* Get lecture repo
* Slides
* PollEverywhere
* Bring up Visual Studio and load the lecture folder
* Bring up live-server and the Console

# Slide 2

Plan for today

I will talk for a bit about Stage 2... hopefully to make things clearer

# Slide 3

Project Stage 2

Object is to make your stage1 page interactive

Clicking, moving the mouse, etc... it's more than mouse-over. We want to see that when the user does something that the page changes. Not just with hyperlinks.

If you decide to move away from what you did in stage 1, that's fine, but you need to keep the same level of styling in place as what you did in your stage 1. It needs to be responsive, cool looking, have a grid, etc...

Phase 2 is still an individual project. This is so that everyone can show individually that they have acquired the skills from this course so far.

One thing to consider is that Phase 3 is when we start working with someone else. And naturally you will combine your work, so you might be better off waiting to change domains until that phase rather than now.

I am hoping to get grading done by Thursday of next week. There are a lot of projects and it's time consuming. You will still have 1.5 weeks before phase 2 is due, and you should be able to make progress prior to getting the grade and feedback from phase 1.

1) Fix the outstanding github issues from phase 1. I'll give you some things to fix. Hopefully they are relatively minor. But resolving these issues is something you will need to do.

2) Dynamically display external data

You'll use AJAX requests to pull in this external data. This isn't too hard and is something like 2 lines of code. It's a bit weird syntax, but you'll get it.

The dynamic data can be real (like poll data for the election) or it can be mock up data from a spreadsheet or CSV

file. It doesn't need to be from a public API. Feel free to generate some data on your own and save to a CSV if that is the most expedient and effective method.

You should start thinking about what data you want to pull in. You might need to change the focus of your app to find a way to use external data.

3) Enable user to interact with page and see content change

Clicking causes the page context to change and not just having and on/off switch

sequential - you need to be able to click on something it changes the context, then click on something in the new context and it will once again change. You need to be able to keep state in a variable to do this. There will be a basic example of this in the lecture demo today.

The Shiny Data Tables example meets the requirement.

polished and frictionless - get it working smoothly.

You are welcome to create a new html page, but it's not required.

4) Utilize another JavaScript Library

This should be another 3rd party library. There are a ton of options here

Calendar, maps, plots

Figure out how to use it. Then integrate into your page.

5) Accessible, responsive, well written

Buttons should be big enough on a small screen.

Use the aria-label stuff to handle when you dynamically alter the DOM so that a screen reader can know

Look through the JavaScript Style requirements link

# SLIDE 4

# SLIDE 5

# SLIDE 6

# SLIDE 7

The DOM is an API with the browsers interact

"document" is the object. we call methods off document object.

.querySelector(cssSelector) is easier than .getelement both of these are available off of the document object.

You use put in the same CSS Selector syntax with.querySelector as we've already been using for CSS selectors

# SLIDE 8

Basic example of how this works:

First you need a reference to the element

You can get the text of an element with the .textContent() method

You can assign a new value to the element, simply by assigning elem.textContent="this is different content!";

You use the .innerHTML method to add styles to the text.

Be careful when using .innerHTLM. If the user is providing the content here, it's a place where script could be injected.

Note that the HTML file is not changed.

What is happening is that the script is loaded after the HTML has been rendered and the script alters the DOM in memory and the browser shows the modified DOM

# SLIDE 9

Demo Example 1

First, we use the 'h1' as our querySelector parameter to get a reference theH1 to the element

we can also grab a paragraph within the header

And update the HTML (formatting) by calling subtitle.innerHTLM = subtitle.textContent + "<em>Prof Carlson</em>"

# SLIDE 10

Demo Example 2

Change an attribute

Not every property allows you to change a property in this way.

# SLIDE 11

Example 3

Change a Style

We use the classList.add to add additional styles. You can remove too.

# SLIDE 12

Example 4

Creating a new element

First, we create, then we set the innerHTML by assigning

then we add to the appropriate place in the DOM

Here we find the UL and append the newLi to the end of the classList

Again, be careful using .innerHTML()

# SLIDE 13

Use Java function to add items

Then call the function;

# SLIDE 14

Now you can iterate and quickly add items without typing a bunch of HTML

# SLIDE 15

Example 6

addEventListener(param1, param2)

First parameter is the kind of event (click, keypress, blah blah blah)

Second parameter is the callback function

Here we're watching the button

# SLIDE 16

Example 7

Keeping state as we toggle back and forth

Need a variable (isPuppy) to keep track of what's currently displaying