HTML & CSS

SWE 432, Fall 2017
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

<p lang="en-us">This is a paragraph in English.</p>

“Start a paragraph element”
Opening tag begins an HTML element. Opening tags must have a corresponding closing tag.

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

“End a paragraph element”
Closing tag ends an HTML element. All content between the tags and the tags themselves compromise an HTML element.
HTML Elements

<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>
```

“HTML content”

“Title”

Used by browser for title bar or tab.

“Header”

Information about the page

“Interpret bytes as UTF-8 characters”

Includes both ASCII & international characters.
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>
  <h1>Prof Jonathan Bell</h1>
  Welcome, students!</h2>
  See how to make this page.</p>
  Some funny links
  <ul>
    <li><a href="http://www.homestarrunner.com">Homestar Runner</a></li>
    <li><a href="http://www.wbjw.net/The%20Original%20Hamsterdance.htm">Hamster Dance</a></li>
  </ul>
  About Prof Bell
  Prof Bell's office is at 4422 Engineering Building. His email address is bellj@gmu.edu.
  Last updated: September 4th, 1999
</body>
</html>
```

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>
  Prof Jonathan Bell
  <div>
    <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>
  </div>
  <h2>Welcome, students!</h2>
  <p>
    See how to make this page<br>
  </p>
  <h2>Some funny links</h2>
  <ul>
    <li><a href="http://www.homestarunner.com">Homestar Runner</a></li>
    <li><a href="http://www.wb3w.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 (<ul>) consist of list items (<li>) that each start on a new line. Lists can be nested arbitrarily deep.

https://seecode.run/#-KQgR7vG9Ds7IUJS1kdq
Text

Level 1 Heading

Level 2 Heading

Level 3 Heading

Level 4 Heading

Level 5 Heading

Level 5 Heading

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;.

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

<p>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.</p>

<hr/>

Text can also be offset with horizontal rules.

Text can also be offset 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><del> Content that was inserted or deleted.
  • <s> Something that is no longer accurate.
Links

- [Absolute link](http://www.google.com)
- [Relative URL](movies.html)
- [Email Prof. LaToza](mailto:tlatoza@gmu.edu)
- [Opens in new window](http://www.google.com)
- [Navigate to HTML element idName](#idName)
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>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>1pm - 2pm</td>
<td>Intro Physics</td>
<td>Calculus 2</td>
<td>Free</td>
</tr>
<tr>
<td>2pm - 3pm</td>
<td>Free</td>
<td>Psychology</td>
<td>Free</td>
</tr>
</tbody>
</table>
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" />"
Specialized controls

- `<input type="date" />
  11/dd/yyyy
  September 2016
  Sun  Mon  Tue  Wed  Thu  Fri  Sat
  28  29  30  31  1   2   3
  4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  1`

- `<input type="time" />
  11:58 PM
  September 2016
  Sun  Mon  Tue  Wed  Thu  Fri  Sat
  28  29  30  31  1   2   3
  4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  1`

- `<input type="datetime-local" />
  mm/dd/yyyy, --:-- AM
  September 2016
  Sun  Mon  Tue  Wed  Thu  Fri  Sat
  28  29  30  31  1   2   3
  4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  1`

- `<input type="color" />
  Cyan`

- `<input type="number" min="0" max="50" />
  30`
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>
```

```html
Label on input box: 
```

Clickable region
Validating input

• Displays errors on invalid input *immediately*, making it easier to fix errors

• Check that input is a valid email

  `<p><label>Email: <input type="email" /></label></p>`

  Email: 

• Check that input is a valid URL

  `<p><label>URL: <input type="url" /></label></p>`

  URL: 

• Check that input matches regex pattern

  `<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`

  `<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

  `<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>`

---

**Inline elements**

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

---

**Hiroshi Sugimoto**

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

- Science: 21 Nov - 20 Feb 2010/2011

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
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

<table>
<thead>
<tr>
<th>BOXES</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width, height</td>
<td>Typeface</td>
</tr>
<tr>
<td>Borders (color, width, style)</td>
<td>Size, color</td>
</tr>
<tr>
<td>Position in the browser window</td>
<td>Italics, bold, lowercase</td>
</tr>
</tbody>
</table>
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>
  <style type="text/css">
    body {
      background-image: url("bluerock.jpg");
      font-family: Comic Sans MS, Comic Sans;
      color: #FFFF00;
    }
  </style>
  <title>Prof Bell's Webpage</title>
</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?

h2, h3 {
  color: LightGreen;
}

“Select all <h2> and <h3> elements”

Type selector selects one or more element types.

* {
  color: LightGreen;
}

“Select all elements”

Universal selector selects all elements.
CSS Class Selectors

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

“Define class imageLarge.”

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

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

```
.img.large { transparency: .50; }
```

“Define transparent class”

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

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

- Advantages
  - Control presentation of individual elements
- Disadvantages
  - Must write separate rule for *each* element
## Additional selector types

<table>
<thead>
<tr>
<th>Selector</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Descendant selector</strong></td>
<td>Matches all descendants of an element</td>
<td>p a {}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select &lt;a&gt; elements inside &lt;p&gt; elements</td>
</tr>
<tr>
<td><strong>Child selector</strong></td>
<td>Matches a direct child of an element</td>
<td>h1&gt;a {}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select &lt;a&gt; elements that are directly contained by &lt;h1&gt; elements.</td>
</tr>
<tr>
<td><strong>First child selector</strong></td>
<td>Matches the first child of an element</td>
<td>h1:first-child {}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select the the elements that are the first child of a &lt;h1&gt; element.</td>
</tr>
<tr>
<td><strong>Adjacent selector</strong></td>
<td>Matches selector</td>
<td>h1+p {}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Selects the first &lt;p&gt; element after any &lt;h1&gt; element</td>
</tr>
<tr>
<td><strong>Negation selector</strong></td>
<td>Selects all elements that are not selected.</td>
<td>body *:not(p)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select all elements in the body that are not &lt;p&gt; elements.</td>
</tr>
<tr>
<td><strong>Attribute selector</strong></td>
<td>Selects all elements that define a specific attribute.</td>
<td>input[invalid]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select all &lt;input&gt; elements that have the invalid attribute.</td>
</tr>
<tr>
<td><strong>Equality attribute selector</strong></td>
<td>Select all elements with a specific attribute value</td>
<td>p[class=&quot;invisible&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Select all &lt;p&gt; 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. ```
   div.menu-bar ul ul {
     display: none;
   }
```

2. ```
   div.menu-bar li:hover > ul {
     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.”

“Select elements that have focus.”
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

```css
body {
  color: Red;
  background-color: rgb(200, 200, 200);
}

h1 {
  background-color: DarkCyan;
}

h2 {
  color: #ee3e80;
}

p {
  color: hsla(0, 100%, 100%, 0.5);
}

div.overlay {
  opacity: 0.5;
}
```
## Typefaces

<table>
<thead>
<tr>
<th>Serif</th>
<th>Sans-Serif</th>
<th>Monospace</th>
<th>Cursive</th>
</tr>
</thead>
</table>

```
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.
Styling text

```css
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