What are functions?

Algorithm ( sequence of steps)

What do functions do?

Do verbs such as 1 + 1 and the verb is “plus”

Do a sequence of steps

Allow us to name that sequence of steps

When do we use functions?

When you have a related sequence of steps

When we’re going to reuse something

When naming a chuck of code helps us understand what that code does

Why do we use functions?

Reason about the business logic

We’re able to code as close to natural language as possible

What does return do?

Return gives us a value as output

Return vs. console.log at the end of a function

Stops the function

Return vs. console.log at the end of a function (or else you won’t see that return value)

function isOdd (number){

var output = (number % 2 == 0) ? true : false;

return output;

}

function isOdd(number) {

var output = (number % 2 == 0) ? true : false;

console.log(output);

}

var resuilt = isOdd(2); (this is called calling a function)

console.log(result);

the first isOdd will work better because you can actually do something with return output but not with console.log, you can console.log but outside the function

remember to push everything to GitHub

Iffe’s go at the beginning and the end of the document. You want to add them when you’re done with the document. It helps to protect your code.

Review:

Functions and finish challenge

review up until Splice for Tuesday night 3.5.2 (Manipulating Arrays)

Work on ecommerce website

Whitney’s Challenge:

Write a function that takes in an array as an argument and returns the average of the numbers in the array

Notes on Arrays:

Imagine each piece or element of an array , each element on an array is its own value

An array is also a value so you can have an array within an array

We do plural variable names for arrays

Var colors = [“papayawhip”, “honeydew”, “blue”, “hotpink”, salmon]

Colors [3]

“hot pink”

“banana”.length

6

colors.length

5

colors[colors.length]

colors[colors.length – 1]

colors[colors.length – 2](this is to access the second to last element on the array)

“hot pink”

var shapes = [“circle”, [“rectangle”, “square”]]

shapes

the number in the brackets is called an index also called key or keys

toppings

[“pineapples”, null , “olive”, “chile”]

toppings[1] = “jalapenos”

toppings

[“pineapple”,”jalapenos”, olive, “chile”]

for (var i = 0; i < toppings.length; i += 1){

console.log(toppings[i]

}

to continue printing everything until it has printed everything on the array

“banana”

“banana”[0]

b

“banana” [1]

a

(function(){

"use strict";

// TODO: Create an array of 4 people's names using literal array notation, in a variable called 'names'.

var names = ["Sarah","Carla","John","Tristan"];

// TODO: Create a log statement that will log the number of elements in the names array.

console.log(names.length);

// TODO: Create log statements that will print each of the names array elements individually.

for (var i = 0; i < names.length; i += 1){

console.log(names[i]);

}

var element;

element = names [0];

console.log(element);

element = names [1];

console.log(element;

element = names[2];

console.log(element);

element = names [3];

console.log(element);

or

var element;

for (var i = 0; I <names.length; I += 1){

element = names[i];

console.log(element);

}

var names = [“John”,”Paul”,”George”,”ringo”;

For Each Loop Iteration

forEach loop that runs on an array

names.forEach(function(element, index, array){

console.log (element == array[index]);

console.log (element == names[index]);

console.log(element);

someArray.push(“third place.. so sad”);

3

someArray

[“first prize”,””second place”, “third place.. so sad”]

someArray.push(“Bob);

[“first prize”,””second place”, “third place.. so sad”, “Bob”]

someArray.unshift(“the real first place”)

5

someArray

[“the real first place,”first prize”, “second place”, “third place ..so sad”, “Bob”]

unshift = add to the beginning of the list

someArray

someArray.pop()

“Bob”

Remember the following terms:

push = adds a value or values at the very end of the array // someArray.push()

pop = removes from the end of the array // someArray.pop()

shift = removes the first element //someArray.shift()

unshift = adds to the beginning of the array //someArray.unshift()

**Splicing**

We have seen how to add and remove elements from the beginning and end of arrays. However, sometimes you need to add or remove elements from somewhere in the middle of an array. Splicing is one of the most powerful array manipulation methods. It allows you to add or remove elements to/from any location in an array, or even replace elements of an array.

The splice method returns an array of the items that were removed or an empty array if no items were removed.

});

Splicing:

myArray = [1,2,3,4,5]

[1,2,3,4,5]

myArray.splice(0,0,7)

[]

myArray

[7,1,2,3,4,5]

myArray

[7,1,2,3,9,,42,,654,4,5]

myArray.splice(myArray.indexOf(2) + 1,2,654)

[3,9]

myArray

[7,1,2,42,654,4,5]

review for Wednesday :

split-join html practice bonus – finish it

Objects Lecture:

Objects: are a way of wrapping data and state

Objects combine state and behavior

Properties, color is a property like a color of a shirt or the number of doors a car has

Each property has its own variable

Methods is the vocab word for a function that lives on an object like a car can honk

Because we model behavior, we can represent our physical notions of things as a single variable with behavior or methods on it. Those methods only run on a particular object

Objects live inside of curly braces

Like

“banana”.length

6

“banana”.toUpperCase()

“BANANA”

var dice = {}; this means we are defining an empty object

undefined

dice.value

undefined

dice.value

undefined

dice.value = 6

6

die.value = 3

3

Treat properties just like variables

If you do:

dice.color

undefined

dice.color = “red”

“red”

dice.numberOfSides = 6

6

dice.numberOfSides

6

var dice = {};

dice.value = 1;

dice.numberOfSides = 6;

dice.color = “red”;

if we just had var color, var sides, we wouldn’t know what we’re talking about so that’s why the properties live in the objects like dice.value , dice.numberOfSides, dice.color

We have key value pairs

Value is a key

Number of sides is a key or a property

Color is a key or a property

Keys are just names

Let’s pretend we’re each an object and we will all have a property of name or hair color

Function rollDice() {

}

var rollFunnyDice = function () {

return Math.ceil(Math.random() \* 6);

}

var dice = {};

dice.value = 1;

dice.numberOfSides = 6;

dice.color = “red”;

die.rollDie = function () {

return Math.ceil (Math.random () \* this.numberOfSides);

};

var die2 = {};

die2.value = 1;

die2.numberOfSides = 6;

die.color = “red”;

die.rollDie = function ()

return Math.ceil (Math.random() \* this.numberOfSides);

};

var firstRoll = die.rollDIe();

var secondtRoll = die.rollDIe();

var thirdtRoll = die.rollDIe();

var die2 = {};

die2.value = 1;

die2.numberOfSides = 6;

die.color = “red”;

die.rollDie = function ()

return Math.ceil (Math.random() \* this.numberOfSides);

};

or another way of saying this:

var die = {

“value” : 1;

“numberOfSides” : 6

“color” : “red”,

“rollDie”:function(){

return Math.ceil(Math.random() \* this.numberOfSides);

}

};

“this” refers to dice

Every html element is an object

HTML elements are objects

JSON is data interchange format

boolProp means Boolean property

var car = {

“make” : ,

“model” : {

“name” : “sonata”,

“suspension” : “sport”,

“trim” : null

},

“year”:1998,

“miles”: 200000,

“mpg”: 19,

“color” : “blue”,

“currentOwner”: {

“firstName” : “Ryan”,

“lastName” : “Orsinger”

},

“previousOwners”: [

{

“firstName” : “Bob”,

“lastName” : “Bobberson”

}

{

“firstName” : “Dave”,

“lastName” : “Davies”

}

]

};

We have [] because we are making an array and we are just sending data, properties. We don’t do functions in JSON. Nesting happens over and over.

For exercise on objects, we will run into problems if we are missing commas and make sure to write valid JSON

Review:

From planets-js.html

//converts a string that represents a 12 hour time to a string that represents

//a 24 hour time

function twelveToTwentyFour (timeString){

timeString = timeString.split(":")

}

console.log(twelveToTwentyFour(12:30pm))

twelveToTwentyFour('4:30 pm');//Should return '16:30'

twelveToTwentyFour ('12:22 pm');//should return '12:22'

twelveToTwentyFour ('12:45 am');//should return '0:45'

twelveToTwentyFour ('9:00 am');//should return '9:00'

Thursday Review from Gist on Thursday:

What methods change the original variable?

ForEach Loops – 5ws + h:

Foreach loop, use it when you’re going to do something to every element in the array. One of the big benefits is looping through a list , say of names, our code reads more cleary and we don’t have to worry about the index. We want to have our parameters be named something other than something so generic. The body of this is the body of a function, we can put any code we want to execute, our break and continue will work inside a for loop but not inside a foreach loop. We don’t have a way to break out from a foreach loop at once. If you need to stop the loop it may be better to just use a for loop. Or you can check for a condition, if some condition is met then return and be done with the function. This will not change the original names variable (array), it will only change in the scope of the function if you add anything. Foreach are only for arrays, not objects.

someArray.forEach(function (element, index, array) {

});

Break + continue , where to use? Best practices:

Continue , you can accomplish what you want with this or with an if statement, but this essentially skips to the next iteration of the loop and go back to the open and curly brace. My condition will still be evaluated. Continue says to forget everything else in the loop and start back up.

For (var i = 0; I < 10 ; i++){

If (i===4){

Break;

}

console.log(i);

}

what we just did would give us a list from one to 9 but it would skip 4.

Whatever makes your code more readable is when this is the best time to use it.

Stringing stuff together nested properties in objects

Json – syntax, what is valid json? Best practices: this will be covered later

Splicing

array.splice(startIndex, numberOfItemsToRemove, itemToInsert1, itemToInsert2, ..)

var index = array.indexOf(something);

//remove 2 elements

array.splice(index,2);

//insert 2 elements

array.splice(index, 0, item1, item2);

we want to make sure we’re commenting our intent to use splice and not do too much at once.

js – if else tying everything together – what good are all these books if we’re only

console.logging

multipages of js

Everytime we reload javascript, javascript doesn’t remember the previous or future executions. If we will have multiple pages talking to each other, we will do this on the back end. As far as communicating this data, this book data, we will learn to have that information in a separate file when we get to JQuery.

scope of “this” keyword:

var me= {

name: “zach”,

sayMyName : function() {

return this.name;

}

}

“this” refers to the object that called the method that this is inside of

manipulating the stuff in a foreach loop. You can also say return me.name but we use this to be a little clearer and makes our code more generic.

When to use alert, confirm, prompt

We will use but not a whole lot. These provide us a basic set of functionality, to make our user experience better.

Playing computer:

Taking a look at a piece of code and playing computer in your head, pretending you’re executing this code and not the computer. Think through what all the values are and what is happening.

Multiline strings, number.max, min value

var myString = ‘hello’

world’; // => ‘hello world’

var myString = ‘hello’;

myString += ‘world’;

myString += ‘world’;

myString += ‘world’;

myString += ‘world’;

myString += ‘world’;

Having multiple console.logs is probably a good indication to write a comment

Control Shift K deletes the line in sublime

/////////////////////////

function returnThree () {

return 3;

}

[1, 1 + 1, returnThree(), returnThree() + 1]

//array literal syntax

//[element1, element2, element 3, …..]

//where element is any valid javascript expression

//object literal syntax

//{key1 : value1, key2:value2, key3: value3, …}

{

one:1,

two: 1+ 1,

three: returnThree(),

four: returnThree() + 1

}

{“property-name” : value}

//where value is any valid javascript expression

//key or propertyName must be a valid identifier or string literal

var a = 123;

var b = a (whatever is an “a” right now is what “b” will be , and b will not change later even if the value of a changes later on. It’s only at this point in time)

var person = {name: ‘Robert’, age:99}

var age = people[4].bio.age;

console.log(‘person.age: “ + person.age + “age:” + age); //person.age: 99 age :99

// JSON: JavaScript Object Notation –json.org

//data interchange format of the modern web e.g. mysql, pg datatypes

//reddit api, github api, pokemon api

//syntax: no logic , just expressing data

//subset of objects we use in our js

//indentation conveys nesting

//values can only be literals

//property names must be strings (must be” “)

//all strings must be double-quoted

//no trailing commas

//no comments

when we write construct json to send, it won’t have comments

databases store different data types, you can store an integer, a float, one of the possible types in the newer version in MySQL is JSON

student.cars[1].make to get the second item in the array

Review for Thursday:

\*Get comfortable with ForEach Loops

Do the math-js.html (circle exercise)

Defuse the bom exercise

Ahead, learn to do the html document like it was done for us in the defuse the bom exercise

The window object gives us access to all the html and the css in addition to being able to talk to the actual tab like the prompts if we like pizza, we were accessing the actual page, the actual window object. JavaScript has access to everything on the page and the prompts but not the bookmars, history , etc.

Settimeout calls a function or evaluates an expression after a specified number of milliseconds

setInterval() writes a function over and over inside a loop in a time loop and you can control how long that thing executes

setTimeout (function() {

console.log(“Eggs are done!”);

} , 1000);

3

Eggs are done!

4

setTimeout returns an integer that is an identifier for that particular time out ,

var shouldEndTimerEarly = false;

function ding() {

if (shouldEndTimerEarly) {

clearTimeout(timeoutId);

console.log(“ding, ding”);

} else {

console.log(“ding, ding”) ;

}

}

var timeoutId = setTimeout(ding, 1);

review :

Nodes and why I got the error of not a function (dom-query-js.html)

Do calculator problem

Go over all lessons we covered

GIT COMMIT