Week 8

Intro to JavaScript

Announcements

Thursday Lab Demos
“How to create files from scratch + code organization” workshop in lab this week!

Weekly homework resumes
HW7 due next Wednesday (go and vote)!

Final Project is out!
Check-in due **Nov 12**
Project due **Dec 1 @ 11:59PM PST**
Your project can be featured on our showcase website!

Final Project Submission: wdd.io/go/project-submit

Come to office hours via Piazza or wdd.io/go/OH

Give us anonymous feedback at wdd.io/go/feedback

HTML

Structure

CSS

Design

JavaScript

Function

What is JavaScript?

What can we do with it?

What does it look like?

Hands-on Basics

What is JavaScript?

- A programming language
  - Some people will distinguish between a scripting language and a programming language ¯_(ツ)_/¯
- Completely different from HTML and CSS (and Java)
- *Not scary!*
- Just like we wrote HTML and CSS to tell the browser what we wanted things to look like, we’re going to write JS to tell the browser what we want those things to do

What can we do with it?

- Our goal: write JavaScript code that will add interactivity to our pages
  - Specifically, JavaScript can *change* our existing HTML and CSS

- Examples: popup dialogs, image carousels, any time you see something (like menus) being expanded or hidden

- Also works on the backend, but we won't go into that...

Thinking about this interactive functionality..

We'll think of a feature to add to our webpage like:

When the user clicks on [this button/link/image],

then [add/change/remove] [this HTML/CSS].

Always think about what you want to accomplish before you start coding.

Examples

- When the user clicks on this image, then increase its size.
  - "increase its size" → *change its CSS* `width`
- When the user clicks on this button, then change colors to dark mode.
  - "change colors to dark mode" → *change CSS* `color` *and* `background-color`
- When the user clicks on this tab, then change the content.
  - "change the content" → *hide the first div and show the second div*
    → *set* `display: none` *to the first div and set* `display: block` *to the second div*
- When the user clicks on the up arrow, then increase the number.
  - "Increase the number" → *take whatever number is currently inside a div, add one to the number, and change the div content to this new number*

JavaScript’s Role in a Webpage

- When we write HTML, it just shows up when we open the .html file.
- When we write CSS, it shows up if there were CSS selectors that referred to specific HTML elements in the .html page.

- What about JS?
  - It’s not going to just “show up” on the page unless we tell it to modify HTML/CSS, which we’ll learn next week.
  - For now, it “lives” in… the console!
    - in Chrome DevTools, there’s a tab that says Console

Okay, how do we try this out?

1. Create a new file, **script.js** and put it in the folder of your website
2. Similar to the `<link>` tag for CSS, we also have to attach our JS file to an HTML page. Inside the `<head>` tag, add this line:

```
<script src="assets/js/script.js"></script>
```

Same rules as always - the filepath must be correct!

There is a closing tag!

Today’s concepts

- Variables
- `console.log()`
- `alert()` for fun
- Strings
- Functions

- Goal: be able to read basic syntax and do math!

Variables

Variables are like in math! We use a variable when we want to store the value of something and give it a name.

```
let name = "Joe";
let age = 20;
```

<u>IMPORTANT:</u> if we're storing text, it must be in **quotes**. This is called a **string.**

value to store, in this case, a number.

"let" means that we're creating a variable (block-scoped)

Make names meaningful! Usually a noun.

String Concatenation (fancy words for “put text together”)

We can add strings (remember: text is always in quotes) together.

Here are some examples:

```
let name = "Nicholas";
let greeting = "howdy " + name + "!";
```

Making JS Show Up - `console.log()`

We haven't made JS touch HTML yet. We will next week!

For now, we can see what's going on by calling the `console.log()` function.

```
let name = "Nicholas";
let greeting = "howdy " + name + "!";
console.log(greeting);
```

Result:

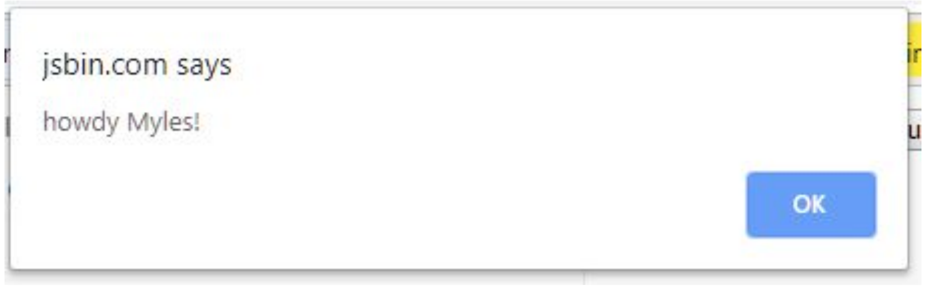
```
"howdy Nicholas!"
```

Making JS Show Up (for fun) - `alert()`

A more interactive (but it'll get annoying) way is to use the `alert()` function.

```
let name = "Myles";
let greeting = "howdy " + name + "!";
alert(greeting);
```

Result:



jsbin.com says

howdy Myles!

What happens? Type in the chat!

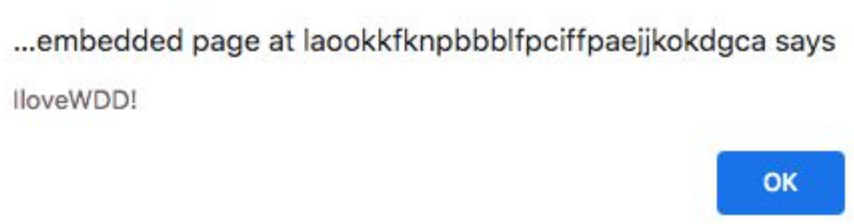
```
let course = "WDD";
let person = "I";
let emotion = "love";
let sentence = person + emotion + course + "!";
alert(sentence);
```

Result:

What happens? Type in the chat!

```
let course = "WDD";
let person = "I";
let emotion = "love";
let sentence = person + emotion + course + "!";
alert(sentence);
```

Result:


...embedded page at laookkfknpbbblfpciffpaejjkokdgca says

IloveWDD!

Basic Math

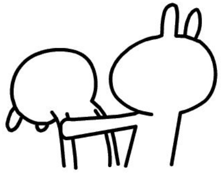
Now we can do math with our variables. We have: `+ - * / %`

Here are some examples:

```
let age = 20 + 5;
age = 20 - 2.5;

let doubleAge = age * 2;
```

Here we *re-assign* the value of `age` but we don't use `let` because the variable already exists.



Questions?

Functions

Also a bit analogous to math - a function takes values as input, does something with them, outputs a value. This is useful if we have a piece of code that we don't want to keep writing out.

means that we're defining a function

Make names meaningful! Usually a verb.

These are the function **parameters** (or **arguments**) that it takes.

```
function square(x) {
    return x * x;
}
```

"return" means, this is the final output value

The function usually does something with the parameters.

Using Functions

```
function square(x) {
    let result = x * x;
    return result;
}
```

This part **defines** the function. This does not *call* or *use* the function.

Note: let statements are block scoped, meaning that you cannot access “result” outside of the block of curly brackets.

“return” means, this is the final output value

```
let y = square(5);
let z = square(y);
```

Here, we **call** the function.So now the value of y is 25.

A variable can be the function input! This is the equivalent of square(25).

Using Functions

Functions do not always need to return a value. For example:

```
function showGreeting(name) {
    let greeting = "howdy " + name + "!";
    alert(greeting);
}

showGreeting("Myles");
```

Comments

Just like in HTML and CSS, we can also write comments in JS files.

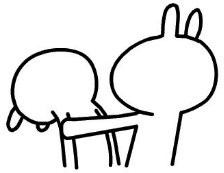
```
// showGreeting takes a string name and shows an alert
function showGreeting(name) {
    let greeting = "howdy " + name + "!";
    alert(greeting);
}

/* this comment is
   multiple lines  */
```

What is returned? Type in the chat!

```
// This comment says that the function does something.
function doSomething(course, num) {
     num = num + 90;
     let res = "I have taken " + course + " " + num + " times";
     return res;
}

doSomething(‘wdd’, 10);
```



Questions?