CMPSC 302
WEB DEVELOPMENT

Planting your garden(s)

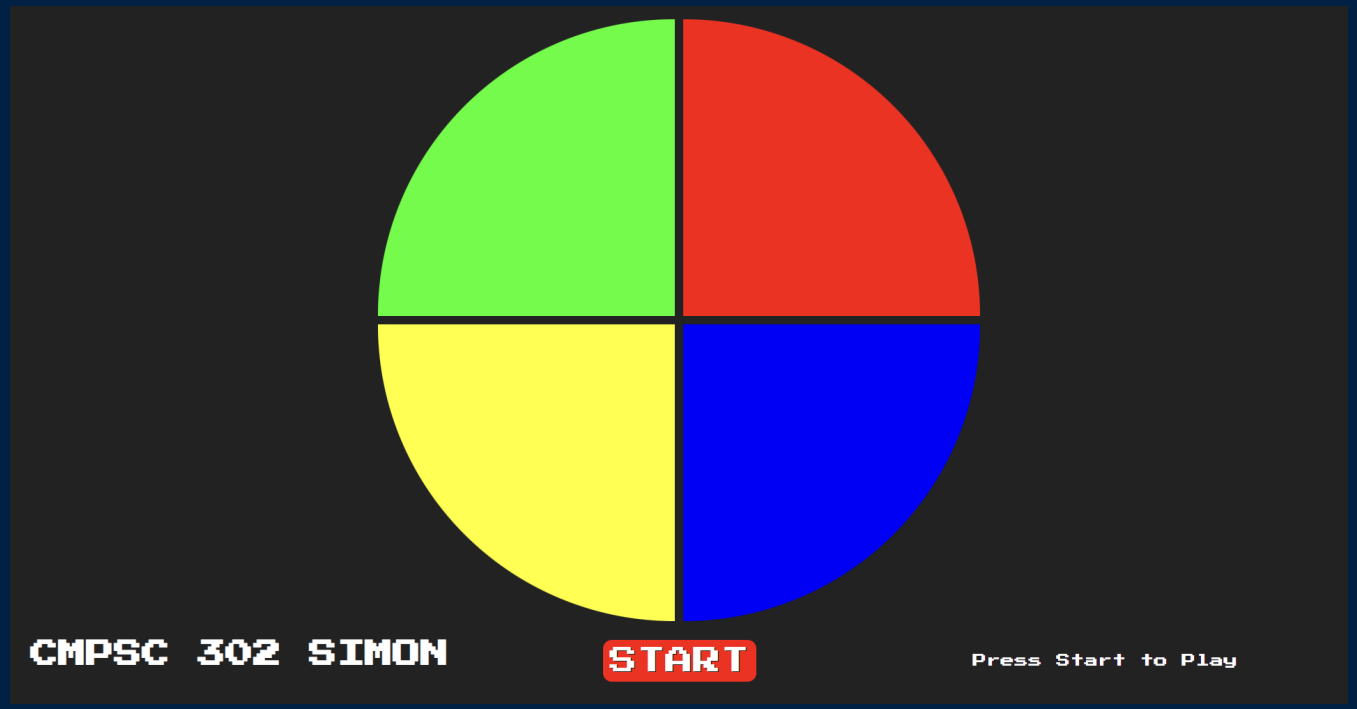
- Together, we are going to perform a Pull Request/Merge of your content in your Week 2 lab
- If you're not already at GitHub, navigate there and follow along
  - If you know what to do or have done any of the steps we're about to, just follow along, or
  - Just do the steps

The week's work

- Today, you're getting a repository that we're going to use for the next 1.5 weeks
- We will gain:
  - Practice in HTML/CSS
  - Experience doing more "responsive" design
    - We've already been kinda doing this
    - This time it's via something called
  - Collective knowledge in Javascript
  - At the end, a cool game

Simon says

- Your job today (and partially Friday) is to make a layout.
- Our layout should look like:



New tools

- To make our touchpads, we're going to use:

```
<button></button>
```

Simon says

- Rules are in the README, but:
  - Name the IDs the names specified in the document
  - Follow other guidelines
- We will tackle the “mobile” part together
  - This will involve learning something new, but likely on Friday
- Today is dedicated to work time

Media training

- We have the ability to change our designs *in response to* changing screen conditions
  - Hence “responsive”
- Typically, this responds to widths and screen “orientations”
  - Screen sizes are typically demoninated in “pixels”
    - This is one of the few times that it’s pretty much the only way
- CSS “Media Queries” accomplish this

Media training

screen size

```
@media only screen and (max-width: 1024px) {
  #selector {
      property: value;
  }
}
```

Media training

screen orientation

```
@media only screen and (orientation: landscape) {
  #selector {
      property: value;
  }
}
```

Media training

- We can simulate these in our “inspect” mode:

Javascript

- Mozilla is responsible for this mess (c. December 1995; the worst holiday gift ever)
- A separate language which is the *third* layer of our "tech stack" (along with HTML/CSS)
- 97% of web sites implement at least *some* Javascript (2021)
  - That's probably an underestimate
- Like HTML/CSS, browsers (e.g. Firefox, Chrome) *interpret it* when users visit a page that uses it

Javascript

Nothing to do with Java
(or coffee)
(v. sad)

- We're now moving from markup ("meaning" things) to script ("doing" things)
- For example:
  - We can *dynamically* change parts of page text
  - Alter element and page colors in response to user actions
  - Animate page interactions for visual appeal/mimicking real-world processes (e.g. drag-and-drop)
  - Give user feedback
    - How does a user know they've clicked a button?
    - How does a user know that they're currently hovering over a clickable element?
    - How does a user know that they've taken the *right* action?

Javascript

- Features a different *syntax* ("way of speaking") than HTML/CSS
- Is organized around variable values and events
  - Variables store information
    - Numbers
    - Text
    - ...and more
  - Events
    - Page load
    - Clicks
    - Mouse entry or exit of a given element
    - ...likewise, more

Javascript

Navigate to your game interface and open the Inspector window.

Using Javascript

- However, *like* HTML, everything starts with a document and various selectors
- In our Inspector, under the console tab, type:

```
document;↵
```

(represents the Enter key)

Using Javascript

- Because we can access the *entire* DOM, we can get anything from it:
  - Anything with a given class attribute
  - Anything with a given element ID
  - Anything with a given tag type
  - ...really, anything on the page
    - (Yes, we can even create new page content)
- For example, to get the #top-left button by element ID:

```
document.getElementById("top-left");
```

object we want to explore

procedure we want to do

thing to find

Sometimes a semicolon is just a semicolon

Try it out

- Get the remaining buttons:
  - top-right
  - bottom-left
  - bottom-right
- Get the “Start” button
- Get the points field
- Get the element you’re using to make your grid

Using Javascript in style

- Now that we've discovered how to "acquire" elements, let's make some changes
- I don't like my start button's color any more; I want it to be pink
  - Pink's the best
- I'm going to *store it*, though, so that I can do more to it later
  - Here, we'll use a *variable*.

```
var startBtn = document.getElementById("#start");
```

Keep this for later under the name "startBtn"

Using Javascript in style

Notify browser that we want to change a style value

Name of item we "saved"

```
startBtn.style.backgroundColor = "#FF3898";
```

Value to set property to

Property to change

But, if we're gonna go '80s, let's go '80s:

```
startBtn.style.color = "#222222";
```

Using Javascript in style

- Experiment changing elements by their IDs and various style combinations!

Learning another (hard) refresh lesson

- Now, refresh the page
- What does this tell us (hint: we've learned this lesson before)?

Scripts, or it didn't happen

- Like HTML/CSS changes in the Inspector, we need to save the script we want to use in a file
  - I've given you this file in: scripts/ui.js
  - Right now, it's blank
- Populate it with selecting each button as an individual variable
- It's already integrated into the HTML, and we'll look at that when we come back together in 10 – 15 minutes.

Putting this into action

- Now that we have a saved script that “persists,” we can add some interesting events
  - Today, we’ll start our game development
- To do this, we use a procedure called:
  - addEventListener
- We’ll attach it to our startBtn.

addEventListener

Event to watch for (a click)

Any context we want to give, none here

Element to “bind”

Start listening

Property to change (not style, here)

```
startBtn.addEventListener(“click”, () => {
    document.getElementById(“#points”).innerText = “Points: 0”;
});
```

Place to look and procedure to “run”

That which hath been open’t must be closéd