CS CAPSTONE WEB DEV NOTES

OCTOBER UNIT

FRONT END ONLY

Caroline Richards

HTML BASICS

- Hyper Text Markup Language
- Skeleton of websites

```
<!DOCTYPE html>          <- must start w/
<html>
    <head>
        <title> My Title </title>          <- page title
        <link href="" rel='stylesheet' />  <- link CSS
    </head>
    <body>
        <!-- other HTML tags here! -->     <- comment
    </body>
</html>
```

BASIC TAGS

`<h1></h1>`	heading #1-6	`<ul></ul>`	unordered list (bullets)
`<p></p>`	paragraph	`<li></li>`	list item
`<div></div>`	generic container	`<a href="">`	hyperlink
`<img src="" />`	image	`<strong></strong>`	bold text
`<ol></ol>`	ordered list w/ numbers	`<em></em>`	italics text

<~~<div></div>~~> bad :(

SEMANTIC HTML

STRUCTURE TAGS

<header></header> encompasses navigation, banner, etc.

<nav></nav> navigation, usually like bar w/ unordered list, logo, etc.

<main></main> main content between header & footer

<section></section> area of page w/ same theme

<article></article> stand-alone content like articles, blog posts, recipes, etc.

<aside></aside> content related to main there but not required, like links to similar things.

<footer></footer> footer, usually copyright & small nav.

MEDIA TAGS

<figure></figure> surrounds images, diagrams, code, & other media

<figcaption></figcaption> describes media, located within <figure>

<video src="" controls></video> video w/ basic player

<audio src=""></audio> audio player

<object data="" application="" /> embed other media, like pdfs

HTML TABLES

```
<table>                          ← define table
    <thead>                      ← contain headings
        <tr>
            <th> Heading 1 </th>
            <th> Heading 2 </th>      } table headers / column titles
        </tr>
    </thead>
    <tbody>                      ← contain actual data
        <tr>                     ← row
            <td colspan = "2"> R1C1 </td>
            <td> R1C2 </td>
            <td> R1C3 </td>          } columns
        </tr>                        } span 2 columns
        <tr>
            <td> R2C1 </td>
            <td> R2C2 </td>
        </tr>
    </tbody>
</table>
```

BASIC CSS

- Cascading Style Sheets
- Add styling

- Must link in HTML file in head

```
<link href="" rel="stylesheet" />
```

- select by element, class, or id

```
<p id="p1" class="paragraph">Hello World</p>
```

ELEMENT

```
p {
    color: 'red';
}
```

CLASS

```
.paragraph {
    color: 'blue';
}
```

ID

```
#p1 {
    color: 'pink'
}
```

1 → 2 → 3

SPECIFICITY

- override all other rules

```
color: 'yellow' !important;
```

- comments

```
/* like multiline Java */
```

- best practice to isolate CSS in files: not do inline styling

CSS RULES REFERENCE

TYPOGRAPHY

font-family
font-size
font-weight: bold/normal/lighter/bolder;
font-style : italic

text-transform : upper/lowercase;
letter-spacing : (px/ems)
word-spacing : (px/ems)
line-height : (unitless)

- import font families

<link href='' rel=stylesheet /> ← HTML
@import url(''); ← CSS

COLOR

color
background-color
opacity : 0-1; (1 = solid)

- named colors, hex, rgb(), rgba(), hsla()

FLOW

position : static — default
: relative — to parent adjust w/ left, right, top, bottom rules
: absolute — doesn't scroll, other elements ignore
: fixed — like absolute but scrolls w/ user
: sticky — like fixed but scrolls only to certain point

BOX-MODEL

width
height
border: 1 px solid black

padding
margin
border-radius

visibility: hidden/collapse/visible
overflow: hidden
: scroll
: display (element)

CSS DISPLAYS

- change how items are displayed

display: block — one line per item
inline — share line but no box
inline-block — share line + box

flex — uhhhhhh

- box sizing

box-sizing: content-box; — padding + margin extra
: border-box; — padding + margin included

REACT

- JavaScript library to aid in web development
- Uses components individual files/small code chunks responsible for one thing, like rendering an element

```
import React from 'react';

function MyComponent() {
    return (
        <div>
            <h1>Hello World</h1>
            <p>This is my component</p>
        </div>
    );
}
export default MyComponent;
```

must return a parent / outer level element

everything must be exported AND imported

React Component File

```
import React from 'react';
import MyComponent from './MyComponent.js';

function App() {
    return <MyComponent />;
}
export default App;
```

components imported

added to App

all JSX must have closing tag

App.js file

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from 'App.js';

ReactDOM.createRoot(
    document.getElementById('app')
).render(<App/>);
```

actually renders on page

from index.html

Index.js file

CREATE REACT APP LOCALLY

- must have node installed

- Create react project

  npx create-react-app appName

- update

  npm install react-script@latest

- Start server - must be called inside project folder

  npm start

- Troubleshoot cache problems

  npm cache verify

  npm cache clean --force ← only use if absolutely necessary
  slows app WAY down

REACT HOOKS

- Functions that allow you to manage the internal state of components and handle post rendering side effects. Declare how user interface should look based on state.
- Only call hooks on the top level
- Only call hooks from React functions

① UseState hook

```
import React, { useState } from 'react';
```

- calling useState() returns an array with
  1) The current state
  2) a stateSetter function to update the state
- Both can be destructured into variables

```
const [currentState, setCurrentState] = useState(<any-default-here>);
```

- it is best practice to update state w/ a callback function to avoid outdated state values

```
setState( prevState => prevState + 1 );
```

- Don't forget about the spread operator: `...<variable-state>`
  or the computer Property Name thing:

```
function myFunc( name, value){
  setState[name] = value);
```

I love this
wish I had found sooner

② useEffect() hook

```
import React, { useEffect } from 'react';
```

- takes a function as an argument and calls it after each render