Functions

Lesson 11

FEWD

GENERAL ASSEMBLY

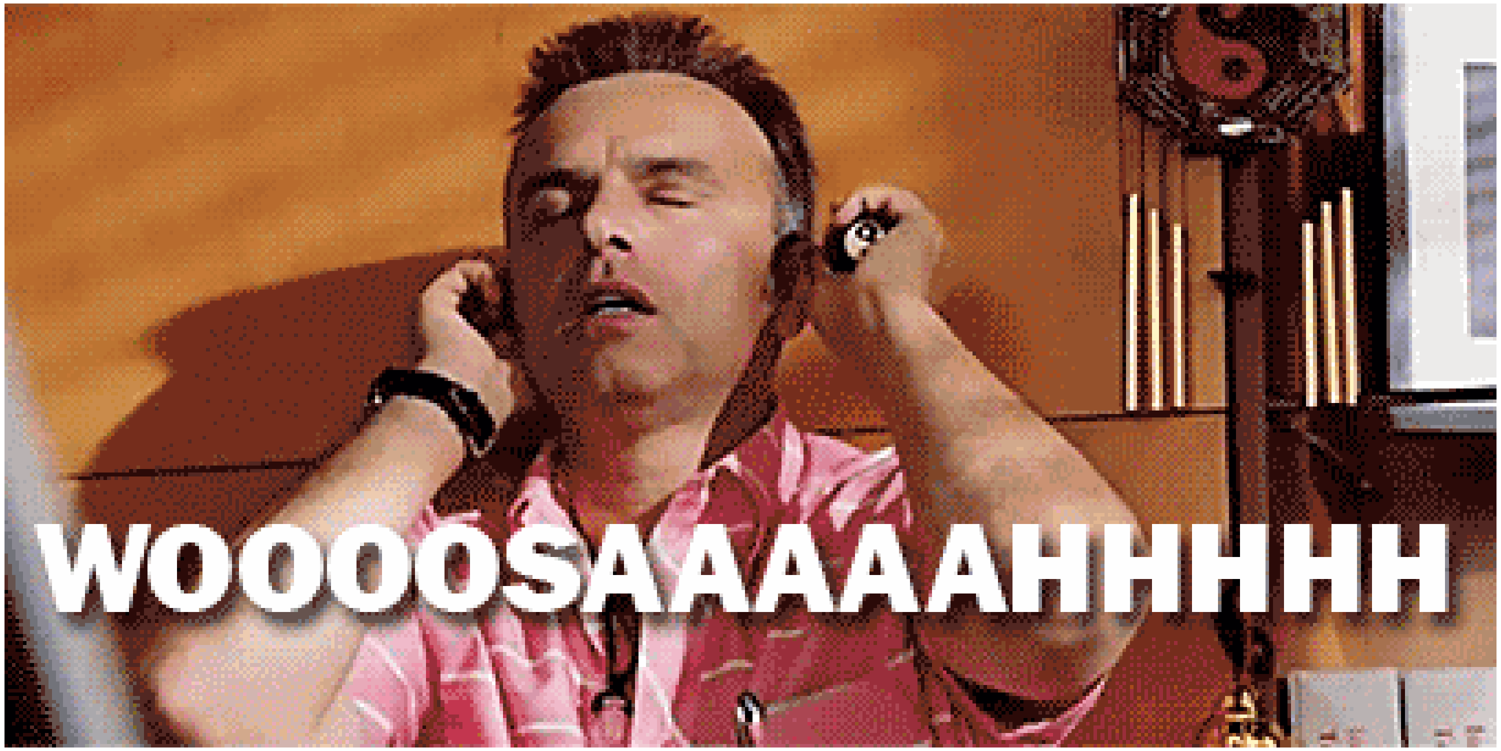
Learning Objectives

- Describe arguments & parameters as they relate to functions.
- Predict values returned by a given function.
- Differentiate control flow between anonymous and named functions.

Agenda

- Intro to Functions
- Function Arguments & Parameters
- Returning Functions
- Anonymous Functions
- Lab Time

Let's breathe for a second...



FEWD Overview

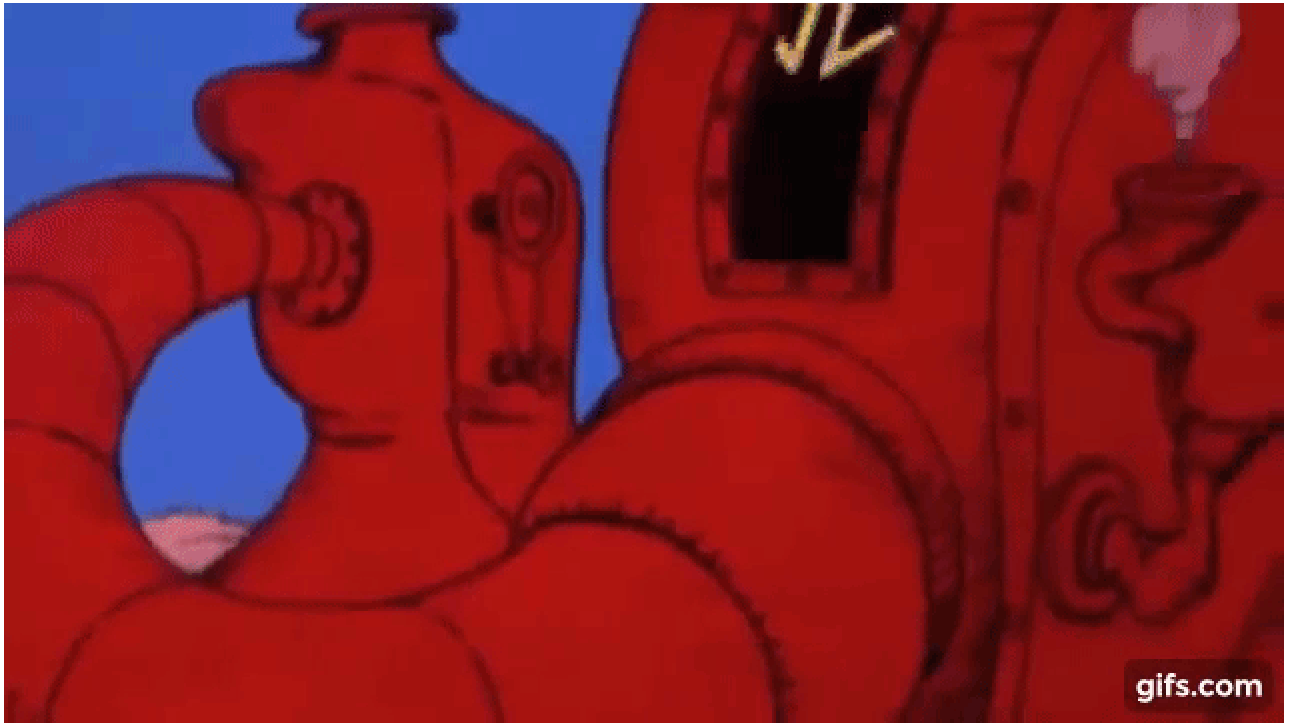
✓ **UNIT 1:** HTML/CSS Basics

✓ **UNIT 2:** Adding Interactivity

✓ **Unit 3:** Building In Concert

- Weeks 1-3
  - Goal: basic structure/design
- Weeks 4-6
  - Goal: JS/jQuery fundamentals
- Weeks 7-8
  - Goal: Responsive Design/Forms
- Weeks 9-10
  - Putting it all together

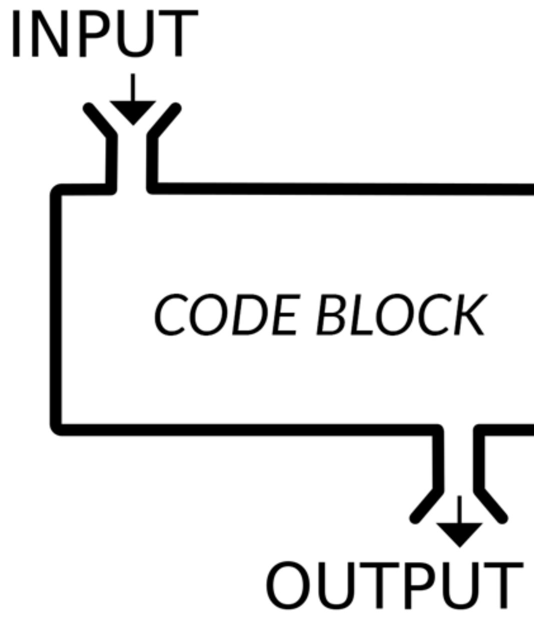
Intro to Functions



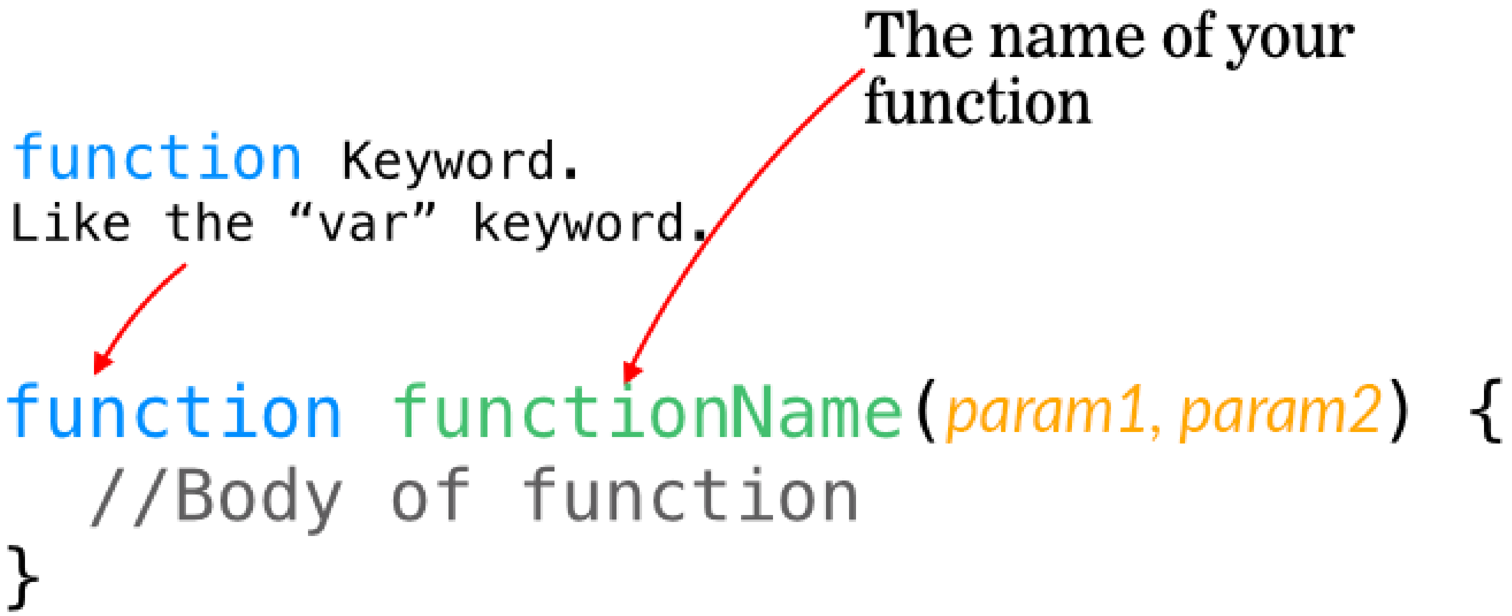
What is a Function?

JS Functions

A **function** is a reusable block of code that performs an action or returns a value.



Function Declaration Syntax



The name of your function

function Keyword.
Like the “var” keyword.

```
function functionName(param1, param2) {
  //Body of function
}
```

Declaring Functions

You can **declare a function** that you plan to use later on in your code by giving it a name.

```
function addNumbers() {
    // Code to add numbers...
}
```

Calling Functions

To **call/invoke** a function, simply type the function name + the parentheses.

This will execute the function immediately.

```
addNumbers();
```

Calling Functions Example

```
function helloWorld() {
  console.log("Hello Functions");
}

helloWorld(); //Prints “Hello Functions to the
console.
```

The brackets execute the function. Try calling the function without them to see what happens.

Pulse Check

Call the function

Let's try calling this function in our console:

```
function helloWorld(){
    console.log("Hello World!");
}
```

Call the function (cont'd)

What happens when you type...

- `helloWorld;`
- `hellowWorld();`

Function Arguments & Parameters

Function Parameters

Function **parameters** are temporary placeholders created when the function is first declared.

These placeholders define what input the function will accept.

Function Parameter Syntax

Parameters let you pass data into the function

```
function functionName(param1, param2) {
  //Body of function
}
```

The functions executed code goes between the { } brackets. Much like an “if” statement.

Function Parameter Example

Let's take a look at this example function with parameters:

```
function fullName(firstName, lastName) {
    console.log("My name is " + firstName + " " + lastName);
}
```

Function Arguments

If parameters are the placeholders in a function, then **arguments** can be thought of as the actual values that are assigned to that placeholder when the function is called.

Function Argument Example

Let's call our previous function, and pass it some string values as arguments:

```
fullName("Mansoor","Siddeeq");
```

Putting it Together

Function parameters

```
function fullName(firstName, lastName) {
    console.log("My name is " + firstName + " " + lastName);
}

fullName("Mansoor","Siddeeq");
```

Function arguments

Important!

When calling your function, be sure the order of your arguments matches up with the order of your parameters.

Order matters!

Code Along

Open: **FEWD Functions Codepen**

Directions:

1. Open the **FEWD Functions Codepen**
2. Run an example using the "Add" option to demonstrate how it should work.
3. Read through the code together

Timing:

- **15 minutes** - As a class, discuss the logic and what's going on...

Returning Functions

Making Functions Useful

Thus far, we've only ever used functions to log something to the console:

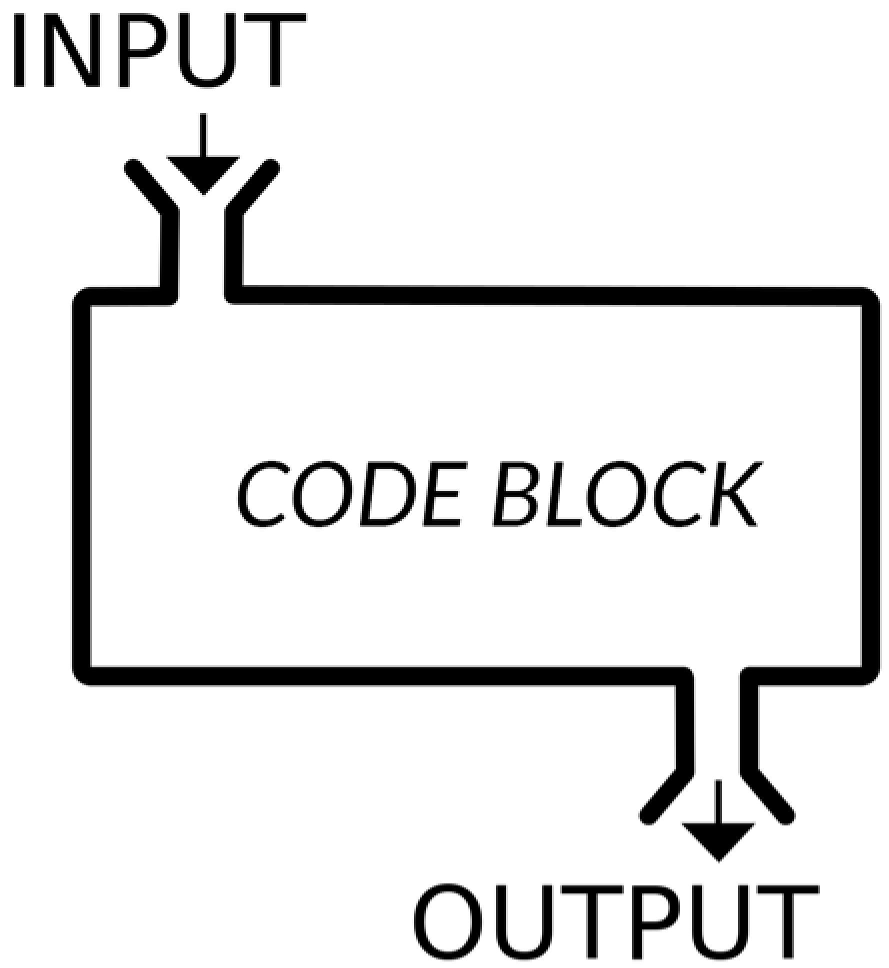
```
function fullName(firstName, lastName) {
    console.log("My name is " + firstName + " " + lastName);
}

fullName("Mansoor","Siddeeq");
```

Or output to the HTML:

```
function addNumbers(num1, num2, num3){
    $('#result').text(num1 + num2 + num3);
}
```

Return Statement

A **return statement** effectively tells the function to stop what it is doing, and "return" a value back to whatever called it -- whether that was a variable, another function, etc.


INPUT
CODE BLOCK
OUTPUT

Return Statement Example

```
function calculateSum(num1,num2) {
    return num1 + num2;
}

var sum = calculateSum(1,2);

sum; // accessing this variable will output 3
```

Return Statement Example 2

```
function calculateAge(currentYear,birthYear) {
    return currentYear - birthYear;
}

function myBio() {
    console.log("My name is Mansoor, and I am " + calculateAge(2018,1987) + " years old");
}

myBio(); // What will be written to the console?
```

Important!

When JavaScript reaches a **return statement**, the function will **stop executing** and exit.

Meaning, if you have code after a return statement, it will not be executed.

```
function myName() {
    return "Mansoor";
    return "Sudi";
}

myName(); // What will the result be?
```

Code Along

Open: **starter_code/cash_register**

Directions:

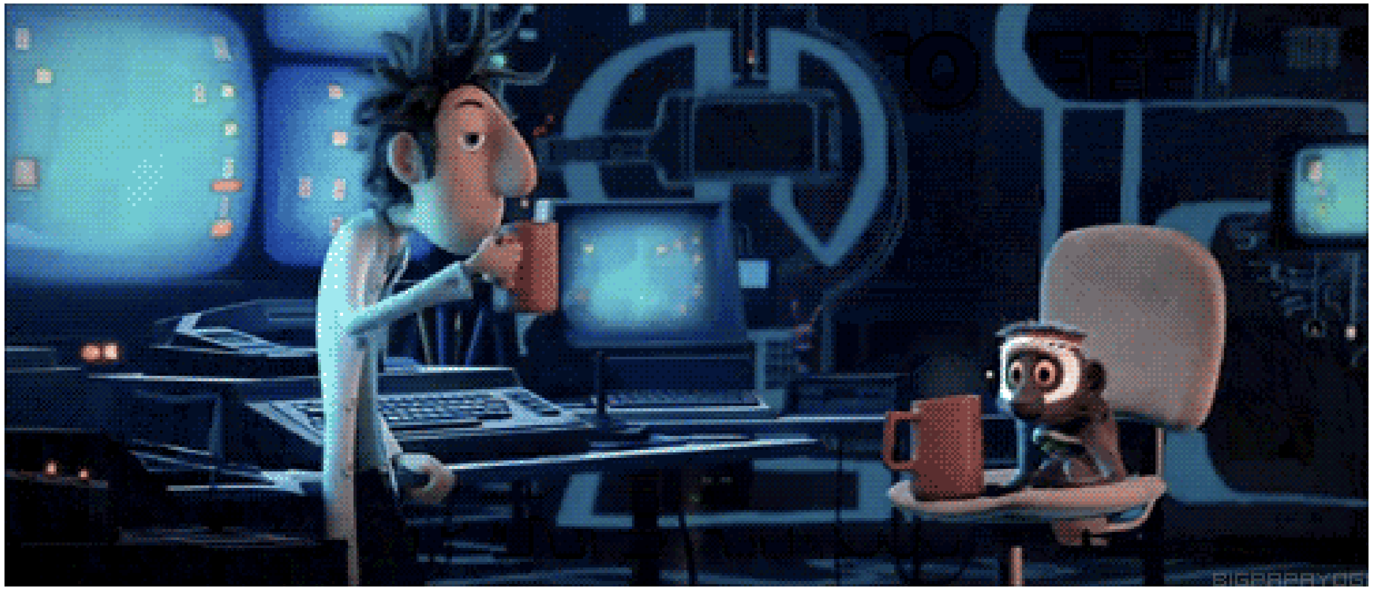
1. Open the **starter_code/cash_register** folder
2. Show how it should work
3. Begin building out the logic of the cash register JS application.

Timing:

- **5 minutes** - write out the pseudocode
- **35 minutes** - As a class, begin building out the JS code to make this lab interactive.

Tips:

- Take it one step at a time! And refer to the pseudocode



Break time!

Let's take 5-10 minutes to decompress...

Anonymous Functions

Defining Functions

There are multiple ways to define a function. So far, today, we've only discussed **function declarations** (aka function statements).

But you can also define a function using **function expressions**.

Whats the difference?

The main differences between a **function declaration** and a **function expression** are:

- When they can be used
- The function name (this can be omitted in function expressions to create anonymous functions)

Function Declaration Example

```
function greeting(){
    console.log("Hello there!");
}
```

Function Expression Example

```
var greeting = function(){
    console.log("Hello there!");
}
```

Calling a Function Declaration Example

```
greeting();

function greeting(){
    console.log("Hello there!");
}

// This works!
```

Calling a Function Expression Example

```
greeting();

var greeting = function(){
    console.log("Hello there!");
}

// This DOES NOT work!
```

Callback Functions

The most common use of anonymous functions are as callback functions -- especially in jQuery.

```
$().click(function(){
    // Do something after a click
});

$().submit(function(){
    // Do something after a form submission
});

$().ready(function(){
    // Do something after you're ready
});
```

Code Along

Open: **starter_code/cash_register**

Directions:

1. Open the **starter_code/cash_register** folder again
2. Let's use anonymous functions/function expressions
3. Observe what happens

Timing:

- **10 minutes** - As a class, update the code to use anonymous functions and/or function expressions

Lab Time

Open: **starter_code/rock_paper_scissors**

Directions:

1. Open the **starter_code/rock_paper_scissors** folder
2. Following your initial pseudocode, build out a basic game of "Rock, Paper, Scissors"

Timing:

- **35 minutes** - begin building out the JS code to make your game interactive
- **10 minutes** - Review solution files as a class

Tips:

- Take it one step at a time!
- Use the pseudocode from README.md file as your guidelines when developing.
- I provided the function that collects the computer's selection for you.

Rock, Paper, Scissors Pseudocode

```
Get available options (rock, paper, scissors)
Get user selection (user types in selection)
Get computer selection (select from available options)
If user selection is the same as computer selection
  then, say "It's a tie"
If user selection is "rock" AND computer selection is "paper"
  then, say "You Lose"
If user selection is "scissors" AND computer selection is "paper"
  then, say "You Win"
If user selection is "paper" AND computer selection is "rock"
  then, say "You Win"
If user selection is "scissors" AND computer selection is "rock"
  then, say "You Lose"
If user selection is "paper" AND computer selection is "scissors"
  then, say "You Lose"
If user selection is "rock" AND computer selection is "scissors"
  then, say "You Win"
```

Exit Tickets

Take 5-10 minutes to give us some

(Link is in Slack Room)

Learning Objectives Review

- We described arguments & parameters as they relate to functions.
- We predicted values returned by a given function.
- We differentiated control flow between anonymous and named functions.

Week 5 HW Due Today

- **Due today by 11:59pm ET**
- Once you upload your HW to your repo, please PM the TA with a link to the Assignment folder.
- This is how we will know whether an assignment has been completed or not
- Once graded, the TA will reply with feedback

Next Class...

Lesson 12 - Arrays, Loops & Refactoring