HTML

HTML (Hyper Text Markup Language)

- Provides **structure** to the text of the document
- Defines data about the text
- Does NOT directly define the appearance
    - Common mistake!
    - You CAN use HTML to make an appearance
        - But it ends up hard to use/change
        - Bad idea to try

Intro to HTML

Is HTML a language?

- It is the "L" in the name

I mean, is it a programming language?

- "No", if you ask for vars+read/write+conditional
- "Yes", if you mean a syntax to instruct a computer
- But why are you asking?

Gatekeeping is not good, don't do it.

- programming is breaking down human-size problems to computer-size

Declaring an HTML document

- HTML has a few different versions
    - Mostly the same
    - But differences do matter!
- All modern HTML has top of document:

```
<!DOCTYPE html>
```

- If omitted, browser uses **quirks mode**
    - Allows weird past behavior to avoid breaking
    - You don't want that
    - Common interview question!

Browser and HTML

- Browser will guess for bad HTML
- MISTAKE to rely on this

Working, but not valid, webpage:

```
hello world
```

Try it in Chrome: File->Open

HTML Elements

HTML is made up of **elements**

- Starting **tag**
- Content
- Ending tag

tags are wrapped in **angle brackets**

- `< >`
- Ex: `<p>This is a paragraph</p>`

Starting Tags

- Wrapped in angle brackets
- **type** of tag is in between angle brackets

```
<p>This is a paragraph</p>
```

Element Contents

Contents can be

- Text
- Other elements
    - NOT "tags" - complete elements
    - Elements can "nest"
    - Elements cannot "overlap"
- Whitespace (spaces, tabs, new lines)
    - All content whitespace "collapses"
        - Renders as a single space

Rules about Children

Elements as content of another element

- **Child element**
- Can have **descendants**
- Some elements have rules about contents
    - Ex: a "item list" element (`<ul>`, `<ol>`)
        - only "list item" (`<li>`) as children
        - but `<li>` may have any kind of children
    - Ex: a "p" element
        - may not have block level descendants

Nesting Elements

- HTML elements **can** "nest"
- HTML elements **can't** "overlap"
- Elements can contain elements, text, and/or comments (`<!-- a comment -->`)

Valid: `<div><p></p></div>`

Not Valid: `<div><p></div></p>`

Nesting Element Rules

A handful of elements have additional rules

- Ex: `<p><div></div></p>` not allowed because a "p" element is a "paragraph"
- a "list" element (`<ol>`, `<ul>`, `<dl>`) can only have "list items" (`<li>`)
    - But list item elements can have anything

Closing Tags

- Type wrapped in angle brackets
- Start with a `/`
- Examples:
    - `</p>`
    - `</ul>`
    - `</li>`

Self-closing Elements

If an element has no content, it might **self-close**

- Also known as **empty elements** or **void elements**
- No separate opening tag/closing tag
- One tag, has `/` before ending angle bracket
    - Example: `<link/>`
    - Not required in HTML, but allowed
    - Is required in JSX (React)
    - Common to always use
        - Programming is communication

Some elements seem weird

Example: `<script>` element CAN be empty

- Often is empty
- But is not self-closing
    - Because it CAN have contents
- MUST have separate closing tag
    - Even when empty

Elements define semantics of contents

Semantic === Related to meaning

- Not appearance, but meaning

A string of words

- Heading?
- Paragraph?
- Emphasized?
- Text to link elsewhere?

Humans use visuals to infer the semantics

- But the visuals don't CREATE the semantics
    - The other way around is true
- HTML should define the semantics
    - Then visuals based on semantics

It isn't a heading because of how it looks

- It looks like a heading because it is defined as a heading

Why do we care about semantics?

HTML is used is MANY ways

- Desktop browsers
    - of many resolutions
- Mobile browsers
    - of many resolutions
- Read by programs
- Assistive Technology
    - Screen Readers
    - Braille interpreters
- Keyboard, mouse, tablets

Elements are data about content

- Semantic meaning
- Also additional data!
    - URL for a link, image, etc
    - Assistive hints
    - Relationship between elements

Provided by **attributes** on the element

Attributes on an Element

Attributes are in the starting tag

- Before the closing angle bracket
- Separated by spaces
    - from type and other attributes
- Either a simple word or 'key="value"' text
    - `<input disabled placeholder="Enter Name"/>`
- Order of attributes does not matter
- For this course, with 'key="value"'
    - No spaces around `=`
    - Double quotes (`"`) around value

Empty Attributes (no value)

- Old advice will tell you to set a value
    - But Internet Explorer is dead
    - Don't follow that old advice!
- Good: `<input disabled />`
- Bad: `<input disabled="true"/>`
    - Why Bad? "false" is same as "true" here!

Attribute examples

```
<button disabled>Click Me</button>

<input type="checkbox">

<img src="https://examplecat.com/cat.png" alt="cat drawing">
```

Special Attribute: `id`

Every element can have an `id` attribute

- Ex: `<button id="accept">Accept</button>`
- Value MUST be unique on the page
    - Becomes difficult as pages get more complex

The `id` uniquely identifies the element

- Most elements are not given an `id` though
- Only if we need to refer to that exact element
    - And only that element

Special Attribute: `class`

- Not related to programming concept "class"
- More like "category"
- Like `id`, identifies elements
- Unlike `id`, does not have to be unique
- An element can have multiple classes
    - Single `class` attribute
    - Assigned a space-separated list of strings
    - Order does not matter
- `<button class="primary good">Accept"</button>`
- `<button class="good primary">Accept"</button>`

Class names are heavily used in CSS + JS

- Many approaches
    - Different Pros/Cons
    - Worst is to mix them up
- For this course, class names
    - Must be lowercase
        - not MixedCase, not camelCase
    - Hyphenated (kebab-case) or BEM (later)
    - Must describe the element (semantic)
        - Not the desired appearance
        - Good: `menu`, `active`, `selected`
        - Bad: `left`, `bold`, `small`

Skeleton of a Page

- Every page must declare `<!DOCTYPE html>`
    - Not HTML, no closing tag
- Every page must have a `<html>` element
    - Everything goes inside this
- The `<html>` element
    - Contains a `<head>` element
        - Contains a `<title>` element
    - Contains a `<body>` element
        - **sibling** of `<head>`
- Anything omitted is assumed
    - Being explicit prevents poor assumptions

Basic page

```
<!doctype html>
<html>
  <head>
  </head>
  <body>
    Hello World
  </body>
</html>
```

Still a few baseline improvements

- Should have a `<title>` in the `<head>`
- Should define the **encoding**
- Should define the language

We can check simple HTML in the W3 Validator

- **https://validator.w3.org/**

Defining the Title

- This is what shows in the browser tab
- Should be concise and informative
- Must be inside `<head>`
- Only one `<title>` per document

```
<!doctype html>
<html>
  <head>
    <title>Cats of the Internet</title>
  </head>
  <body>
  </body>
</html>
```

What is encoding?

- Computers store binary
- What binary numbers represent which characters?
    - That definition is the "encoding"
    - We have MANY encodings
- These terms overlap a lot,
- HTML5 must use "utf-8"

Setting the encoding

```
<!doctype html>
<html>
  <head>
    <title>Internet Cats</title>
    <meta charset="utf-8"/>
  </head>
```

- Is set as attribute on `<meta/>`
    - `<meta>` is self-closing element
- Inside `<head>` to set charset
- `<meta>` is used for page-wide things
    - This is all we'll use it for
- This is a "just do it" thing

Setting the language

- set as `lang` attribute on `<html>`
- uses **country code** based **language tag**
- Examples
    - `en` (English)
    - `en-US` (United State English)
    - `ja` (Japanese)
    - `zh-Hant` (Chinese written using the Traditional Chinese script)
    - `zh-Hans` (Chinese written using the Simplified Chinese script)

Setting the language in HTML

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <title>Internet Cats</title>
  </head>
  <body>
  </body>
</html>
```

Cat list, simple

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <title>Internet Cats</title>
  </head>
  <body>
    <ul>
      <li>Grumpy Cat</li>
      <li>Jorts</li>
      <li>Nyan cat</li>
    </ul>
  </body>
</html>
```

What is UL?

Notice we have a `<ul>` element

- unordered list
- there's an order, it just isn't important
- want to guess what `<ol>` is?

The list is made up of individual **list items** (`<li>`)

Why not have many list items without `<ul>` element?

- how to separate two lists next to each other?

Semantic HTML

You don't want "just" HTML

- you want "Semantic HTML"

Semantic means "related to meaning"

HTML where the structure is meaningful

- structure is not based on appearance
- structure is not ignored

More on this later, key lesson:

- Pick elements based on what they *mean*
    - not what they *look like*

So what is all of HTML?

Honestly, I don't remember it all. MDN is a good friend.

https://developer.mozilla.org/en-US/docs/Web/HTML/Element

Core elements:

- html
- head
- body

Common head elements

- title
- meta
- link
- style, script (more later)

Elements commonly in the body

- a
- b/strong, i/emphasis
- img
- p
- ol, ul, dl
- h1-h6
- div
- section, article, aside, header
- nav
- table elements
- various form elements (more later)

Table Elements

Back in the bad old days, tables were used to control the layout of web pages

DO NOT USE TABLES FOR LAYOUT

- Hard to understand
- Hard to change
- Semantically wrong
- a11y problems

Use tables for tables of data

Linking

The core of the web is actually LINKS

- originally a format to share and crosslink data like scientific papers

Before you can understand links, you have to understand URLs

Uniform Resource Locator (URL)

A URL is an address (not just web, all of internet)

- what syntax to use (protocol)
- what port to use that on (port)
    - different protocols have default ports
- what computer to talk to (domain)
- what thing to request (path + file)

`http://northeastern.edu/wp-content/uploads/COE.jpg`

"Hey NEU server, I want `/wp-content/uploads/COE.jpg`"

Linking Pages

A link tells the browser to allow navigation to a different web page

```
<a href="http://neu.edu">Go to NEU</a>
```

"a" elements (anchor) have text content and an `href` attribute (hypertext reference) that says where to go when followed.

Let's create an "About Cats" page as a separate html file, and link to it from our Cat List page

Cat list, with link

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <title>Internet Cats</title>
  </head>
  <body>
    <a href="about.html">About Cats</a>
    <ul>
      <li>Grumpy Cat</li>
      <li>Jorts</li>
      <li>Nyan cat</li>
    </ul>
  </body>
</html>
```

About Cats, with link

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <title>About Cats</title>
  </head>
  <body>
    <a href="cats.html">Internet Cats</a>
    <p>
      This is a site about cats.
      What better way to master the web?
    </p>
  </body>
</html>
```

Not Fully Qualified

Why were those `href` so short?

We didn't use **fully qualified** urls

- No protocol? Same protocol as current page
- No domain? Same domain as current page
- No path? Same path as current page

Just listing the filename means it links to different files in the same directory

Relative vs Absolute

Common to omit protocol + domain

- Makes it easier to develop and move

File references can be **relative** or **absolute**

- Relative to current directory
- Absolute based on a **root** directory

The *root* is NOT the filesystem root

- it is the webserver **document root**

Otherwise any file on the computer is requestable

How to make absolute references

Absolute file references will always begin with `/`

- Sorry Windows users, the Internet is Unix-based

```
<a href="/examplecat.png">See Cat</a>
<a href="/games/minecraft/data/guide.html">Punch Trees</a>
```

If it isn't absolute, it is relative

```
<a href="about.html">About Us</a>
<a href="../dogs/why.html">Drool and barks</a>
```

Where to use URLs/references

Different elements use references differently

- `a` tag uses `href`
- `img` tag uses `src`
- `link` tag uses `href` (e.g. to load CSS)
- `script` tag uses `src` (to load JS)
- Because life is not easy

`src` is for "replaced" elements, `href` to connect to a resource without replacement

- but you often have to look up to know this

```
<a href="https://examplecat.com/cat.png">A cat</a>
<img src="https://examplecat.com/cat.png">
```

Link Text

The contents of the `<a>` element are the "link text"

- May not be text
- For a11y, there should be some text
- Do not use "Click here to..."
    - Definitely not "click here"
- Do not use the url itself
- Do name the destination

Summary - HTML Intro

- HTML creates a **structured document**
- Semantic HTML describes structure
    - Not appearance
- Elements are **tags** and **content**
- Elements can **nest** but not overlap
- Elements may have restrict content

Summary - HTML Attributes

- Elements may have **attributes**
- Attribute order does not matter
- Attributes may have values
    - or may be simple "present"
- `id` is a special attribute
    - Unique to that element
- `class` is a special element
    - space-separated list
    - list order does not matter
    - categories associated with element

Summary - HTML Page

- `<!doctype html>` declaration
    - Not an element
- `<html>` element
    - `lang` attribute
- `<head>` element
    - contains `<title>` element
    - contains `<meta>` element
        - `charset="utf-8"` attribute
- `<body>` element

Summary - URLs

- **URL** is an internet address to a resource
- protocol, domain, port, path, query, hashref
- non-**fully qualified URL** takes page as defaults
- often used as `src` or `href` attribute values
    - elements will specify which they use
- a path can be **relative** or **absolute**
    - Absolute path starts with `/`
        - Relative to webserver **document root**
    - Relative path does not start with `/`
        - Relative to current page path
    - url with domain+path is absolute

Summary - Links

An `<a>` tag takes an `href` attribute

- Creates a link
- Contents are link text
- Browser will **navigate** when following
    - Loads new page from url
    - Previous page is no longer loaded
- Link text should follow a11y tips
    - Name destination (as text)
    - Avoid "click here"
    - Avoid urls