies to and 2) a declaration block which sets one or more CSS style properties such as the color or font-family for HTML elements that match the selector.

Selectors

Let's take a look at some of the most common selectors we can use — we'll take a look at the CSS properties available after we're done with the selectors.

- The simplest selector is a type selector. This selector affects all HTML elements of a particular type. Here we turn all <h3> headings to the color red:

```
h3 { color: red; }
```

[1] These example <link /> uses have changed from one version of HTML to the next and do not seem to have taken off. I list them here to give you a sense of why we have a general purpose <link /> tag rather than something that is specifically designed to load stylesheets only.

- We can create a rule that affects any one of a number of different types by separating our HTML element names with commas. Here we turn all six headings to red. Make sure to use a comma to separate the HTML tag names, if you separate them using just spaces, it does something entirely different.

  ```
  h1, h2, h3, h4, h5, h6 { color: red; }
  ```

- Sometimes we'll want to write a rule for some elements of a particular type, but not others. We can do this by giving these HTML elements a class attribute-value pair. Here I've marked the first of these two <h3> tags as "important" but not the other one.

  ```
  <h3 class="important">Computer Science</h3>

  <h3>Electrical Engineering</h3>
  ```

  We can change all elements marked as important by using a class selector. Form a class selector by using the period, followed by the name of the class. With this rule, every element that has class="important" will be underlined.

  ```
  .important { text-decoration: underline; }
  ```

  You can limit the class selector to just elements of a particular type by combining it with a type selector. This modified rule only affects h3's which are important. If you have, for example, a <p> paragraph marked as important, this rule won't affect it.

  ```
  h3.important { text-decoration: underline; }
  ```

  Finally, if you have an element that you want to give more than one class to, you can. Simply list multiple classes separated by spaces all in the same class attribute-value pair. Here I've specified an <h3> that is both important and fun – this means that I can use .important or .fun class specifiers to select this element:

  ```
  <h3 class="important fun">Computer Science</h3>
  ```

- We can also write rules that affect only a single element by using an id selector.[2] To use this we add an id attribute-value pair to the element in our HTML.

  ```
  <h3 id="graduation">June</h3>
  ```

  We then use the id selector, formed by using the number sign followed by the id.

  ```
  #graduation { text-size: xx-large; }
  ```

There are quite a few additional selector types available including selectors that only choose elements if they have a specific HTML attribute set, or that choose elements if they are contained within another specific type of element. You can find a summary of different selector types here:

https://www.w3schools.com/cssref/css_selectors.asp

[2] You may wonder why we bother with the special id selector when we could do the same thing by creating a class and only using it on one element. id acts as a form of documentation letting anyone maintaining the webpage know that there should only ever be one element that fits in that particular category.

The Cascade
You might have noticed that more than one rule can affect an element. What happens if one of the rules says to turn an element red, and a different rule, which also applies to the element, says to turn the element green?

Conflicts between rules are resolved by a process called *the cascade*. The cascade compares the different selectors and gives a different weighting to each rule. The rule with the most weight takes precedence over all other rules. In general, the rule that is most specific will win the cascade. So a class selector takes precedence over a type selector, but a selector that combines both class and type will beat a regular class selector.

If two rules conflict and they both have equal precedence, whichever rule comes last will win the cascade.

Span and Div
What happens if we want to style part of our HTML file, but that part of HTML file isn't actually an element. Suppose for example, I want to turn all instances of the word "Stanford" to the color red. If every time the word "Stanford" appears, it's also in italics, then I could do this by 1) adding a class="Stanford" to all <i> tags surrounding the word Stanford and 2) writing a style rule with a class selector for .stanford. This does have the side effect thought that in order for this to work, Stanford must appear in italics everywhere.

In HTML:

```
… <i class="stanford">Stanford</i> …
```

In CSS:

```
.stanford {color: red;}
```

The <span> and <div> tags allow us to do the same thing, except minus the italics. You can place the <span> tag around any words or phrases in your document. By itself, the <span> tag doesn't actually do anything. However, you can put a class on the <span> tag, and then you'll be able to use a class selector to select your word or phrase.

In HTML:

```
… <span class="stanford">Stanford</span> …
```

In CSS:

```
.stanford {color: red;}
```

<span> is an inline tag, and as we saw in the last lecture, you can't surround a block-level tag with an inline-tag. Therefore, HTML also provides the <div> tag, which is a block-level tag. You can surround block-level tags with <div> to write style rules affecting everything in the <div>.

Declarations and CSS Properties
Now that we've got some sense of how to select elements, let's take a look at the second half of a CSS rule, the declaration block. A CSS rule consists of a selector and a declaration block. As we've seen the declaration block is surrounded by curly braces '{' and '}'. Inside the curly braces we can list as many CSS properties and values as we want, separating the pairs with semicolons. This rule, for example, sets h1 headings to 48pt font, makes them red, and underlines them:

```
h1 {
   text-size: 48pt;
   color: red;
   text-decoration: underline;
}
```

If you start creating webpages seriously, you'll probably want to look through a full list of CSS properties. The list has gotten quite long, and unfortunately there is also some variation on which properties are supported by which web browsers. You can find a list here:

https://www.w3schools.com/cssref/default.asp

Broadly speaking though, we can separate most CSS properties into some specific categories.[3] Here's an overview mentioning some of the most important properties.

Font Properties allow you to set the font-family, set text to italics (with font-style) or bold (with font-weight).

Text Properties allow us to control word-spacing and letter-spacing. We can also control vertical-align and text-align (for horizontal alignment), these alignments work on some, but not all HTML elements.

Color and Background Properties allow us to set color and background-color. There are also a number of properties that allow us to set a background-image behind an HTML element and control a variety of different properties of that image.

Box Properties allow us to put borders around items and to float items to the sides of the webpage with text flowing alongside. We'll take a look at this in some more detail below.

Classification Properties allow us to turn a block-level tag into an inline, text-level tag.[4] They also allow us to change the bullet in front of an unordered list to a different symbol, such as a square, and allow us to control the indices used on ordered lists, switching them from numbers to letters, or even Roman numerals.

Position Properties allow us to place elements at specific locations on the webpage.

Inheritance

Some CSS properties are inherited. This means that if an HTML element is inside another element and the outer element is selected by a CSS rule, the inner element will inherit the property from the outer elements. For example, if I have a <div> that contains a <h1> and a <p>, and a rule that says that the color of the <div> is Red, text in both the <h1> and <p> will also appear in red.

Other CSS properties are not inherited. Continuing my example, if I have a rule that says the <div> should have a 3-pixel solid blue border, this does not mean that I want my <h1> to also have its own solid blue border and my <p> to have a different solid blue border. Border properties are not inherited.

[3] This is the category list from CSS2. CSS3 consist of many different specs in various states of development. However, the CSS2 category list is still a good way for beginners to categorize properties until they dig deeper.

[4] This is commonly done with lists. If I have an ordered list, items in the list will typically be displayed as blocks, with bullets in front of each block. Sometimes though, I'll want the items in the list to be displayed left-to-right instead of top-to-bottom. I can do this by changing the list items in the list from block display to inline display.

In general, the inheritance settings are setup so that everything works the way you might expect it to. However, it's worth keeping in mind, as you may run into a case where something does not appear working correctly on your webpage until you look more carefully and see that either something is inheriting and you weren't expecting it, or vice-versa.

Colors in CSS
There are a number of different ways to specify color in CSS.

- The traditional method consists of listing the amount of Red, Green, and Blue as hexadecimal digits preceded by a number sign '#'. Here are a few examples:

  Red is #FF0000 where the first two hexadecimal digits FF represent the amount of red. As you may recall F is the maximum value we can represent in a single hexadecimal digit so FF is the maximum value in a pair of hexadecimal digits. Here we've turned Red all the way up. The first 00 is the amount of Green and the second 00 is the amount of Blue. #FF0000 means maximum Red, no Green, and no Blue.

  Purple is #800080. Here the Red is 80, which is halfway to FF. We have no Green (that's the middle 00) and Blue is at 80, so it's also at half maximum.

  Navy Blue is #000080. No Red, No Green, half intensity on blue. #0000FF would be a much more intense Blue.

- CSS3 provided a list of 147 predefined colors. These include such colors as cornflowerblue, darkslategray, firebrick, and lightgoldenrodyellow. These will work in most web browsers, although if you really want to be careful, you can use their hexadecimal equivalents, as most tables listing the names will include both the name, and the hex codes for them. Here's a link to the official list.

- CSS will also allow you to specify the intensities of Red, Green, and Blue in decimal like this: rgb(255,0,0) for Red, rgb(128,0,128) for Purple, and rgb(0,0,128) for Navy Blue.

- CSS also supports some other conventions that I'll let you look up, including specifying the color as Hue, Saturation, and Brightness.

Linking to Other Webpages
- We've previously seen that we can create a link between webpages using the <a> anchor tag in conjunction with an href attribute-value pair:

```
<a href="http://www.stanford.edu/">Stanford University</a>
```

- There are actually two type of links we can work with

  Absolute Links are used to create a link from our webpage to a different webpage *on another web server*. With this type of link, you list the full URL for the page you are linking to.

  Relative Links are used when linking from one webpage to another webpage *on your own website*. Here, for example, I use a relative link to my web tutorial page (which is in the same directory as the current HTML file), but an absolute link to the validator website.

```
<a href="tutorial.html">Web Tutorial</a>
<a href="https://validator.w3.org/">Validator</a>
```

- Use relative links when you can. They will make it easier to move files from one web server to another or from one folder on your website to another folder.

- We can reference subdirectories and parent directories with href links. Use a forward slash '/' (even on computers that normally use the back slash) to get to subdirectories and a double period ".." to get to a parent directory.

```
<a href="examples/web.html">Web Examples</a>
<a href="../css/css-examples.html">CSS Examples</a>
```

- In addition to http: and its secure equivalent https:, there are other types of URLs used on websites. These include:

  mailto: which can be used to initiate the email program.

  javascript: which executes JavaScript code.

Styling Links using Pseudo-Classes

- The standard styling for a <a> link is to turn it blue or purple and to underline it. Blue indicates a link to a webpage we haven't visited before and purple indicates a link we've seen previously. In fact, there are other colors involved as well, if you watch carefully, you'll discover that the link text changes to a third color (typically red) when you click the mouse on it, but before you release it.

- How can we set these colors and other styling information for our links? We can write a style rule that applies to all <a> tags, but we can't really write a rule for <a> when the user has visited the link destination and a different rule for those the user hasn't visited, since these rules vary from one user to another.

- You can imagine if all the <a> tags either had class="link" for unvisited links and class="visited" for links the user had previously visited, we could write rules for each of these. But of course this doesn't work, because the class settings on the <a> tags would have to change from one user to another.

- CSS provides a number of pseudo-classes. These act as if we had put a class="link" or class="visited" on the <a> tag, but we don't actually have to do it, and the web browser calculates whether or not to use class="visited", based on browser history. These work exactly the same as regular classes, except we precede them with a colon ':'. Here is an example of setting different styles for links. I've set the standard link colors to red and removed the standard underline.[5] I've set links to webpages the use has already visited to green.

[5] Be careful removing the underline. Without the underline the user may not realize something is actually linked.

```
a:link {
  color: red;
  text-decoration: none;
}
a:visited {
  color: green;
  text-decoration: none;
}
```

- Here are the most common pseudo-classes you'll run into. There are also some additional ones now available, which you can look up if you decide to explore web development further.

  :link provides styling for standard links

  :visited provides styling for links that a user has already visited.

  :active provides styling when the user is actually pressing on an element

  :hover provides styling when the user is moving the mouse cursor on top of an element (but is not actually clicking on it).

  While these initially were only supported on <a> tag links, :active and :hover work on other items as well. In fact, you can define a :hover for any tag that renders something on screen[6] and the element will use your :hover style rules when the user moves the mouse on top of them. Because of this, you want to make sure to specify the element type the rule works on. The following rule will put a line above and below a link when the user moves their mouse on top of it:

  ```
  a:hover {text-decoration: overline underline;}
  ```

  In contrast, this rule applies to every element (as we have not specifically limited it to <a> elements). Any element the user moves their mouse on top of will display overline and underline.

  ```
  :hover {text-decoration: overline underline;}
  ```

Pseudo-Elements
CSS also supports something called pseudo-elements, which act as if an element was there, even when it isn't. These will allow you to do things like select and write style rules for the first line of an element or the first letter in an element.

This rule sets the first letter in every paragraph to 36pt font:

```
p:first-letter {text-size: 36pt;}
```

Images and Box Properties
Let's take a look at how to place images on the webpage, and explore use of box properties. Please note that while box properties are often used on images, they can also be used with other types of elements as well.

- We can place an image on our webpage using the <img /> tag. This tag requires two attribute-value pairs. 1) a src attribute (short for source) provides the URL where the actual image file is located. 2) an alt attribute provides some sort of text identifier for the image, this can be used by web browsers for the visually impaired. Here is an example using the <img /> tag:

[6] Most HTML elements render something on the screen, but not all. For example, the <head> tag and the <meta> tag don't display anything on screen and are not eligible for :hover.

```
<img src="molly.jpg" alt="photo of my dog" />
```

- If I want to write style rules for my image, unless those rules will apply for all the other images on my webpage, I'll need to either give my image an id or a class:

```
<img src="molly.jpg" alt="photo of my dog"
     id="photo" class="left-images" />
```

- While the actual photo has a native height and width, I can also specify a height and width in CSS. Even if I want the image to appear at its actual native height and width, specifying a height and width in CSS will allow the web browser to leave the proper amount of space for the image while it is still waiting for the image to download. If the web browser does not know the height and width of the image, it will instead display a small-sized image icon for the image and then will need to reformat the webpage when the actual image arrives.

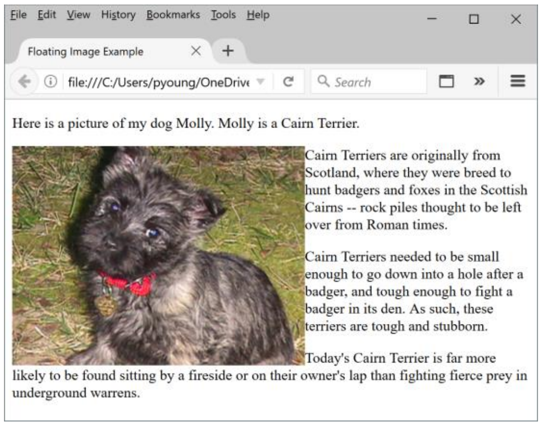
```
#photo {
  height: 640px; width: 480px;
}
```

  make sure to specify your units of measurement (in this case px, which stands for pixels). If you do not provide a measurement unit, the web browser will ignore your settings.

- Floating an element, such as an <img /> causes the web browser to place the element either on the right-side of the webpage or the left-side of the webpage and allows text to flow around it.

```
.left-images {
  float: left;
}
```

  here is an image floated to the left, which allows the text to flow around it on the right.


File Edit View History Bookmarks Tools Help
Floating Image Example
file:///C:/Users/pyoung/OneDrive
Search

Here is a picture of my dog Molly. Molly is a Cairn Terrier.

Cairn Terriers are originally from Scotland, where they were breed to hunt badgers and foxes in the Scottish Cairns -- rock piles thought to be left over from Roman times.

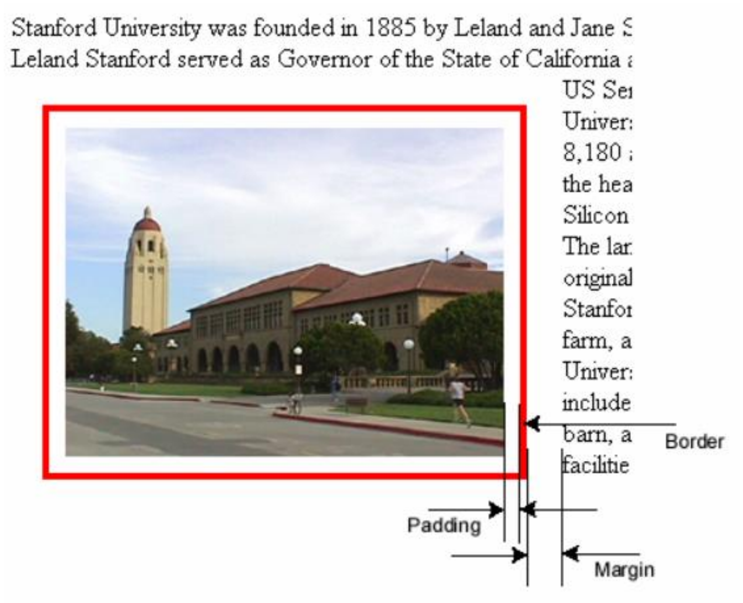
Cairn Terriers needed to be small enough to go down into a hole after a badger, and tough enough to fight a badger in its den. As such, these terriers are tough and stubborn.

Today's Cairn Terrier is far more likely to be found sitting by a fireside or on their owner's lap than fighting fierce prey in underground warrens.

- CSS allows us to control the border, padding, and margin.
  - Padding is the space between an element and its border. Margin is the space between the border and any surrounding items.
  - The top, bottom, left, and right border, padding, and margin can all be controlled individually. You can use this for a variety of different purposes. If you want to put a line between two paragraphs, for example, you could specify that the bottom paragraph had a border-top but no borders on the bottom, left, or right.

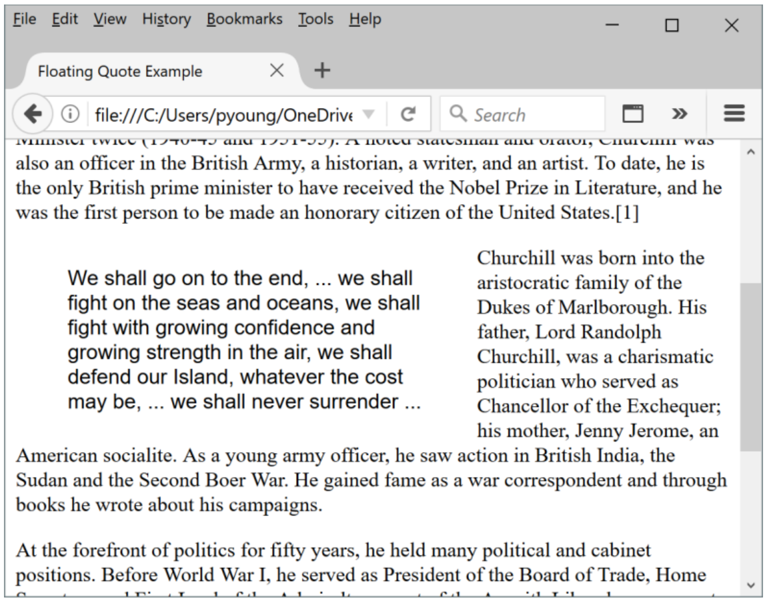
- While the padding and margin simply need a size specified, the border also needs both a color and a style. Don't forget to specify a style, if you specify a border size and a border color, but skip the style, no border will appear.
  - Border styles include solid, dashed, dotted, inset, and a few other options.

```
#stanfordPhoto {
    float: left;
    border: 3px solid red;
    padding: 5px;
    margin: 10px;
}
```



Stanford University was founded in 1885 by Leland and Jane S
Leland Stanford served as Governor of the State of California
US Se
Univer
8,180
the hea
Silicon
The lar
original
Stanfor
farm, a
Univer
include
barn, a
facilitie

- Don't forget the box properties can be used for things other than just images. Here I've floated a <blockquote>. When floating text, don't forget to specify a width. Most text-based elements do not have a natural width and will expand to fill all available space. For this example, I explicitly set the <blockquote> to a width of 275px.



Minister twice (1940-45 and 1951-55). A noted statesman and orator, Churchill was also an officer in the British Army, a historian, a writer, and an artist. To date, he is the only British prime minister to have received the Nobel Prize in Literature, and he was the first person to be made an honorary citizen of the United States.[1]

> We shall go on to the end, ... we shall fight on the seas and oceans, we shall fight with growing confidence and growing strength in the air, we shall defend our Island, whatever the cost may be, ... we shall never surrender ...

Churchill was born into the aristocratic family of the Dukes of Marlborough. His father, Lord Randolph Churchill, was a charismatic politician who served as Chancellor of the Exchequer; his mother, Jenny Jerome, an American socialite. As a young army officer, he saw action in British India, the Sudan and the Second Boer War. He gained fame as a war correspondent and through books he wrote about his campaigns.

At the forefront of politics for fifty years, he held many political and cabinet positions. Before World War I, he served as President of the Board of Trade, Home