

Week 3b Lecture : More About CSS

Screencast of Recorded Lecture (2022) -- around 33 minutes into the lecture:

- https://sjeccd-edu.zoom.us/rec/play/_/Eq_fV5LdA8qGaLKZDIBwtUCcvrAZdqPZk7HWPeJltD9YS1zhrk5znqda7QwvOhJVErM6bPDVxBMzKvRV.wf5a_ahGvAKjT5bQ ➞ [\(https://sjeccd-edu.zoom.us/rec/play/_/Eq_fV5LdA8qGaLKZDIBwtUCcvrAZdqPZk7HWPeJltD9YS1zhrk5znqda7QwvOhJVErM6bPDVxBMzKvRV.wf5a_ahGvAKjT5bQ\)](https://sjeccd-edu.zoom.us/rec/play/_/Eq_fV5LdA8qGaLKZDIBwtUCcvrAZdqPZk7HWPeJltD9YS1zhrk5znqda7QwvOhJVErM6bPDVxBMzKvRV.wf5a_ahGvAKjT5bQ)

Screencast of Recorded Lecture (2021):

- https://sjeccd-edu.zoom.us/rec/play/_/uOcLnFnkv2w2pZHYaP8rt4zOCFPqtiR9SomT0H93J6kQ6apgQf8DRyZUYRmzGdIAJFd12BSrvu1XxsPL.OMCsUpiLudKzoTjM [\(https://sjeccd-edu.zoom.us/rec/play/_/uOcLnFnkv2w2pZHYaP8rt4zOCFPqtiR9SomT0H93J6kQ6apgQf8DRyZUYRmzGdIAJFd12BSrvu1XxsPL.OMCsUpiLudKzoTjM\)](https://sjeccd-edu.zoom.us/rec/play/_/uOcLnFnkv2w2pZHYaP8rt4zOCFPqtiR9SomT0H93J6kQ6apgQf8DRyZUYRmzGdIAJFd12BSrvu1XxsPL.OMCsUpiLudKzoTjM)

Cascading Style Sheets

This week I want to talk more about Cascading Style Sheets (CSS).

Unlike HTML, which mixes data and formatting instructions into a single document, XML is all about data. An XML document contains no rules concerning how its data structure should be formatted and displayed in a browser or other software.

Cascading Style Sheets is a declarative language introduced in 1996 as a standard means of adding information to HTML documents about style properties such as fonts and borders. However, CSS also works great with XML.

There are actually three W3C recommended ways to apply formatting to an XML document:

- Cascading Style Sheets (CSS)
- Extensible Stylesheet Language Formatting Objects (XSL-FO) -- although it's now more-or-less no longer used due to CSS3
- Extensible Stylesheet Language Transformations (XSLT)

For now we will focus on Cascading Style Sheets.

The basic process is to create a set of formatting rules in a CSS stylesheet, link the stylesheet into

the XML document, and then open the XML document in a browser that processes XML and CSS. We can call CSS a browser-based styling language.

There are six important things to keep in mind when using Cascading Style Sheets:

1. CSS syntax looks much different than XML
2. The rule is the primary component of a CSS stylesheet
3. CSS properties can be inherited from parent to child elements
4. CSS allows descriptive comments
5. How CSS handles conflicting rules
6. Pseudo-elements offer greater control over formatting

Let's take a closer look at each of the above items.

CSS syntax is not XML

Unlike most of the technologies that work with XML documents, CSS has its own peculiar syntax.

Look at this sample stylesheet:

```
entree {
  background-color: silver;
  border-width: medium;
  border-style: solid;
  margin-bottom: .75em;
  padding: .25em;
}

name {
  color: white;
  font-weight: bold;
  background-color: red;
  border-width: thin;
  border-style: groove;
}
```

When you work with Cascading Style Sheets , you write instructions (*rules*) in a set of rules called a stylesheet. There are actually 3 different types of stylesheets -- external style sheets are a separate file, with an extension of .css; internal style sheets are found in the file itself (usually only in HTML files, not in XML files); inline style sheets are used in the element itself (again, usually only found in HTML files).

An external style sheet is just an ordinary text file that is read by the CSS processor, which applies the styling rules to the XML document for display. An external CSS stylesheet has a .css extension. Although case is a big deal to XML, CSS stylesheets are case-insensitive, except for the parts that reference external names. In the above example, the entree and name words in the stylesheet reference XML elements with lowercase names of entree and name.

In Cascading Style Sheets, the primary component of a stylesheet is a rule. A rule selects the elements to which you want to apply formatting and then specifies the formatting properties. A rule consists of two parts; the selector and the declaration:

- The selector identifies one or more XML elements that you want to format
- The declaration consists of a property-value pair that provides instructions on how to format the elements identified by the selector

The syntax differs, but the declaration section is similar to attributes (name-value pairs) of an XML element.

The general syntax of a rule is:

```
selector {property: value}
```

You can put a rule all on one line or spread it out over multiple lines, such as:

```
selector (  
    property: value;  
)
```

A rule can have multiple property-value pairs inside of curly brackets, each separated by a semicolon, such as:

```
selector {  
    property1: value1;  
    property2: value2;  
    property3: value3;  
}
```

Inheritance

When you apply CSS properties to your XML data, you need to be aware of the hierarchical, tree-like nature of an XML document. The reason for this is that some properties are inherited by a child from its parent.

Consider the following XML code:

```
<cast>
```

```
<character>George</character>
<character>Mary</character>
</cast>
```

Because font properties are inherited properties, suppose you set the font family of the cast element:

```
cast { font-family: "Courier New";}
```

Because font-family is an inheritable property, the character children automatically assume this property. If you didn't want the character elements to inherit this property, you would need to explicitly set an alternative, such as:

```
cast { font-family: "Courier New";}
character { font-family: Arial;}
```

Most CSS properties, such as background, margins, and borders, are not inherited from parent to child. However, font and many text-related properties are passed down the hierarchy.

Comments

Like XML documents, CSS supports comments, allowing you to add descriptive information in your stylesheet. Comments are identified by a `/*` at the start and a `*/` at the end, such as:

```
/* Here is a comment */
```

Often, I'll use extra `*` characters to make the comment more readable:

```
/******
  footer area styles
  *****/
```

Here is another example:

Given part of a poem in XML format as follows:

```
<POEM>
  <TITLE>Darest Thou Now O Soul</TITLE>
  <POET>Walt Whitman</POET>
  <STANZA>
    <VERSE>Darest thou now O soul,</VERSE>
    <VERSE>Walk out with me toward the unknown region,</VERSE>
```

```
<VERSE>Where neither ground is for the feet nor any path to follow</VERSE>
</STANZA>
</POEM>
```

with the CSS as follows:

```
POEM {display: block;}
TITLE {display: block; font-size: 16pt; font-weight: bold; }
POET {display: block; margin-bottom: 10px; }
STANZA {display: block; margin-bottom: 10px; }
VERSE {display: block; }
```

Lets look at the meaning of these five CSS rules:

- The first rule says that the contents of the POEM element should be displayed in a block by itself (**display: block**).
- The second rule says that the contents of the TITLE element should be displayed in a block by itself (**display: block**) in 16-point (**font-size: 16pt**) bold type (**font-weight: bold**).
- The third rule says that the POET element should be displayed in a block by itself (**display: block**) and should be set off from what follows it by 10 pixels (**margin-bottom: 10px**).
- The fourth rule is the same as the third rule except that it applies to STANZA elements.
- The fifth rule simply states that VERSE elements are also displayed in their own block.

Color values

CSS provides many ways to specify color; by name, by hexadecimal components, by integer components, and by percentages. Defining color by name is the simplest. CSS1 understands these 16 color names adopted from the Windows VGA palette:

- aqua
- black
- blue
- fuchsia
- gray
- green
- lime
- maroon
- navy
- olive
- purple
- red
- silver

- teal
- white
- yellow

In CSS3 (the latest version of CSS), we use a modified version of the X11 color chart. There are now over 200 named colors!

Wikipedia lists all of the color names:

https://en.wikipedia.org/wiki/X11_color_names ↗ https://en.wikipedia.org/wiki/X11_color_names

More Background (written originally for HTML and web programming, but useful information for XML coders, too...)

Why is it called CSS?

Cascading - Rules are interpreted in a top-to-bottom, cascading fashion. Rules that aren't conflicting are aggregated (e.g., **body { background: red; color: white; }** produces a background of red with a font color of white. For rules that DO conflict, the last one generally "wins" (e.g., **{background: red; background: white;}** produces a background of white, not red.

Style - Refers to the fact that CSS rules are used to describe the style (look and feel) of the web content. The basic idea is that we use XML to describe the information itself and to describe the structure of the content, but we use CSS to describe the look and feel. As an analogy, if we think of web content in terms of human clothing, we might say that if I'm wearing pants, a shirt, and a jacket, that the clothing would be the XML, but that the fact that my pants are black and made of cotton and my shirt is button-down and red, and my jacket is brown -- the brown, black, red, cotton, button-down - those would all be the style (CSS).

Sheet - Refers to the fact that CSS rules are written in a list of rules (sheet). In XML, all sheets must be separate files, and must be referred to using the following code:

```
<?xml-stylesheet type="text/css" href="filename.css"?>
```

Rules in CSS are defined as a **selector** and a **declaration**, which are made up of a **property** and

a value:

selector { property: value; }

There are basically 3 types of selectors (3 things you can "do" with CSS):

1. Redefine an element

```
firstname { background: pink;}
lastname { font-family: Verdana;}
```

2. Create your own style by using a class or ID

```
.class { property: value; }
#id { property: value;}

.highlight { background: #eee; }
#headerarea { border: thin dotted #000; }
```

3. "Advanced" "Stuff"

1. Multiple styles

- More than one style per selector, separated by a semi-colon:

person { color: purple; background: yellow;}

2. Multiple selectors

- More than one selector per style, separated by a comma:

lastname, firstname { font-style: italic;}

3. Contextual selectors

- Rule that only happens inside of another rule.
- In this example, only **em** tags inside of **li** tags inside of **ul** tags will be colored red. All other em tags would be their default color:

ul li em { color: red;}

- NOTE: Elements being modified by contextual selectors need not appear immediately inside one another. For example, using the style above with the XHTML below, blah would still be red text:**

** blah **

The child element can be anywhere along the parent-child relationship -- the great-grandparent-great-grandchild relationship still works in a contextual selector.

4. Direct child selectors

- Rule only happens when a child element is directly inside of the parent.
- In this example, only em tags inside of li tags would be red:

li > em { color: red;}

Using this CSS, this XHTML would produce red text: **red**

But this would not (although it would if it were a contextual selector and not a direct-child selector): **not red**

5. Adjacent selectors

- Adjacent selectors allow you to specify that something will change only when preceded by something else.

form + p { background: yellow;}

This would make the first paragraph after a form have the background of yellow.

6. Pseudo elements

- To affect the first letter of an element:

:first-letter

p:first-letter { font-size: 72px;}

- To affect the first line of an element:

:first-line

p:first-line { font-size: 72px;}

7. Pseudo classes

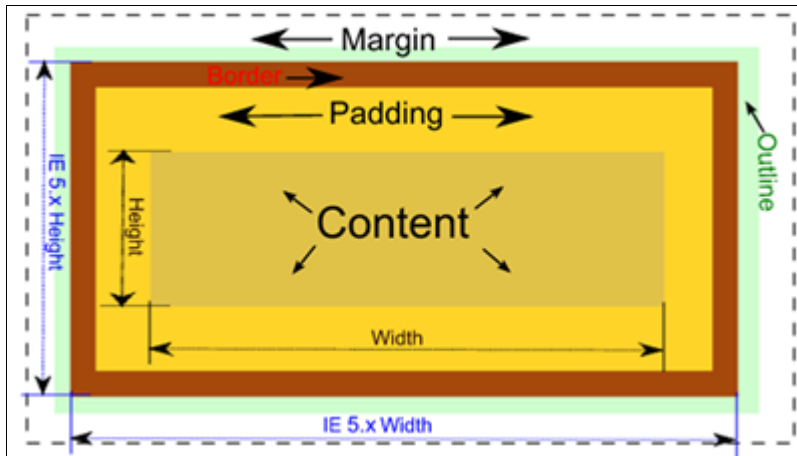
- To affect a link, use the style **a:link**
- To affect a visited link, use the style **a:visited**
- To affect an active link, use the style **a:active**
- To affect a hovered link, use the style **a:hover**

8. Generated content

- To add content before an element, use **:before**
h1:before { content: "Text coming before the h1 tag"; }
- To add content after an element, use **:after**
h1:after { content: "Text coming after the h1 tag";}

Box model

When a browser draws an object on a page, it places the object into an invisible rectangular space called a “bounding box.” In CSS, you can specify the size, look, and feel of the margins, the padding, the border, the outline, and the content of that bounding box.



Centering a box

- To make a block centered, you need to first give it a width, and then auto-margin the left and right margins:

```
main { width: 80%; margin: auto; }
```

Block vs. Inline

- Elements are either **block** or **inline**.
- Using CSS, you can change their default status:

```
em { display: block; }
```


```
p { display: inline; }
```

- You can even make them disappear:

```
.hidden { display: none; }
```

CSS3

In CSS3, there are a number of new selectors, new pseudo-elements, and new properties.

NOTE: Not all browsers currently support all of the CSS3 specifications. See <http://caniuse.com>  (<http://caniuse.com>) to check on the likelihood that your visitor will be able to see things the way you think they will!

New selectors

1. Selector siblings

◦ element1 ~ element 2

In this example, the first ul group is after a <div>; the second is after a <p>:

```
p ~ ul { background: green;}
```

In this example, Items 1, 2 and 3 in the div should not have a green background.
Items 1, 2, and 3 of both the paragraph and the H2 heading SHOULD have a green background.

There is nothing in this div. Nothing to see here...

- Item 1
- Item 2
- Item 3

A paragraph.

- Item 1
- Item 2
- Item 3

A H2 heading

- Item 1
- Item 2
- Item 3

2. Attributes

- text^ = matches only the first part of an attribute
- text\$ = matches only the last part of an attribute
- text* = matches any part of an attribute

In this example the div with the class "testy" will be green and italic; the div with the class "this-is-a-test" will be big and italic; the div with the class "this-test-fails" will only be italic.

```
div[class^="test"] { background: green;}
div[class$="test"] { font-size: 36px;}
div[class*="test"] { font-style: italic;}
```

This div has a class of "testy". It should have a green background and be italic but not 36 px.

This div has a class of "this-is-a-test". It should be size 36 and italic, but not green.

This div has a class of "this-test-fails". It should be only italic.

3. New pseudo-elements:

1. :first-of-type

:last-of-type

:only-of-type

```
li:first-of-type {background: green;}  
li:last-of-type {font-size: 24px;}  
li:only-of-type {font-style: italic;}
```

- Item 1 - green (it's the first list item in this UL)
- Item 2 - nothing cool
- Item 3 - font size 24px (it's the last item in this UL)

1. Sub-item - green (it's the first item in this OL); size 24 (it's also the last item of this OL); italic (it's the only item in this OL)

2. :first-child

:last-child

:only-child

```
li:first-child {background: green;}  
li:last-child {font-size: 24px;}  
li:only-child {font-style: italic;}
```


- Item 1 - green (it's the first list item in this UL)
- Item 2 - nothing cool
- Item 3 - font size 24px (it's the last item in this UL)

1. Sub-item - green (it's the first item in this OL); size 24 (it's also the last item of this OL); italic (it's the only item in this OL)

3. :empty

```
li:empty {background: green;}
```

Only the fourth item (with no content) has a green background:

- Item 1
- Item 2
- Item 3
- 

4. :target

```
div#clickme:target {background: green;}
```

[Click this link, and the div will go green.](#)



```
div#clickme:target {background: green;}
```

[Click this link, and the div will go green.](#)



5. :checked

```
:checked {width: 36px;}
```

☐ Item 1 ☐ Item 2 ☐ Item 3



```
:checked {width: 36px;}
```

☒ Item 1 ☐ Item 2 ☐ Item 3

6. :not

```
div *:not(p) em {color: green;}
```

This is green. `<div> ...</div>`

This is not green. `<div><p> ... </p> </div>`

This is green. `<div> <h2> ... </h2> </div>`

7. **::selection** (yes, there are two colons)

```
p::selection { background: pink;}  
p::-moz-selection { background: pink;}  
p::-webkit-selection { background: pink;}
```

Select this paragraph. It should go pink!

4. New Properties

1. **Animation** ➞ <http://web.stanford.edu/group/csp/cs22/css3/animation.html>)
2. **Transition** ➞ <http://web.stanford.edu/group/csp/cs22/css3/transition.html>)
3. **Transform** ➞ <http://web.stanford.edu/group/csp/cs22/css3/transform.html>)
4. **Background-clip** ➞ <http://web.stanford.edu/group/csp/cs22/css3/background-clip.html>)
5. **Background-gradient** ➞ <http://web.stanford.edu/group/csp/cs22/css3/background-gradient.html>)
6. **Rounded border** ➞ <http://web.stanford.edu/group/csp/cs22/css3/roundcorners.html>)
7. **Box shadow** ➞ <http://web.stanford.edu/group/csp/cs22/css3/boxshadow.html>)
8. **Color** ➞ <http://web.stanford.edu/group/csp/cs22/css3/color.html>)
9. **Font and Text shadow** ➞ <http://web.stanford.edu/group/csp/cs22/css3/font.html>)
10. **Multi-column layout** ➞ <http://web.stanford.edu/group/csp/cs22/css3/columns.html>)
11. **User interface -- appearance, box-sizing, outline effect, and resize** ➞ <http://web.stanford.edu/group/csp/cs22/css3/appearance.html>)
12. **Hyphens** ➞ <http://web.stanford.edu/group/csp/cs22/css3/hyphens.html>)

That's enough CSS1, CSS2, and CSS3 for now!