EECS 1012: Introduction to Computer Science

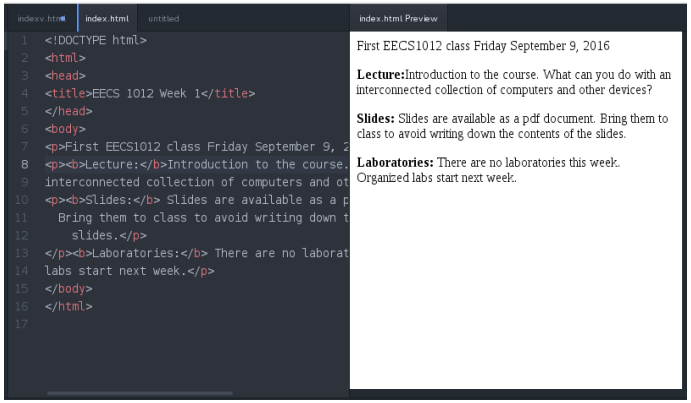
September 26, 2016

Validating CSS

- http://jigsaw.w3.org/css-validator
  - Much like the html validator, this will find bugs in your css that will take you forever to find.

A larger example

- Lets make a site for the course
- One directory per week
  - week1 through week12

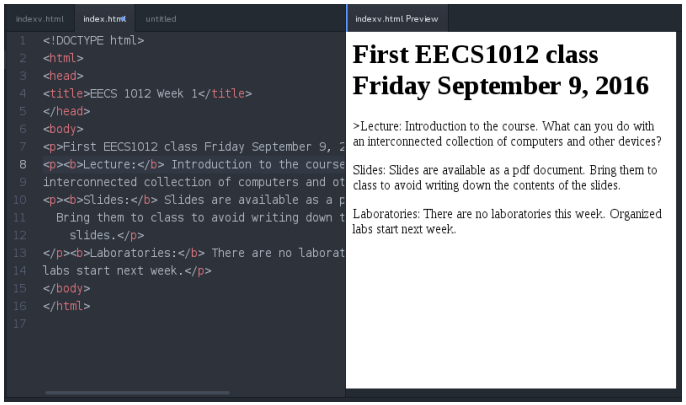


Lets not use <b> for bold

- … versus …
  - span for inline style, div for divisions (changes in page)
- Have many bold areas, define a class that makes things bold

Fonts

- font-weight - normal, bold, (other options)
  - See textbook for other options



FONTS

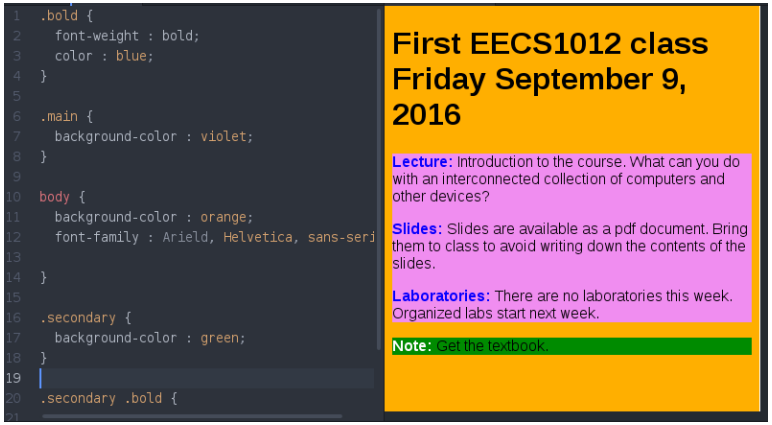
- Fonts are very complicated. There are thousands, not all supported on all platforms, etc.
- **Serif** and sans-serif are two classes. (Serif fonts have flourishes on characters, sans-serif do not.)
  - font-family: a, b, c, d;
    - try first one on the hardware. End with generic.
    - font-family: Arial, Helvetica, sans-serif;

Attached to body so that by default everything gets styled.

Now lets be more serious

- We will want one page per week
  - Same style on every page (share the style file)
- Want a top level course page
  - root (eecs1012)
    - index.html
    - style.css
    - week1/index.html
    - week2/index.html
    - and so on

Use to format regions



And add a top-level page

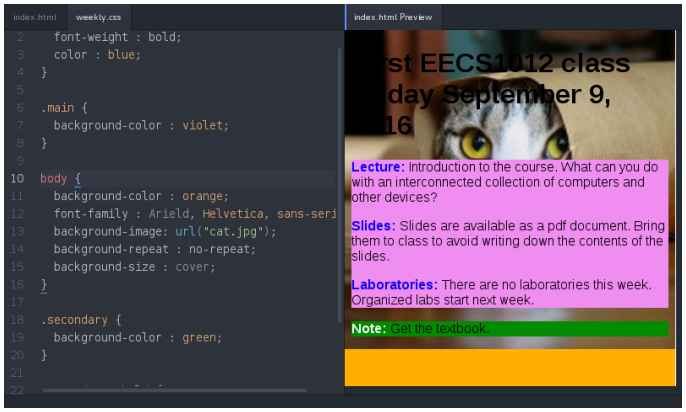
Format the main page

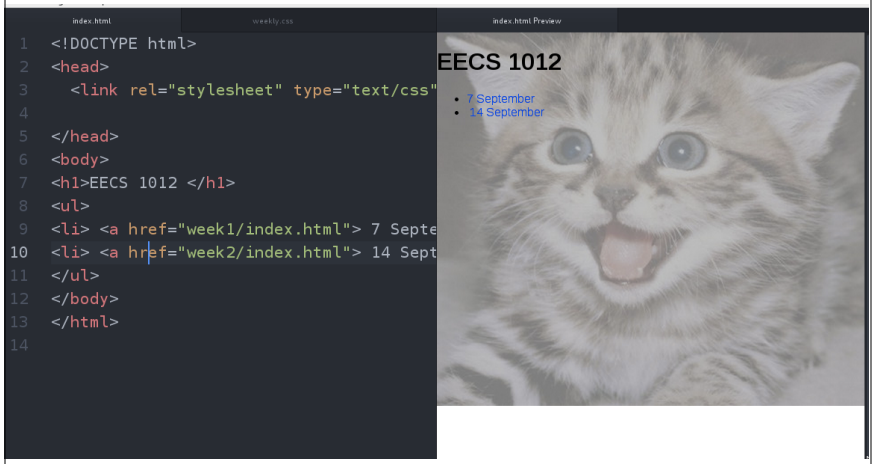
- Hyperlink formatting
  - a
  - a:hover, a:visited, a:decorated (others) specific for the state of the hyperlink

```
a:link {
  text-decoration: none;
}
```

Playing with global style

- Centring h1 ? Better ? WORSE?
- Change the base font ?
- Change the background colour
- Change the highlighted background colour
- Add a background image





Selectors

- .foo all elements with class="foo"
- #bar the element with id="bar"
- tree all <tree> elements
- tree,wheel all <tree> and <wheel> elements
- tree wheel all <wheel> within <tree>
- and many more

Cascading style sheets

- "Styles within styles"
  - Can have more than one style sheet, and in a style sheet more than one style rule may apply
- How does CSS choose which of competing style sheets to use?

Selector rules

- The most specific rule wins.

Selector Rules: Calculating Specificity

Style sheets can also override conflicting style sheets based on their level of specificity, where a more specific style will always win out over a less specific one. It is simply a counting game to calculate the specificity of a selector.

a. Count the number of **ID** attributes in the selector.
b. Count the number of **CLASS** attributes in the selector.
c. Count the number of HTML tag names in the selector.

Finally, write the three numbers in exact order with no spaces or commas to obtain a three digit number. (Note, you may need to convert the numbers to a larger base to end up with three digits.) The final list of numbers corresponding to selectors will easily determine specificity with the higher numbers winning out over lower numbers. Following is a list of selectors sorted by specificity:

```
#id1          {xxx} /* a=1 b=0 c=0 --> specificity = 100 */
UL UL LI.red  {xxx} /* a=0 b=1 c=3 --> specificity = 013 */
LI.red        {xxx} /* a=0 b=1 c=1 --> specificity = 011 */
LI            {xxx} /* a=0 b=0 c=1 --> specificity = 001 */
```

Order of Specification

To make it easy, when two rules have the same weight, the last rule specified wins.

To here

- We have static web pages
  - Content in HTML
  - Style in CSS
- Time to make them dynamic

JavaScript

- JavaScript enables us to make web pages ‘dynamic’ and ‘responsive’.
- Much JavaScript in web pages is ‘event driven’
  - You assign a callback (a function) to an event, and when the event happens your code is executed.
  - When your callback ends, your code is quiescent until another event occurs.

Style..

- As with CSS, you can mix JavaScript in the HTML page. (please don’t).
- You can use JavaScript to actually write the HTML page as it is loaded. (please don’t)
- Spaces between language tokens don’t count, so you can write pretty illegible code. (please don’t)

```
<head>
<script src=“myscript.js” type=“text/javascript”></script>
</head>
```

From HTML to JavaScript

- Certain HTML tags support user interaction (selection). Such as buttons
  - <button onclick=“myfunction()”>Click Me</button>
- When clicked invokes the myfunction in your JavaScript


<!DOCTYPE html>
<head>
<script src="myscript.js" type="text/java
</head>
<bod>
<button onclick="myfunction()">Click me</
</body>
</html>
Click me


function myfunction() {
alert("alert!");
}