HTML & CSS

SWE 432, Fall 2016
Design and Implementation of Software for the Web
HTML: HyperText Markup Language

• Language for describing *structure* of a document

• Denotes hierarchy of elements

• What might be elements in this document?
HTML History

• 1995: HTML 2.0. Published as standard with RFC 1866

• 1997: HTML 4.0  Standardized most modern HTML element w/ W3C recommendation
  • Encouraged use of CSS for styling elements over HTML attributes

• 2000: XHTML 1.0
  • Imposed stricter rules on HTML format
    • e.g., elements needed closing tag, attribute names in lowercase

• 2014: HTML5 published as W3C recommendation
  • New features for capturing more *semantic* information and *declarative* description of behavior
    • e.g., Input constraints
    • e.g., New tags that explain *purpose* of content
  • Important changes to DOM (will see these later….)


HTML Elements

\[ \text{“Start a paragraph element”} \]
Opening tag begins an HTML element. Opening tags must have a corresponding closing tag.

\[ \text{“Set the language to English”} \]
HTML attributes are name / value pairs that provide additional information about the contents of an element.

\[ \text{“End a paragraph element”} \]
Closing tag ends an HTML element. All content between the tags and the tags themselves compromise an HTML element.

- Open tag, close tag

\[ \text{<p lang=“en-us”>This is a paragraph in English. </p>} \]
<input type="text" />

“Begin and end input element”

Some HTML tags can be self closing, including a built-in closing tag.

<!-- This is a comment. Comments can be multiline. -->
A starter HTML document

“Use HTML5 standards mode”

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Hello World Site</title>
</head>
<body>
  Hello world!
</body>
</html>

“Header”
Information about the page

Hello world!

“Interpret bytes as UTF-8 characters”
Includes both ASCII & international characters.

“HTML content”

“Title”
Used by browser for title bar or tab.

“Document content”
Use `<h1>`, `<h2>`, ..., `<h5>` for headings

```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="main.css">
    <title>Prof Bell's Webpage</title>
  </head>
  <body>
    Prof Jonathan Bell
  </body>
</html>
```

Prof Jonathan Bell

This is Prof Bell’s ACTUAL homepage from 1999!

Some funny links

- Homestar Runner
- Hamster Dance

About Prof Bell

Prof Bell’s office is at 4422 Engineering Building. His email address is <a href="mailto:bellj@gmu.edu">bellj@gmu.edu</a>

Last updated: September 4th, 1999

https://seecode.run/#-KQgR7vG9Ds7IUJS1kdq
HTML Example

```html
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="main.css">
    <title>Prof Bell's Webpage</title>
</head>
<body>
    <h1>Prof Jonathan Bell</h1>
    <p><img alt="My really cool laptop" src="http://www.wonder-tonic.com/geocitiesizer/images/laptop-01.gif" />
        This is Prof Bell's ACTUAL homepage from 1999!
    </p>
    <h2>Welcome, students!</h2>
    <p><a href="https://www.youtube.com/watch?v=d0w4w9WgXcQ">See how to make this page</a></p>
    <h2>Some funny links</h2>
    <ul>
        <li><a href="http://www.homestarunner.com">Homestar Runner</a></li>
        <li><a href="http://www.wbjw.net/The%20Original%20Hamsterdance.htm">Hamster Dance</a></li>
    </ul>
    <h3>About Prof Bell</h3>
    <p>Prof Bell's office is at 4422 Engineering Building. His email address is bellj@gmu.edu.</p>
</body>
</html>
```

Paragraphs (`<p>`) consist of related content. By default, each paragraph starts on a new line.
Unordered lists (\texttt{<ul>}) consist of list items (\texttt{<li>}) that each start on a new line. Lists can be nested arbitrarily deep.
<h1>Level 1 Heading</h1>

<h2>Level 2 Heading</h2>

<h3>Level 3 Heading</h3>

<h4>Level 4 Heading</h4>

<h5>Level 5 Heading</h5>

<h6>Level 6 Heading</h6>

Text can be made <b>bold</b> and <i>italic</i>, or <sup>super</sup> and <sub>sub</sub> scripts. White space collapsing removes all sequences of two more spaces and line breaks, allowing the markup to use tabs and whitespace for organization.

Spaces can be added with &nbsp; &nbsp; &nbsp;.

New lines can be added with &lt;BR/&gt;.

A paragraph consists of one or more sentences that form a self-contained unit of discourse. By default, a browser will show each paragraph on a new line.

Text can also be offest with horizontal rules.

Text can also be offest with horizontal rules.
Semantic markup

• Tags that can be used to denote the *meaning* of specific content

• Examples

  • `<strong>` An element that has importance.
  • `<blockquote>` An element that is a longer quote.
  • `<q>` A shorter quote inline in paragraph.
  • `<abbr>` Abbreviation
  • `<cite>` Reference to a work.
  • `<dfn>` The definition of a term.
  • `<address>` Contact information.
  • `<ins>` Content that was inserted or deleted.
  • `<del>` Content that is no longer accurate.
  • `<s>` Something that is no longer accurate.
Links

<a href="http://www.google.com">Absolute link</a><br />
<a href="movies.html">Relative URL</a><br />
<a href="mailto:tlatoza@gmu.edu">Email Prof. LaToza</a><br />
<a href="http://www.google.com" target="_blank">Opens in new window</a><br />
<a href="#idName">Navigate to HTML element idName</a>
Images, Audio, Video

• HTML includes standard support for `<img>`, `<audio>`, `<video>`

• Common file formats
  • Images: .png, .gif, .jpg
  • Audio: .mp3
  • Video: .mp4

Important attributes for `<video>`
- `src` - location of video
- `autoplay` - tells browser to start play
- `controls` - show the default controls
- `poster` - image to show while loading
- `loop` - loop the video
- `muted` - mutes the audio from the video
Tables

```
<table>
    <tr>
        <th>Monday</th>
        <th>Tuesday</th>
        <th>Wednesday</th>
    </tr>
    <tr>
        <th>1pm - 2pm</th>
        <td rowspan="2">Intro Physics</td>
        <td>Calculus 2</td>
        <td>Free</td>
    </tr>
    <tr>
        <th>2pm - 3pm</th>
        <td>Free</td>
        <td>Psychology</td>
    </tr>
</table>
```
Forms

```html
<form action="http://www.server.com" method="post">
  <input type="text" name="username" value="" />
  <input type="submit" />
</form>
```

“Send the results to www.server.com”

Action attribute should be omitted if not using form to submit data.

“Send form data in HTTP headers”

Method specifies how data is transmitted to server. method="get" sends data appended to URL.

- Elements located in a form may have name and value attributes. This data is used in submission to server.

  - Note: name is used for a very different purpose than id.

- Controls may (or may not) be enclosed in a form.

  - If not form submission mechanism to submit data to server, no need for form.
Controls

Text Input: <input type="text" maxlength="5" />
Password Input: <input type="password" />
Search Input: <input type="search" />
Text Area: <textarea>Initial text</textarea>
Checkbox:
  <input type="checkbox" checked="checked" /> Checked
  <input type="checkbox" /> Unchecked

Drop Down List Box:
<select>
  <option>Option1</option>
  <option selected="selected">Option2</option>
</select>

Multiple Select Box:
<select multiple="multiple">
  <option>Option1</option>
  <option selected="selected">Option2</option>
</select>

File Input Box: <input type="file" />
Image Button: <input type="image" src="http://cs.gmu.edu/~tlatoza/images/reachabilityQuestion.jpg" width="50"/>
Button: <button>Button</button>
Range Input: <input type="range" min="0" max="100" step="10" value="30" />

Search input provides clear button
Specialized controls

- `<input type="date" />
- `<input type="time" />
- `<input type="datetime-local" />

- `<input type="color" />
- `<input type="number" min="0" max="50"/>`
Labeling input

• Can place suggested input or prompt *inside* input element

```html
<p>Input box: <input type="text" placeholder="Enter keyword" /></p>
```

• Disappears after user types

```html
<p>Input box: <input type="text" placeholder="Enter keyword" /></p>
```

• Label attaches a label *and* expands the clickable region of control, making form easier to use

```html
<p><label>Label on input box: <input type="text" /></label></p>
```

**Clickable region**
Validating input

- Displays errors on invalid input *immediately*, making it easier to fix errors
- Check that input is a valid email
  
  ```html
  <p><label>Email: <input type="email" /></label></p>
  Email: 
  ```
- Check that input is a valid URL
  
  ```html
  <p><label>URL: <input type="url" /></label></p>
  URL: 
  ```
- Check that input matches regex pattern
  
  ```html
  <p><label>Would you like an apple or orange? <input type="text" pattern="apple|orange" /></label></p>
  Would you like an apple or orange? 
  ```
- Constrain input to be at most `maxlength`
  
  ```html
  <p><label>Enter a username up to 10 characters: <input type="text" maxlength=10 /></label></p>
  Enter a username up to 10 characters: 
  ```
- Prevent all edits
  
  ```html
  <p><label>Autogenerated text <input type="text" readonly="true" /></label></p>
  Autogenerated text 
  ```
Block vs. Inline Elements

Block elements

Block elements appear on a new line. Examples: `<h1><p><li><table><form>`

```
<h1>Hiroshi Sugimoto</h1>
<p>The dates for the ORIGIN OF ART exhibition are as follows:</p>
<ul>
    <li>Science: 21 Nov - 20 Feb 2010/2011</li>
</ul>
```

Hiroshi Sugimoto

The dates for the ORIGIN OF ART exhibition are as follows:

- Science: 21 Nov - 20 Feb 2010/2011

Inline elements

Inline elements appear to continue on the same line. Examples: `<a><b><input><img>`

```
Timed to a single revolution of the planet around the sun at a 23.4 degrees tilt that plays out the rhythm of the seasons, this *Origins of Art* cycle is organized around four themes: *science, architecture, history*, and *religion*.
```

Timed to a single revolution of the planet around the sun at a 23.4 degrees tilt that plays out the rhythm of the seasons, this *Origins of Art* cycle is organized around four themes: *science, architecture, history*, and *religion*.
Grouping elements

- Creates a **parent** or **container** element and a set of **child** elements
- Enables group to be styled together
- Can use any block or inline element or **generic** element
  - `<div>` is the generic block element
  - `<span>` is the generic inline element
- Semantic layout elements are block elements that associate meaning with group
  - Very useful for CSS selectors (coming soon)

```html
<body>
  <header>
    <h1>How to Get a PhD</h1>
    <nav>...</nav>
  </header>
  <article>
    <section>
      <figure><img src="benfranklin.jpg"></figure>
      <h3>Bribing your Committee</h3>
      <p>When blackmail fails...</p>
    </section>
    <aside>
      <h4>Useful Links</h4>
      <a href="www.bevmo.com">Research Supplies</a>
    </aside>
  </article>
</body>
```

Some popular semantic layout elements:
- `<header>`
- `<footer>`
- `<nav>`
- `<article>`
- `<aside>`
- `<section>`
- `<figcaption>`
HTML Style

• Tags
  • Use lowercase for names
  • Use indentation to reflect hierarchy
  • Always close tags
    • Or use self-closing tags <tagname /> notation
  • Use attributename="value" format for attributes
  • Use blank lines to break up documents into closely connected regions
  • Use comments to describe purpose of regions
HTML Best Practices

• Use specialized controls or input validation where applicable

• Always include elements of HTML starter document

• Use label or placeholder for labeling controls

• Use alt to make images accessible
In Class Exercise

• Form two person groups.
  • Build a simple personal website.
  • It should use basic HTML elements such as `<img>`, `<a>`, `<div>`, semantic tags.
  • Use a pastebin such as seecode.run or jsbin.com

• When you are done, log in to Socrative, post link to your pastebin.
CSS: Cascading Style Sheets

- Language for styling documents

```
p {
  font-family: Arial;
}
```

“Select all <p> elements”
Selector describes a set of HTML elements

“Use Arial font family”
Declaration indicates how selected elements should be styled.

- Separates visual presentation (CSS) from document structure (HTML)
  - Enables changes to one or the other.
  - Enables styles to be reused across sets of elements.
CSS History

• 1994: Cascading HTML style sheets—a proposal
  • Hakon W Lie proposes CSS
  • Working w/ Tim-Berners Lee at CERN
• 1996: CSS1 standard, recommended by W3C
  • Defines basic styling elements like font, color, alignment, margin, padding, etc.
• 1998: CSS2 standard, recommended by W3C
  • Adds positioning schemes, z-index, new font properties
• 2011: CSS3 standards divided into modules, begin adoption
  • Add more powerful selectors, more powerful attributes

https://dev.opera.com/articles/css-twenty-years-hakon/
https://en.wikipedia.org/wiki/Cascading_Style_Sheets#History
CSS Styling

• Invisible box around every element.

• Rules control how sets of boxes and their contents are presented

Example Styles

**BOXES**
- Width, height
- Borders (color, width, style)
- Position in the browser window

**TEXT**
- Typeface
- Size, color
- Italics, bold, lowercase
Using CSS

External CSS

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="main.css">
  <title>Prof Bell's Webpage</title>
</head>
</html>
```

Internal CSS

```html
<!DOCTYPE html>
<html>
<head>
  <title>Prof Bell's Webpage</title>
  <style type="text/css">
    body {
      background-image: url("bluerock.jpg");
      font-family: Comic Sans MS, Comic Sans;
      color: #FFFFFF;
    }
  </style>
</head>
</html>
```

- External CSS enables stylesheets to be reused across multiple files
- Can include CSS files
- Can nest CSS files
- `@import url("file.css")` imports a CSS file in a CSS file
CSS Type Selectors

• What if we wanted more green?

```css
h2, h3 {
  color: LightGreen;
}
```

“Select all `<h2>` and `<h3>` elements”

Type selector selects one or more element types.

```css
* {
  color: LightGreen;
}
```

“Select all elements”

Universal selector selects all elements.

---

“Select all `<h2>` and `<h3>` elements”

Type selector selects one or more element types.

```css
h2, h3 {
  color: LightGreen;
}
```

“Select all elements”

Universal selector selects all elements.

---

“Select all `<h2>` and `<h3>` elements”

Type selector selects one or more element types.

```css
h2, h3 {
  color: LightGreen;
}
```

“Select all elements”

Universal selector selects all elements.
CSS Class Selectors

```
<img src="profilePic.jpg" class="imageLarge" />
```

“Label `<img>` element with `imageLarge` class”

```
.imageLarge {
  width: 200px;
  height: 200px;
}
```

“Define class `imageLarge`.”

```
<img src="profilePic.jpg" class="large transparent" />
```

```
img.large {
  width: 200px;
  height: 200px;
}
```

“Define large class that applies only to `<img>` elements”

```
.transparent {
  opacity: .50;
}
```

“Define transparent class”

- Classes enable the creation of sets of elements that can be styled in the same way.
CSS id selectors

• Advantages
  • Control presentation of individual elements

• Disadvantages
  • Must write separate rule for each element

```html
<div id="exampleElem">
  Some text
</div>
```

```css
#exampleElem {
  font-weight: bold;
}
```
### Additional selector types

<table>
<thead>
<tr>
<th>Selector</th>
<th>Meaning</th>
<th>Example</th>
<th>Example Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descendant selector</td>
<td>Matches all descendants of an element</td>
<td>p a {}</td>
<td>Select <code>&lt;a&gt;</code> elements inside <code>&lt;p&gt;</code> elements</td>
</tr>
<tr>
<td>Child selector</td>
<td>Matches a direct child of an element</td>
<td>h1&gt;a {}</td>
<td>Select <code>&lt;a&gt;</code> elements that are directly contained by <code>&lt;h1&gt;</code> elements.</td>
</tr>
<tr>
<td>First child selector</td>
<td>Matches the first child of an element</td>
<td>h1:first-child {}</td>
<td>Select the the elements that are the first child of a <code>&lt;h1&gt;</code> element.</td>
</tr>
<tr>
<td>Adjacent selector</td>
<td>Matches selector</td>
<td>h1+p {}</td>
<td>Selects the first <code>&lt;p&gt;</code> element after any <code>&lt;h1&gt;</code> element</td>
</tr>
<tr>
<td>Negation selector</td>
<td>Selects all elements that are not selected.</td>
<td>body *:not(p)</td>
<td>Select all elements in the body that are not <code>&lt;p&gt;</code> elements.</td>
</tr>
<tr>
<td>Attribute selector</td>
<td>Selects all elements that define a specific attribute.</td>
<td>input[invalid]</td>
<td>Select all <code>&lt;input&gt;</code> elements that have the invalid attribute.</td>
</tr>
<tr>
<td>Equality attribute</td>
<td>Select all elements with a specific attribute value</td>
<td>p[class=&quot;invisible&quot;]</td>
<td>Select all <code>&lt;p&gt;</code> elements that have the invisible class.</td>
</tr>
</tbody>
</table>
CSS Selectors

• Key principles in designing effective styling rules
  
  • Use classes, semantic tags to create sets of elements that share a similar rules
  
  • Don’t repeat yourself (DRY)
    
    • Rather than create many identical or similar rules, apply single rule to all similar elements
  
  • Match based on semantic properties, not styling
    
    • Matching elements based on their pre-existing styling is fragile
Cascading selectors

• What happens if more than one rule applies?

• Most *specific* rule takes precedence
  • \( p b \) is more specific than \( p \)
  • \#maximizeButton is more specific than button

• If otherwise the same, *last* rule wins

• Enables writing generic rules that apply to many elements that are overridden by specific rules applying to a few elements
CSS inheritance

• When an element is contained inside another element, some styling properties are inherited
  • e.g., font-family, color

• Some properties are not inherited
  • e.g., background-color, border

• Can force many properties to inherit value from parent using the inherit value
  • e.g., padding: inherit;
Exercise - What is selected?

1. \[ \text{div.menu-bar ul ul} \{
\text{display: none;}
\}
\]

2. \[ \text{div.menu-bar li:hover > ul} \{
\text{display: block;}
\}
\]

ul: unordered list
li: list element
Pseudo classes

- Classes that are automatically attached to elements based on their attributes.

```
.invisible {
    display: none;
}

input:invalid {
    border: 2px solid red;
}

input:invalid + div {
    display: block;
}

input:focus + div {
    display: none;
}
```

"Select elements with the invalid attribute."

"Select elements that have focus."

- Classes that are automatically attached to elements based on their attributes.
Examples of pseudo classes

• :active - elements activated by user. For mouse clicks, occurs between mouse down and mouse up.
• :checked - radio, checkbox, option elements that are checked by user
• :disabled - elements that can’t receive focus
• :empty - elements with no children
• :focus - element that currently has the focus
• :hover - elements that are currently hovered over by mouse
• :invalid - elements that are currently invalid
• :link - link element that has not yet been visited
• :visited - link element that has been visited
Color

• Can set text color (color) and background color (background-color)

• Several ways to describe color
  • six digit hex code (e.g., #ee3e80)
  • color names: 147 predefined names
  • rgb(red, green, blue): amount of red, green, and blue
  • hsla(hue, saturation, lightness, alpha): alternative scheme for describing colors

• Can set opacity (opacity) from 0.0 to 1.0
Typefaces

Serif
Sans-Serif
Monospace
Cursive

font-family: Georgia, Times, serif;

“Use Georgia if available, otherwise Times, otherwise any serif font”.

font-family enables the typeface to be specified. The typeface must be installed. Lists of fonts enable a browser to select an alternative.
# Type scales

## Pixels

<table>
<thead>
<tr>
<th>12 pixel scale</th>
<th>16 pixel scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1 24px</td>
<td>h1 32px</td>
</tr>
<tr>
<td>h2 18px</td>
<td>h2 24px</td>
</tr>
<tr>
<td>h3 14px</td>
<td>h3 18px</td>
</tr>
<tr>
<td>body 12px</td>
<td>body 16px</td>
</tr>
</tbody>
</table>

## Percentages

<table>
<thead>
<tr>
<th>12 pixel scale</th>
<th>16 pixel scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1 200%</td>
<td>h1 200%</td>
</tr>
<tr>
<td>h2 150%</td>
<td>h2 150%</td>
</tr>
<tr>
<td>h3 117%</td>
<td>h3 112.5%</td>
</tr>
<tr>
<td>body 75%</td>
<td>body 100%</td>
</tr>
</tbody>
</table>

Sets absolute font size.

Sets font size relative to default text size (16px)
Styling text

h2 {
    text-transform: uppercase;
    text-decoration: underline;
    letter-spacing: 0.2em;
    text-align: center;
    line-height: 2em;
    vertical-align: middle;
    text-shadow: 1px 1px 0 #666666;
}

- text-transform: uppercase, lowercase, capitalize
- text-decoration: none, underline, overline, line-through, blink
- letter-spacing: space between letters (kerning)
- text-align: left, right, center, justify
- line-height: total of font height and empty space between lines
- vertical-align: top, middle, bottom, …
- text-shadow: [x offset][y offset][blur offset][color]
Cursor

• Can change the default cursor with cursor attribute
  
  • auto, crosshair, pointer, move, text, wait, help, url(“cursor.gif”)
  
• Should only do this if action being taken clearly matches cursor type
Boxes, by default, are sized *just* large enough to fit their contents.

- Can specify sizes using px or %
  - % values are relative to the container dimensions

- margin: 10px 5px 10px 5px; (clockwise order - [top] [right] [bottom] [left])

- border: 3px dotted #0088dd; ([width] [style] [color])
  - style may be solid, dotted, dashed, double, groove, ridge, inset, outset, hidden / none
Centering content

• How do you center an element inside a container?

• Step 1: Must first ensure that element is *narrower* than container.
  • By default, element will expand to fill entire container.
  • So must usually explicitly set width for element.

• Step 2: Use *auto* value for left and right to create equal gaps
Visibility and layout

• Can force elements to be inline or block element.
  • display: inline
  • display: block

• Can cause element to not be laid out or take up any space
  • display: none

  • Very useful for content that is dynamically added and removed.

• Can cause boxes to be invisible, but still take up space
  • visibility: hidden;
Positioning schemes

Normal flow (default)

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Block level elements appear on a new line. Even if there is space, boxes will not appear next to each other.

Relative positioning

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

p.example {
    position:relative;
    top: 10px;
    left: 100px;
}

Element shifted from normal flow. Position of other elements is not affected.

Absolute positioning

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

h3 {
    position: absolute;
    background-color: LightGray;
    left: 350px;
    width: 250px;
}

Element taken out of normal flow and does not affect position of other elements. Moves as user scrolls.

Fixed positioning

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

h3 {
    position: fixed;
    background-color: LightGray;
    left: 40px;
    width: 250px;
}

Element taken out of normal flow and does not affect position of other elements. Fixed in window position as user scrolls.

Floating elements

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

h3 {
    float: left;
    background-color: LightGray;
    left: 40px;
    width: 250px;
}

Element taken out of normal flow and position to far left or right of container. Element becomes block element that others flow around.
Stacking elements

- Elements taken out of normal flow may be stacked on top of each other
- Can set order with z-index property
  - Higher numbers appear in front
- Can set opacity of element, making occluded elements partially visible
Transform - examples

- Can modify coordinate space of element to rotate, skew, distort
Transitions

- transition: [property time], …, [property time]

  - When new class is applied, specifies the time it will take for each property to change

  - Can use all to select all changed properties
Fixed width vs. liquid layouts

• Fixed width
  • Use width="[num]px" to force specific sizes
  • Allows for tightest control of look and feel
  • But can end up with extra whitespace around edge of web page

• Liquid layout
  • Use width="[num]%" to size relative to container sizes
  • Pages expand to fill the entire container size

• Problems
  • Wide windows may create long lines of text can be difficult to read
  • Very narrow windows may squash words, breaking text onto many lines

• (Partial) solution
  • Can use min-width, min-height, max-width, max-height to set bounds on sizes
Designing for mobile devices

• Different devices have different aspect ratios.

• Important to test for different device sizes.

• May sometimes build alternative layouts for different device sizes.

• Using specialized controls important.

• Enables mobile browsers to use custom device-specific widgets that may be much easier to use.
CSS Preprocessors

- Languages extend CSS to offer more powerful ways to specify rules (e.g., LESS, SASS)
- Make rules more compact and less redundant
- Examples from LESS:

**Variables**

```less
@nice-blue: #5B83AD;
@light-blue: @nice-blue + #111;

#header {
  color: @light-blue;
}
```

Compiles to

```css
#header {
  color: #6c94be;
}
```

**Mixins**

```less
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}
```

```css
#header {
  color: black;
}

#navigation {
  font-size: 12px;
}

.logo {
  width: 300px;
}
```

**Nested rules**

```less
.menu a {
  color: #111;
  .bordered;
}
```

```less
.menu a {
  color: red;
  .bordered;
}
```

http://lesscss.org/features/
CSS Best Practices

• When possible, use CSS to declaratively describe behavior rather than code

  • Easier to read, can be optimized more effectively by browser

• Don’t repeat yourself (DRY)

  • Rather than duplicating rules, create selectors to style all related elements with single rule

• CSS should be readable

  • Use organization, indentation, meaningful identifiers, etc.
Supplementary Materials

• Tutorials, reference materials, and examples
  • CSS: https://developer.mozilla.org/en-US/docs/Web/CSS
  • Pastebin for experimenting with HTML & CSS
    • https://seecode.run
In Class Activity

Style the website that you made at the beginning.

You should try to use CSS features such as

• class selectors, id selectors, descendent selectors, etc.

• pseudo classes

• alternative positioning schemes