Info Dump

Fast-paced intro to HTML & CSS

- A smidge of JS
- If new to HTML/CSS
    - VERY FAST and Shallow
        - Follow Readings+Resources to get more
    - Important highlights
- If experienced w/HTML+CSS
    - Pay attention
        - Details I care about are emphasized

What is HTML

- H - Hyper
- T - Text
- M - Markup
- L - Language

In other works, text that can link to other text, with "markup" in it to apply non-textual details.

```
This has a <a href="other-file.html">link</a> to another file
```

This has a **link** to another file

The Trinity of the Web

- HTML - The *structured content* of the page
    - WITHOUT regard to appearance
- CSS - The appearance of the content
    - Defined by structure
- JS - Interactions with the content
    - Other than navigation

Browser Rendering an HTML Page

- Figure out size and visual properties
    - of every element "box"
- Download CSS/images/etc files
    - as references encountered
- Applying those files
    - updating the sizing and visuals as needed
- Downloading JS as encountered
    - Run that JS
    - modify the output-to-render as needed

Semantic HTML

HTML is the structured content of the page

Think an organized list of everything in the page

- Like an outline, but with the text

You can try to use HTML for looks

- But that will fail
- Devices (mobile, desktop, versions) work diff
- Browsers show things differently
    - How does a paragraph, button, list look?

What does "Semantic" mean?

"related to meaning"

Several words

- a paragraph?
- a heading?
- an item in a list?

It might be part of a navigation, or a section, or a link.

But these aren't APPEARANCE related.

Don't say where they appear or what they look like.

HTML Tags

- the start/end indicators: "tags"
- indicators + content: "element"

"tags"/"elements" are often used interchangeably

- technically different

```
<a href="cats.html">More Cats</a>
```

A tag is a term in angle brackets `< >`

Tags should be lowercase text

Opening and Closing Tags

A tag can be an "opening" or "closing" tag

- Closing tags begin with a slash `/` inside angles
  - `<p>This is a paragraph</p>`

An element can be "self-closing" (no content)

- `<img src="cat-with-glasses.png"/>`
- Some elements require content (open/close)
- Some elements don't (self-closing/empty/void)

Weird Exceptions

A few elements are exceptions to the normal rules

Examples:

- `<script></script>` must have separate open/close tags
    - even when no contents
- The "empty/void" elements don't require a closing
    - But in HTML5 CAN optionally self-close
    - `<input>`
    - `<meta>`
    - `<img>`

The
 element

A great example

- Used to create a visual line break
    - But that's not semantic!
        - Except for poetry
- Should almost never be used!
    - Except for poetry
- Does not require a close
- Has no content
- `<br>` or `<br/>`
    - But you shouldn't be using it

Attributes

A tag can have "attributes"

- after tag name, before angle bracket
- `name="value"`
    - `<img src="cat.png" alt="Jorts the Cat"/>`
- name without quotes
- value with quotes
- (tradition) no space around the `=`
- (tradition) double quotes (`"`) around the value
- This traditional syntax **required** for this class

Empty Attributes

Some attributes don't have values

- simply exist or do not exist
- indicate boolean states
- Ex: `disabled`, `readonly`, `selected`

```
<input type="text" disabled/>
```

Old Internet Explorer required values :(

Do not give these attributes values

Just include them or not

- Because the values are strings, not booleans

References

Elements can refer to other files in different ways

This is annoying, but you just have to learn them

- `<img src="cat-wearing-hat.png"/>`
- `<a href="other-file.html">Link</a>`
- `<link href="file.css"/>`
- `<script src="file.js"></script>`

HTML element ids

2 common ways to identify specific elements

- one being by "id"

The `id` attribute

- Unique per-page
    - ex: only one id "root" per page
- only one `id` per element
    - ex: element w/id "root" has no other id
- Commonly used in direct HTML
- Commonly AVOIDED in dynamic HTML

```
<div id="root">This is the root element</div>
```

HTML element Classes

Specific elements can be identified by "class"

- No relation to programming concept (**None**)
- Many elements can have the same class
- An element can have many classes
- Multiple classes separated by spaces in value
- Order in the attribute value doesn't matter

```
<div class="selected example">A div with classes</div>
<div class="example">Another div on the same page</div>
```

- For INFO6250: lowercase and kebab-case (or BEM) (**Required**)

Capitalization Styles Matter!

- `kebab-case` (**CSS; HTML attributes**)
  - ALL lowercase; words hyphenated (`-`)
- `MixedCase` (**JS Components**)
  - Words squished together; each capitalized
- `camelCase` (**JS variables**)
  - Words squished together; each capitalized
  - First letter NOT capitalized
- `snake_case` (**Not used in JS/HTML/CSS**)
  - ALL lowercase; underscored (`_`) words
- `UPPER_SNAKE_CASE`/`CONSTANT_CASE` (**JS Constants**)
  - ALL uppercase; underscored (`_`) words

What words to use for HTML classes

- HTML classes are used for CSS and JS
    - Sometimes call "CSS classes" for this reason
- Different standards exist
    - We will use `semantic` and `kebab-case`
    - BEM style is fine if you know it
- Like with HTML semantics
    - Semantic classes name what they identify
    - Not for the intended effect.
        - Bad: `bold`, `red`, `left`
        - Good: `review`, `selected`, `menu`

(**Required**) INFO6250 requires semantic class names

What is CSS

- C - Cascading
- S - Style
- S - Sheet

A set of rules for appearance

- that apply in "cascading" layers
- based on STRUCTURE
    - elements
    - classes
    - relationships
    - attributes
    - states

Rules and Selectors

A "CSS Rule" is a "selector"

- and a block of "declarations"
- setting the "value" of "properties"

A "selector" decides what the declarations apply to.

Each declaration ends in a semi-colon

```
p {
  font-family: sans-serif;
  text-align: center;
  font-size: 1.2rem;
  color: #BADA55;
}
```

Selectors

- HTML ids `#root { color: aqua; }`
- elements `p { color: #C0FFEE; }`
- HTML classes `.wrong { color: red; }`
- combinations `p.wrong { color: red; }`
- descendants `.wrong p { color: red; }`
- children `.wrong > p { color: red; }`

Any mix of the above, plus less common selector types

But you can't apply rules based on descendants (yet!)

Selectors ultimately match elements

Specificity

What if many rules can apply to an element?

- Rules have "Specificity"
- Declarations marked `!important` win (don't do)
- Inline CSS on the element wins (don't do)
- The more specific selector wins
    - #id is most specific
    - class less so
    - tag type is least
    - totals combine, so `.some.class` is twice as specific as `.class`
- If all else equal, most "recent" rule overrides older rule

Exceptions

Use `!important` when overriding outside styling

- `.some-lib div { color: #FEF1F0 !important; }`

You can use inline CSS on an element if

- you're making changes via JS **AND**
- those changes have unknown values in advance
- Inline CSS Okay: changing size by dragging a mouse
- Inline CSS Not Okay: setting an element to hidden/not hidden

CSS Use

CSS styles the document

If we want parts to change

- Have new styles existing
    - Matching different selectors
    - Change HTML to match alternate selectors
        - Usually a class change

This is not intuitive! (but is powerful!)

- Need to think about structure and classes
    - Describe state of page
    - Map to appearances

What is Javascript (JS)

Core rule: Understand the difference between:

- JS on the browser
- JS on the server

They are dramatically different

- A little in syntax
- A lot in what they do
    - And when they do it
    - And on which computer they run

JS in the browser

JS in the browser

- Runs in the browser
- On THEIR machine (not on the server!)
- Knows only the data in this JS and in the page
- Can change the HTML
- Can add in reference to more CSS or JS
- Completely visible to the user

JS is the only (real) option to run in the browser

JS on the server

Code running on the server can be in any language

- JS not special here like it is on the browser

For us JS is just convenient for the same language

- No access to the rendered page
- No awareness of what user is "doing"
- Server can only respond to requests

JS on server vs browser are completely disconnected

Summary

- The different roles of HTML, CSS, JS
- What is semantic HTML
- Dos and Do Nots for element class names
- Different kinds of CSS selectors
- CSS rules of Specificity
- Diff between server-side JS and client-side JS

Summary - Requirements for this Course

In and out of this course:

- HTML used semantically
- HTML boolean attributes have no values

In this course (and I recommend outside):

- HTML attributes with no spaces around `=`
- HTML attributes with double quotes around value
- CSS class names are semantic
    - and lowercase and kebab-case/BEM

```
<input name="street-address" class="address" disabled />
```