Contents

[1. JavaScript as an Object Oriented Language 2](#_Toc451106271)

[1.1 Ways to create a JavaScript object 2](#_Toc451106272)

[1.1.1 Object Literal Pattern 2](#_Toc451106273)

[1.1.2 Build-in Object Prototype 2](#_Toc451106274)

[1.2 Object Prototype Default Functions 3](#_Toc451106275)

[1.3 Creating a Prototype or Class in JavaScript 3](#_Toc451106276)

[1.4 Using the prototype property 3](#_Toc451106277)

[1.5 Property Scope – Public vs Private! 3](#_Toc451106278)

[1.6 Object-Oriented Definitions for JavaScript 4](#_Toc451106279)

[1.7 Namespaces!!!! We all need them to remain sane! 4](#_Toc451106280)

[1.8 IFFE – Executing your Objects immediately!!! 4](#_Toc451106281)

[1.9 Inheritance!!! 5](#_Toc451106282)

[1.10 Putting it all together! Best practice for defining a prototype / class 5](#_Toc451106283)

[2.0 JQuery – The most fun you can have with JavaScript on the Web 6](#_Toc451106284)

# JavaScript as an Object Oriented Language

JavaScript is a prototype-based, object-oriented programming language. In JavaScript, everything is an object, and you either create a new object from nothing, or you create an object from a clone of an existing object, known as a prototype.

## Ways to create a JavaScript object

### 1.1.1 Object Literal Pattern

This mode of object creates an object from nothing. The example below creates a car object with some properties.

var car1 = {

year: 2000,

make: ‘Ford’,

model: ‘Fusion’,

repairs: [‘repair1’, ‘repair2’, ‘repair3’],

getInfo: function() {

return ‘Vehicle: ‘ + this.year = ‘ ‘ + this.make + ‘ ‘ + this.model;

}

}

### Build-in Object Prototype

You can create an object using a new keyword. This clones the Object prototype in JavaScript. The benefit of creating an object this way is that you inherit all the properties of the Object prototype.

Function CreateVehicleObject(theYear, theMake, theModel) {

var vehicle = new Object();

vehicle.year = theYear;

vehicle.make = theMake;

vehicle.model = theModel;

vehicle.getInfo = function () {

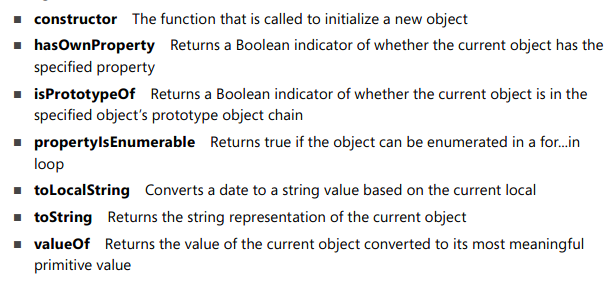
return ‘Vehicle: ‘ + this.year + ‘ ‘ + this.make + ‘ ‘ + this.model;

};

return vehicle;

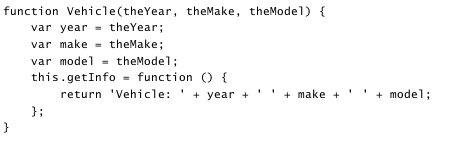
}

## 1.2 Object Prototype Default Functions



## 1.3 Creating a Prototype or Class in JavaScript

JavaScript simulates a class by creating a constructor function for an object.



The class above is instantiated using the following syntax.

var car1 = new Vehicle(2000, ‘Ford’, ‘Fusion’);  
var vehicleInfo = car1.getInfo();

## 1.4 Using the prototype property

In JavaScript everything has a prototype property. A prototype is an object containing properties and methods that should be available to all instances of the object type you are working with.

When you use the prototype property you will not have access to private variables. So, there are two ways that you can declare an object to allow the use of prototype properties. The first is using private variables with getters and setters. The second is using public variables. The example in the link below shows an example of each.

Example: <https://jsfiddle.net/7chf01me/1/>

## 1.5 Property Scope – Public vs Private!

* Variables that do not have “var” in front of them are automatically declared as global variables.
  + function Example () {

this.test = ‘test’;

}

The test variable is global within the program.

* Variables with “var” infront of them are local variables to that function.
  + function Example () {

var test = ‘test’;

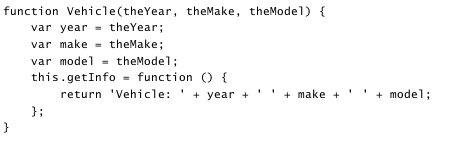
}

The test variable is local to the Example () function within the program.

## 1.6 Object-Oriented Definitions for JavaScript

**this:** This is used to reference an object that owns the current code. Properties specified using “this” are considered public scope.

**Privileged Method:** A method that is public but has access to private data. In the image below, getInfo is a privileged method.



## 1.7 Namespaces!!!! We all need them to remain sane!

You can use namespaces in JavaScript!!!

Common practice for initializing a namespace is as follows:

var namespace = namespace || {};

You can add variables to the namespace by saying namespace.MyVariable = “namespace variable”, where MyVariable is the name of your variable.

You can add functions to the namespace by saying namespace.MyFunction = function () { return this; };, where MyFunction is the name of your function.

You can call the variables and functions in your namespace by saying namespace.MyVariable and namespace.MyFunction().

You can have a namespace inside of a namespace. An example of this is as follows:

Namespace1.namespace2.MyVariable = “test”;

## 1.8 IFFE – Executing your Objects immediately!!!

An IFFY (pronounced iffy) is an anonymous function that gets executed.

An example of using this is as follows:

( function() {

// this gets executed immediately as an anonymous function

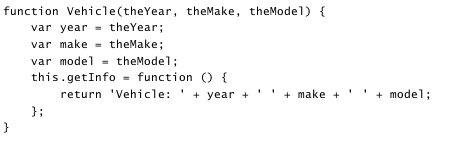
} ();

);

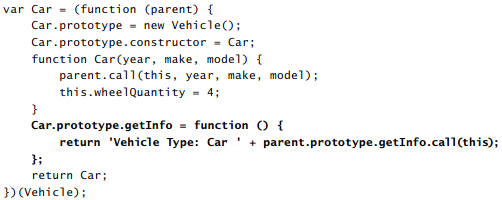
## 1.9 Inheritance!!!

You can do inheritance in JavaScript!! Pretty exciting stuff considering its supposed to be an OO framework! It’s actually really easy to do as well! Not as easy as writing a simple keyword tho :p

First you define a base class prototype as shown below.

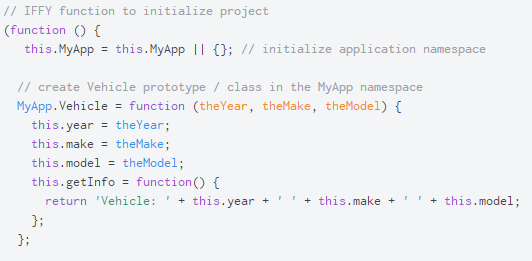


Then you define a sub class using the following syntax. The example below defines car as a sub class of Vehicle.



## 1.10 Putting it all together! Best practice for defining a prototype / class

Online example: <https://jsfiddle.net/u4jfanko/4/>



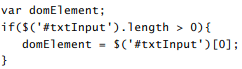
# 2.0 JQuery – The most fun you can have with JavaScript on the Web

## 2.1 Finding Elements

* $ is the selector before every jQuery call
* You can use jQuery to find an element on the HTML page with a CSS selector.

For example, $(‘#txtInput’) finds an HTML element with the id name “txtInput”.

* You can run the following code to ensure that a DOM element is found on the page.
* You can find the document itself by running the code $(document)



## 2.2 JQuery Object Functions

* cotrol.**val()** gets or sets the value property of a form control that has a value property.
* $(‘#btnSubmit’).on(‘click’, my Function) sets up an event listener for clicking a button.
* $(‘#btnSubmit’).off(‘click’, my Function) removes an event listener for clicking a button.
* $(‘#btnSubmit’).triggerHandler(‘click’) executes the event click immediately. Useful for testing.
* $(document).ready(function () {

initialize(); // run your initialize code

}); is used to run initialize code when the document loads

## 2.3 Optional Exercise: Create Calculator starts on Page 297