Object Oriented & Functional Programming Notes

JavaScript object with attributes:

var car = {

"wheels":4,

"engines":1,

"seats":5

};

# Examples

## Constructor function:

var Car = function() {

this.wheels = 4;

this.engines = 1;

this.seats = 5;

};

Note: This refers to the new object being instantiated.

### Calling constructor function

Call with the “new” keyword in front of the constructor name:

var myCar = new Car();

myCar is an instance of Car that has the same attributes that were described for Car. Once created, myCar’s properties can be accessed and modified as per usual. For example, myCar.turboType = "twin"; adds the property turboType with a value of “twin”.

### Make Unique Objects by Passing Parameters to our Constructor

Parameters can be added to the constructor. For example:

var Car = function(wheels, seats, engines) {

this.wheels = wheels;

this.seats = seats;

this.engines = engines;

};

This allows us to pass arguments into it when calling our constructor. Ie: var myCar = new Car(6, 3, 1);

The myCar object will look like this:

{

wheels: 6,

seats: 3,

engines: 1

}

## Make object properties private

Create the variable inside the constructor using var instead of creating it as a property of .this .

var Car = function() {

// this is a private variable

var speed = 10;

// these are public methods

this.accelerate = function(change) {

speed += change;

};

this.decelerate = function() {

speed -= 5;

};

this.getSpeed = function() {

return speed;

};

};

## Map

var oldArray = [1, 2, 3];

var timesFour = oldArray.map(function(val){

return val \* 4;

});

console.log(timesFour); // returns [4, 8, 12]

console.log(oldArray); // returns [1, 2, 3]

Note: In our example the callback only uses the value of the array element (the val argument) but your callback can also include arguments for the index and array being acted on.

## Reduce

To use reduce you pass in a callback whose arguments are an accumulator (in this case, previousVal) and the current value (currentVal).

The accumulator is like a total that reduce keeps track of after each operation. The current value is just the next element in the array you're iterating through.

reduce has an optional second argument which can be used to set the initial value of the accumulator. If no initial value is specified it will be the first array element and currentVal will start with the second array element.

**var** array **=** **[**4**,**5**,**6**,**7**,**8**];**

**var** singleVal **=** 0**;**

// Only change code below this line.

singleVal **=** array**.**reduce**(function(**array**,** singleVal**)** **{**

**return** array **+** singleVal**;**

**},** 0**);**

## Return

Use filter to create a new array with all the values from oldArray which are less than 6. The oldArray should not change.

var oldArray = [1,2,3,4,5,6,7,8,9,10];

// Only change code below this line.

var newArray = oldArray.filter(function(val) {

return val < 6;

});

## Sort

Sort elements from smallest to largest:

var array = [1, 12, 21, 2];

array.sort(function(a, b) {

return **a - b**;

});

Sort elements from largest to smallest:

var array = [1, 12, 21, 2];

// Only change code below this line.

array.sort(function(a, b){

return b - a;

});

## Reverse

var myArray = [1, 2, 3];

myArray.reverse(); //Returns [3, 2, 1].

Another ex:

var array = [1,2,3,4,5,6,7];

var newArray = [];

newArray = array;

newArray.reverse();

## Concat

//Concatenate concatMe to the end of oldArray. This will be assigned to newArray.

var oldArray = [1,2,3];

var newArray = [];

var concatMe = [4,5,6];

newArray = oldArray.concat(concatMe);

## Split

var string = "Split me into an array";

var array = [];

// Split into an array where there are spaces in string array.

//New array will be [“Split”, “me”, “into”, “an”, “array”]

array = string.split(" ");

## Join

var joinMe = ["Split","me","into","an","array"];

var joinedString = '';

// This results in joinedString being a string equaling “Split me into an array”

joinedString = joinMe.join(" ");

# Definitions

Constructor function: A form of object creation. It is capitalized to make it clear it is a constructor.

Properties: An object’s attributes.

Methods: object’s functions.

private properties and private methods aren't accessible from outside the object.

Map method: will iterate through every element of the array, creating a new array with values that have been modified by the callback function, and return it. Note that it does not modify the original array.

Reduce - array method allowing you to iterate through an array and condense it into one value.

Filter - method is used to iterate through an array and filter out elements where a given condition is not true. filter is passed a callback function which takes the current value (we've called that val) as an argument. Any array element for which the callback returns true will be kept and elements that return false will be filtered out.

Sort: Method sorts the values in an array alphabetically or numerically. Sort ALTERS the array in place.

sort can be passed a compare function as a callback. The compare function should return a negative number if a should be before b, a positive number if a should be after b, or 0 if they are equal.

If no compare (callback) function is passed in, it will convert the values to strings and sort alphabetically.

Reverse: Reverse elements of an array. Alters the array in place.

Concat: concat can be used to merge the contents of two arrays into one. concat takes an array as an argument and returns a new array with the elements of this array concatenated onto the end.

Split: Splits string into an array. split uses the argument you pass in as a delimiter to determine which points the string should be split at.

Join: join each element of an array into a string separated by whatever delimiter you provide as an argument.

# FreeCodeCamp Sample

## Make object properties private

Bike **=** **function()** **{**

// Step 1: Set "var gear" - the challenge states to leave it undefined.

**var** gear**;**

// Step 2: The setGear method should "set" the gear equal to whatever the input is. In this case it's "change".

**this.**setGear **=** **function(**change**)** **{**

gear **=** change**;**

**};**

// Step 3: The getGear method should "get" the gear, so you basically just ask it to return "gear".

**this.**getGear **=** **function()** **{**

**return** gear**;**

**};**

**};**

**var** myCar **=** **new** Car**();**

**var** myBike **=** **new** Bike**();**