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 *content* of the page
  - WITH regard to structure
  - WITHOUT regard to appearance
- CSS - The appearance of the content
  - Defined by structure
- JS - Interactions with the content

BROWSER RENDERING

Browser rendering an HTML page

- Figuring out the size and visual properties of very element "box"
- Downloading CSS and IMG files as references are encountered
- Applying those files to the rendered output, updating the sizing and visuals as needed
- Downloading JS as encountered
- Running that JS, modifying the output-to-render as needed

SEMANTIC HTML

HTML is the content, and the structure of that content.

Think an organized list of everything in the page, like an outline, but including the text.

You can try to make the HTML look like what you want to see, but that will fail

- Devices (mobile, desktop, old, new) work differently
- Browsers display things differently (How does one indicate a paragraph?)

WHAT DOES "SEMANTIC" MEAN?

"related to meaning"

A sentence might be a paragraph, or a heading, or part of 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 text to create: "tag"
- the concept: "element"

the terms are often used interchangeably

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>`

A tag can be "self-closing"

- `<img src="cat-with-glasses.png"/>`

Technically self-closing tags are no longer a direct thing, but you'll find them commonly used

- Some tags require content (open/close)
- Some tags don't (self-closing)

ATTRIBUTES

A tag can have "attributes"

- after tag name, before angle bracket
- name="value"
- name without quotes
- value with quotes

Traditionally there is no space around the = and use double quotes around the value

- This traditionally syntax **required** for this class

EMPTY ATTRIBUTES

Some attributes don't have values, but are simple defined or not.

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

Previous versions of HTML had the option to give these values. Don't. Just include them or not.

REFERENCES

Tags can refer to other files in different ways.

This is annoying, but there for historical reasons.

- `<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

Specific instances of elements are identified in two common ways, one being by "id".

- Unique per-page: only one element should have a given id (example: only one id "root" per page)
- only one ID per element (example: the element with id "root" can't have any 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">An example of a div with classes</div>
<div class="example">Another div on the same page</div>
```

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

NAMING HTML CLASSES

- HTML classes are used for CSS and JS to affect certain elements
  - Sometimes call "CSS classes" for this reason
- Like with HTML semantics, classes should be named for what they identify, not for the intended effect.
  - Bad: `bold`, `red`, `left`
  - Good: `review`, `selected`, `menu`

WHAT IS CSS

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

A set of rules for appearance that apply in "cascading" layers

CASCADING

Cascading is hotly debated as whether it is good/bad

It is NOT like any other styling rules for other tech

It allows for reuse of rules

Arguments over whether it does so well

RULES AND SELECTORS

A "CSS Rule" is a "selector" and a block of "declarations"

A "selector" decides what the declarations apply to.

Each declaration ends in a semi-colon

```css
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 to your ancestors!

PRECEDENCE

What if my rules can apply to an element?

This is known as "precedence"

1. Inline CSS on the element wins (don't do)
2. Declarations marked `!important` win (don't do)
3. 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`
4. 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

WHAT IS JAVASCRIPT (JS)

We will cover this a lot.

Core rule: Understand the difference between:

- JS on the browser
- JS on the server

JS IN THE BROWSER

JS in the browser

- Runs in the browser
- On their machine (not on the server!)
- Does not know anything other than what it has and the page it is on
- Can change the HTML
- Can add in reference to more CSS or JS
- Completely visible to the user

For the most part, JS is the only option to run in the browser

JS ON THE SERVER

Programs 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
- Server can only respond to requests

JS on server and JS on 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
- CSS rules of Precedence
- Difference between server-side JS and client-side JS