Reminders

- Attendance is 15% of grade. Grades for attendance is in progress.
- Quiz 5 Due Thursday
- Lab 2 Due Tuesday, October 3 by midnight
- T.A. Hours
  - Monday and Wednesday 4-5 pm or by appointment
- Zoom Class Week 6, September 28th is optional to allow you to attend job fair
  - We will be using the Collaborate Ultra feature on the blackboard.
- The videos for the class will be posted on blackboard under "Video for Sept26"
- Quiz 6 will be assigned by Saturday morning. It will be due by Thursday, Oct. 5.

ITMD 441/541
Web Application Foundations

Week 6

FALL 2023 – SEPTEMBER 25, 2023

This week’s Agenda

- JavaScript Function Revisit
- Lab 2 Revisit
- JavaScript OOP
- Visibility
- ES6+
- Assignments

JavaScript Functions (Revisited)

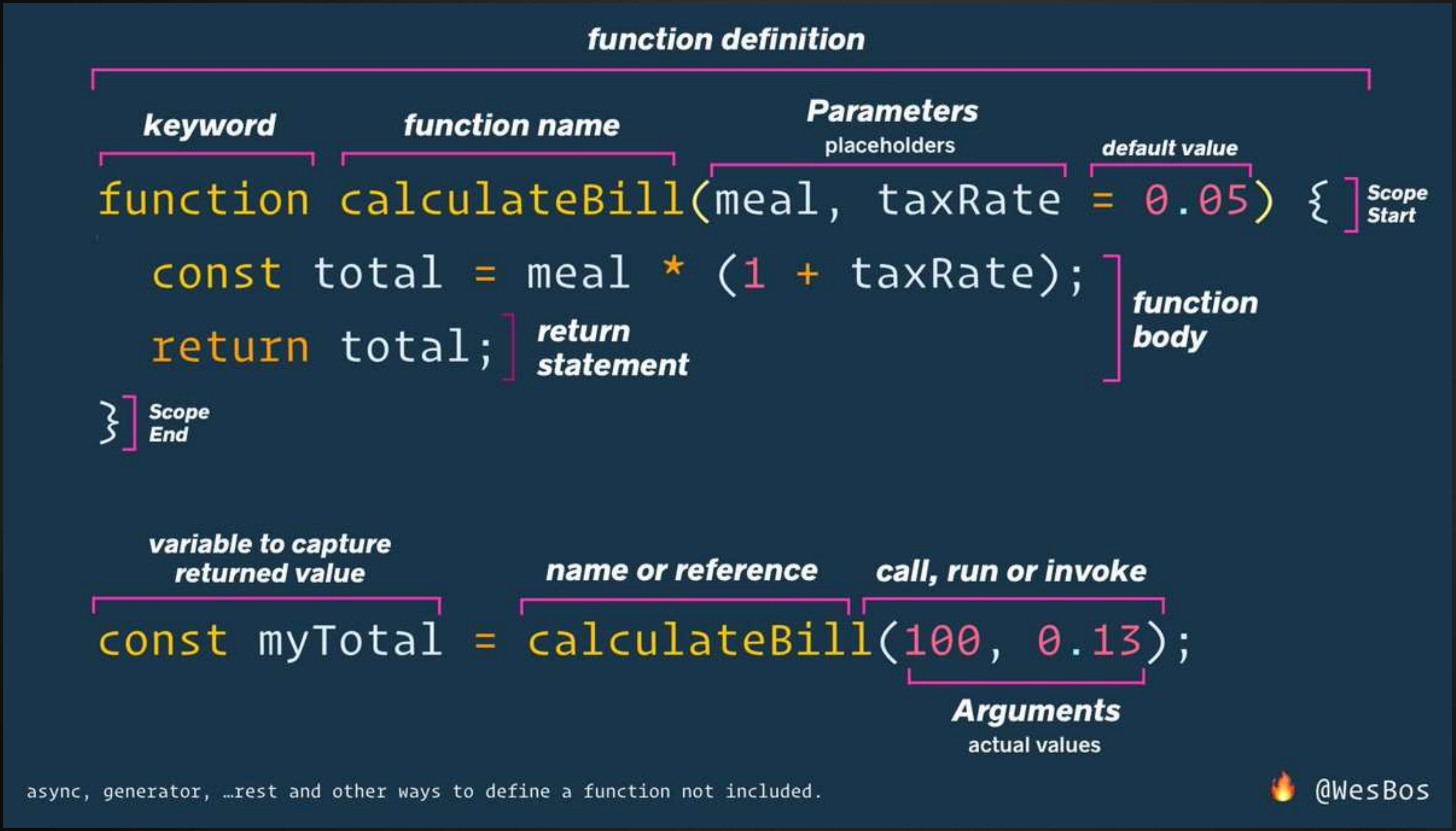
- Named blocks of code that can be called and executed by events or other code, may take parameters
- Functions are objects in JavaScript

```
function funcname (var1, var2, …) {
  code block (may make use of parameters)
}
```

- The return statement will stop executing the function and return the value

```
function addnum(n1, n2) {
  return n1 + n2;
  console.log("This statement will not print");
}
```

JavaScript Functions Examples



```
function calculateBill(meal, taxRate = 0.05) {
  const total = meal * (1 + taxRate);
  return total;
}

const myTotal = calculateBill(100, 0.13);
```

async, generator, …rest and other ways to define a function not included.

@WesBos

Console.log(myTotal)

113

NOTES:

- Parameters are defined for a function and arguments are passed to a function

JavaScript Functions (Revisited)

You can watch the following videos for more information about JavaScript functions:

https://youtu.be/FOD408a0EzU?si=Fw7DvND5oZGn1Cwh

This video is approximately 10 minute. Will not be shown in class but you are responsible for its content.

Lab 2 revisit

- The Zip file is your starting point. It's got all the files you need for this lab.
- When you unzip Lab 2 you will see:
  - asset directory
  - index.html
  - style.css
- Once you have completed the assignment then you will commit all those files to either the repository you created already and just make another folder in it, and put the files there and then submit the URL link to that page. Or if you want to create a separate repository you're welcome to do that. But either way, it still has to be done through Git GitHub pages because ultimately what we're going to do is look at the rendered browser page and make sure it matches what it's supposed to. But it's completely up to you whether you do it in the same repository in just a subfolder or a separate one.

Lab 2 revisit

- Asset directory
  - Do not touch the global.css file. It has the rules to make the lab work.
- index.html
  - Only one change should be made to this file, Your name. DO NOT make any other changes.
- style.css
  - Contains all the selectors you need. You need to write the rules.
  - Below each comment section are images, and basically you need to write the rules to make what you are given match the image. You can get the correct image with the selectors that are in there with just a few rules.
  - You are matching gaps
- Make sure the asset directory and the index.html are submitted with the changes you make for style.css. You are not submitting a link to the style.css alone. It will not work.

JavaScript OOP Patterns

LET'S SIMULATE TRADITIONAL OOP PATTERNS IN JS

JavaScript OOP Patterns

- Patterns to implement Object Oriented Programming in JavaScript
- We will be looking at a few basic ones:
  - Object Literal
  - Constructor Function (Constructor Pattern)
  - Constructor Function with Prototype (Constructor Pattern)
  - Function that returns an object (Factory Pattern)
  - ES6 Classes
- https://leoasis.github.io/posts/2013/01/24/javascript-object-creation-patterns/

Object Member Visibility

WITH CONSTRUCTOR FUNCTION

Public, Private, Privileged

- These are ways we can have private and public members and methods for objects when we model them with the Constructor pattern.
- Functions and variables assigned in a constructor with the `this` keyword will be publicly visible. Functions and variables assigned normally will be private.
- Be careful, the `this` keyword can get bound to the window object so it is common to see a **`var that = this;`** line in the object to bind that to the proper `this` value.
- We will do an example to show these three ideas.
- http://crockford.com/javascript/private.html
- http://robertnyman.com/2008/10/14/javascript-how-to-get-private-privileged-public-and-static-members-properties-and-methods/

JavaScript ES6+

NEXT GENERATION JAVASCRIPT

ES6+

- The 6 edition of the ECMAScript-262 standard for JavaScript
- Also known as ECMAScript 2015
- Was finalized and published in June 2015
- This added significant new syntax and features
- Native support in all modern browsers currently, for most features
- Can use a tool like Babel to transpile to ES5 for compatibility
- New additions are published yearly. Browser support can lag behind slightly.
- https://en.wikipedia.org/wiki/ECMAScript#6th_Edition_-_ECMAScript_2015
- https://kangax.github.io/compat-table/es6/

ES6 Features

- http://es6-features.org/
- ES6 adds some new syntax and features. Some of the bigger changes are:
  - Classes
  - Block-Scoped Constructs let and const
  - Arrow Functions
  - Default Parameters
  - Rest and Spread Parameters
  - Destructuring Assignment
  - Template Strings
  - Multi-line String
  - Maps & Sets
  - Modules
- http://kangax.github.io/compat-table/es6/

Classes

- ES6 added a class syntax similar to other languages.
- Just syntactic sugar, still prototypal under the hood.
- Constructor is used to setup parameters and set properties.
- https://javascript.info/classes
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Classes_in_JavaScript

```
class Song {
  constructor(title, artist, duration) {
    this.title = title;
    this.artist = artist;
    this.duration = duration;
    this.isPlaying = false;
  }

  start() {
     this.isPlaying = true;
  }
}
```

Classes

- Classes can have getter and setter properties/methods
- These allow you to add methods that handle class properties when you use dot notation to get or set their value
- https://javascript.info/property-accessors

```
class Song {
  constructor(title) {
    this._title = title;

  }

  get title() {
    return this._title;
  }

  set title(t) {
    this._title = “new title: ” + t;
  }
}
```

let s = new Song(‘hello’);

s.title; //returns ‘hello’

s.title = ‘goodbye’;

s.title; //now returns ‘new title: goodbye”

Classes

- Classes can have inheritance
- Uses the extends keyword to define a “subclass”
- Still uses prototype inheritance in the background
- https://javascript.info/class-inheritance

```
class RockSong extends Song {
  constructor(title, artist, duration, type) {
    super(title, artist, duration);
    this.type = type;
  }

  logType(){
   console.log(this.type);
}
}
```

Classes

- Public & static class fields
- Newer features, may not be supported in all runtimes. (ES2019)
- Allows you to define public class fields outside the constructor.
- Private class fields would be prefixed with #. Expanded use of that or introducing a private keyword is in discussions.
- Also allows for a static keyword for static fields that are shared on the class

```
Class ClassWithPublicAndStaticFields {
  static staticProperty = ‘aValue’;
  instatceProperty = ‘aValue’
  #privateProp = ‘aValue’;

  static staticMethod() {
    return ‘static method statements’;
  }

  publicMethod(){
    return ‘public method statement’;
  }

}
```

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/Public_class_fields
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/static
- https://www.sitepoint.com/javascript-private-class-fields/

Block Scope

- Remember that when using the `var` keyword to declare a variable there is no block scope, only functional scope.
- ES6 adds two new keywords that declare block scoped constructs
  - `let`
  - `const`
- `let` is similar to `var` but has block scope
- *let is the new var*
- `const` declares an immutable variable that also has block scope. Once the value is assigned it is fixed and can't be changed. You can change the value of object properties or array elements though.

Arrow Functions

- This is a new syntax to declare a function
- Using an arrow function fixes the problems associated with the `this` keyword in some cases. The `this` keyword will have the same value as in the context of the enclosing function. It fixes the problem when creating closures and makes the `that = this` less necessary. Not to be used in all cases, see MDN link.
- If the function executes a single statement, it will implicitly return the result of that statement.

Old Style

```
function add (a, b) {
    return a + b
}
```

New Style

```
const add = (a, b) => a + b;
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

Default Parameters

- Default parameters are parameters to a function that are given some default values in the function declaration.
- The value can be changed when calling the function

```
const add = (a = 5, b = 6) => {
     return a + b;
}


function add(a = 5, b = 6) {
     return a + b;
}
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Default_parameters

Rest Parameter

- You can use the rest prefix of … to extract the rest of the args from a function.
- This lets you accept unlimited number of parameters and process them dynamically.
- They come in as an array and only has the arguments that were not provided explicitly.

```
function add(a, b, ...more) {
     // more variable in this block is an array of as many
params that were passed
}
add(5, 6, 8, 2, 5, 6);
```

- Inside the add function more is.   [8, 2, 5, 6]

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/rest_parameters

Spread Operator

- Similar to the rest parameter in look but functions differently
- It spreads out the values of an array or iterable to separate elements
- Uses the … before the variable name

```
var params = [“hello”, true, 7];
var other = [1, 2, ...params ];  // results are [1, 2, “hello”, true, 7]
var val = [1, 2];
function add(a, b) { return a + b }
add(...val)    // outputs 3
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax

Destructuring

- Destructuring allows you to extract values from arrays and object
- Uses the array brackets or object curly brackets

```
let arr = [5, 8, 2, 5];
let [x, y, z, w] = arr;
// x is 5, y is 8, z is 2, w is 5
```

```
var o = { p: 42, q: true };
var { p, q } = o;
// p is 42 and q is true
```

- Can rename obj properties to new variable names

```
var {p: foo, q: bar} = o;
//now foo is 42 and bar is true
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

Template and Multi-line String

- Use the back tick (`) to define and wrap the string.
- This is for both template strings and multi-line strings.
- With template strings you can then evaluate variables in the string with `${}`
- Multi-Line String
```
var text = `This is a mult line string that
continues on to the next line.`;
```
- Template String
```
var name = ‘anita';
var greeting = `Hello ${name}!`;
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

Maps and Sets

- Map
  - Holds key-value pairs similar to an object but keeps track of insertion order. Maps are iterable. Keys don't have to be strings but can be any object.
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map
- Set
  - Collections of unique values of any data type. Sets are iterable and also keep track of insertion order. A value in a set can only occur once.
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set
- https://javascript.info/map-set

Assignments

Reading/Assignments

- Quiz 5 – Week 5 Content – Due end of day September 28
- Reading:
  - https://javascript.info/classes
  - Objects in MDN:
    - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics
    - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object_prototypes
    - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_programming
    - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Classes_in_JavaScript
  - Work to read though section 4 of eloquent javascript
- Quiz 6 – Week 6 Content – Assigned this weekend
- Lab 3 – Coming soon