Object in JavaScript - Object Creation

[Anil Singh](https://www.blogger.com/profile/09359926778482233933)  [4:56 AM](http://www.code-sample.com/2015/06/object-in-javascript.html)

**How to create a JavaScript object?**

The JavaScript [**object**](http://www.code-sample.com/2015/06/object-in-javascript.html) is a collection of properties and the each property associated with the **name-value pairs**. The object can contain **any data types** (numbers, arrays, object etc.)

The example looks like,

Var myObject= {empId : “001”, empCode :”X0091”};

In the above example, here are two properties one is empId and other is empCode and its values are “001” and “X0091”.

The properties name can be string or number. If a property name is number i.e.

Var numObject= {1 : “001”, 2 :”X0091”};

Console.log(numObject.1);  //This line throw an error!

Console.log(numObject[“1”]);  // will access to this line not get any error!

As per my thought, the number property name should be avoided.

**Types of creating an object (**There are two types**)**

1.      Object literals

2.      Object constructor

**Object Literals**: This is the most common way to create an object with object literal and the example as given below.

The empty object initialized using object literal i.e.

var emptyObj= {};

This is an object with 4 items using object literal i.e.

var emptyObj ={

  empId:”Red”,

  empCode: “X0091”,

 empDetail : function(){

    alert(“Hi”);

};

};

**Object Constructor**: The second way to create object using object constructor and the constructor is a function used to initialize new object.

The example looks like,

Var obj = new Object();

Obj.empId=”001”;

Obj.empCode=”X0091”;

Obj.empAddressDetai = function(){

       Console.log(“Hi, I Anil”);

};

**Summary:**

There is no best way to create an object. It’s depending on your uses case. You can choose all one as per your case. All have some benefits.

There are some styles are available to create an objects,

1.      Object Constructor,

2.      Literal Constructor,

3.      Function Based,

4.      Prototype Based,

5.      Function and

6.      Prototype Based Singleton Based.