Discussion Board 7.1 – Classes

Eric McCool

Bellevue University - College of Science and Technology

Web231-339A Enterprise JavaScript I

Professor Krasso

November 29, 2022

This week we look at classes in JS, what they are, what constructors are, why you would use classes in JS, and what some advantages and disadvantages to using classes might be.

First, what is a class? A class is a new level of abstraction and a 'blueprint' to create an instance of an object. A class is different than a data-structure in that it contains data that influences the behavior of the object, while a data-structure just holds data. For instance, the string "Hello World!" is just data. It does nothing *to* the data. It just represents the data. A class might hold the string, as well as other data-types, but these properties can be changed with its own class methods. For example, a class would build an instance of a bank customer, giving them a name, an address, and an account balance by using its constructor. It has now constructed an object named 'Bob' with an address and a balance of $40. A class method would then change the balance every time money is withdrawn or deposited by way of its methods, while a data-structure of 'balance = $40' has no methods of its own. It would be changed based on function calls not held within a class, or an iterative loop like we have seen using i++. Remember, a function is on its own and a method is a function on an object. In short, I think of the class as 'behavior' and a structure as 'data.'

While JS was designed as a scripting language, it has matured into a full-fledged programming language. Scripting sounds different than object-oriented, but at the end of the day, everything in JS is an object, yes? A string is an object, a variable is an object, etc. Therefore, *by definition*, JS could (and probably should) be considered an object-oriented language with the addition of the *class* keyword. What is a class? It is a blueprint for an object.

Why would we use a class in JS? Well, there seems to be many arguments both for an against if you look online. I found dozens of examples, but I think I will just freestyle this one. While classes in JS can be made very difficult to understand and use if one so chooses (apparently binding and the keyword 'this' are the biggest drawbacks I have found), they can also be written cleanly and simply to help with readability and modularity. While I do not have a bunch of experience with classes in JS, yet, I have found them great to use in other languages. For example, it allows code to be separated into different files, making a crazy-long program much shorter and easier to differentiate and read. Using the bank customer from above, if I were to put that class into its own file, I could then re-call (import) that object constructor later on if I was creating a different app and needed a bank customer, without having to rewrite everything. Mmmm…reusability. I can be lazy like that.

Since I touched on the keyword 'this' a moment ago, I will try to explain it a little from my own practical use and understanding. I hope this helps. Take the constructor:

constructor customer (title, balance) { ← title and balance are parameters.

let this.name = title; ← the argument passed into the constructor through the parameter is on the right-side of the assignment operator. (Sometimes our examples in books and online will use 'name' as the parameter, and name for everything else. This confuses the heck out me. By using title, I hope I made it easier to follow).

On the left-side of the assignment operator, we are saying that 'name' (the title parameter who's argument is 'Bob', in this case) that was passed to the constructor is now the 'name' for 'this' object – and ***only*** this 'Bob' object. Bob was probably assigned to a new variable or map or array, or whatever (data-structure), when the call came to make a new bank customer object. So that data-structure has a name property now with a value of 'Bob.' When the next call to create a new bank customer comes in for a Carl-object, Carl will be the argument passed to the constructor's title parameter. It will then be assigned to 'this' name property for the new object only – not the Bob-object from earlier.

This is why I like classes. I can save them as their own files. They are like their own little programs *outside* the main program that I can call and use in *any* program that I write in the future. I just have to import it into my new program without re-writing all the code that is in them. The hardest part is just remembering where I saved it so I can use it again =)

As my disclaimer stated, I am just freestyling this one. I have no references. I am going strictly by my understanding of classes and objects from previous programming courses and practical experience in working with them. I have studied a couple other purely OOP languages. If my logic is flawed, don't hesitate to change my mind. Be good.