Object & It's Property

```
<script>
var dog = {}; //Object with properties using dot notation
      dog.fur = "black combination brown";
      dog.eyes = "blue";
      dog.age = 8;

var cat = {}; //Object with properties using square bracket
     cat["fur"] = "white combination grey";
     cat["eyes"] = "blue";
     cat["age"] = 6;

var bird =  //Object using literal notation
{
     fur : "blue green",
     eyes : "red",
     age : 3
};

var horse = { fur : "brown", eyes : "red" }; //Object with combination literal notation & →
      horse.age = 20; //← add properties outside object by dot notation

var frog = new Object(); //Object using constructor
      frog.fur = "green dot black";
      frog.eyes = "white";
      frog.age = 1;

var Animal = function(fur, eyes, age) //Object using custom constructor
{                                                      (capitalize the name of constructor to distinguish from regular function)
     this.fur = fur;
     this.eyes = eyes;
     this.age = age;
}

var cow = new Animal("white", "black", 10);
var duck = new Animal("black white", "black", 2);


document.write("This my dog = " + dog["fur"] + ", " + dog["eyes"] + ", " + dog["age"] + "<br>");
document.write("This my cat = " + cat.fur + ", " + cat.eyes + ", " + cat.age + "<br>");
document.write("This my bird = " + bird["fur"] + ", " + bird["eyes"] + ", " + bird["age"] + "<br>");
document.write("This my horse = " + horse.fur + ", " + horse.eyes + ", " + horse.age + "<br>" );
document.write("This my frog = " + frog.fur + ", " + frog.eyes + ", " + frog.age + "<br>");
document.write("This is my cow = " + cow.fur + " " + cow.eyes + " " + cow.age + "<br>");
document.write("This is my duck = " + duck["fur"] + " " + duck["eyes"] + " " + duck["age"]);
</script>
```

browser

```
This my dog = black combination brown, blue, 8
This my cat = white combination grey, blue, 6
This my bird = blue green, red, 3
This my horse = brown, red, 20
This my frog = green dot black, white, 1
This is my cow = white black 10
This is my duck = black white black 2
```

Methods
(function inside object)

```
<script>
//object using literal notation with properties outside using dot notation  & square bracket
var rectangle = {};
            //object properties
            rectangle.slong = "100";
            rectangle.swidth = "50";
            rectangle["fillColor"] = "red";
            rectangle["lineColor"]= "black";

            rectangle.getWide = function() //method
            {
                        this.wide = this.slong*this.swidth;
                        return "Wide is : " + this.wide;
            }

var wide = rectangle.getWide(); //call method
document.write(wide); //print
</script>
```

Object using literal notation with properties outside

```
<script>
//object using literal notation
var rectangle =
{
            //object properties
            slong : "100",
            swidth : "50",
            fillColor : "red",
            lineColor : "black",

            getWide : function() //method
            {
                        this.wide = this.slong*this.swidth;
                        return "Wide is : " + this.wide;
            }
};

var wide = rectangle.getWide(); //call method
document.write(wide); //print
</script>
```

Object using literal notation with properties inside

```
<script>
//object using constructor
var rectangle = new Object();
            //object properties
            rectangle.slong = 2;
            rectangle.swidth = 5;

            rectangle.setLong = function(newLong)  //method
            {           this.slong = newLong;       }

rectangle.setLong(100); // set new value
document.write(rectangle["slong"]); //print properties
</script>
```

Object using constructor

Methods
(function outside object)

```
<script>
	this.rectanglePerimeter = function() //method outside object
	{	return "Perimeter is : " + (2*this.slong + 2*this.swidth);	}

//object using custom constructor
var Rectangle = function (itsLong, itsWidth)
{
	//object properties
	this.slong = itsLong;
	this.swidth = itsWidth;

	this.getRectanglePerimeter = rectanglePerimeter; //call method outside this object

	this.getRectangleWide = function() //method inside object
	{
		this.wide = this.slong*this.swidth;
		return "Wide is : " + this.wide;
	}

	this.getArea = function() //method inside object
	{
		if(this.wide > 1000)
		{	return "It's big area"; }
		else
		{	return "It's small area"; }
	}
};

var rectangle1 = new Rectangle(100, 50); //new instance from object

var wide = rectangle1.getRectangleWide(); //call method
var perimeter = rectangle1.getRectanglePerimeter(); //call method
var area = rectangle1.getArea();
document.write(wide); //print
document.write("<br>");
document.write(perimeter); //print
document.write("<br>");
document.write(area);
</script>
```

Object using custom constructor

browser

```
Wide is : 5000
Perimeter is : 300
It's big area
```

Passing object into function

```
<script>
//object using custom constructor
var Rectangle = function (itsLong, itsWidth)
{
            //object properties
            this.slong = itsLong;
            this.swidth = itsWidth;
};
```

Object using custom constructor

```
            //function (this is not method)
            var getRectangleWide = function(rec1, rec2)
            {
                        var wideRec1 = rec1.slong*rec1.swidth;
                        var wideRec2 = rec2.slong*rec2.swidth;
                        return "Wide rectangle 1 : " + wideRec1 + " <br> Wide rectangle 2 : " +wideRec2;
            };
```

Function 1

```
            //function (this is not method)
            var compareWide = function(rec1, rec2)
            {
                        var wideRec1 = rec1.slong*rec1.swidth;
                        var wideRec2 = rec2.slong*rec2.swidth;

                        if(wideRec1 > wideRec2)
                                    return "Rectangle 1 wide bigger than rectangle 2 wide";
                        else
                                    return "Rectangle 2 wide bigger than rectangle 1 wide";
            };

var rec1 = new Rectangle(100, 50); // new instance from object
var rec2= new Rectangle(50, 70); // new instance from object

var area = getRectangleWide(rec1, rec2); // call function with 2 parameters
document.write(area); // print

document.write("<br>");

var compare = compareWide(rec1, rec2); // call function with 2 parameters
document.write(compare); // print
</script>
```

Function 2

browser

```
Wide rectangle 1 : 5000
Wide rectangle 2 : 3500
```

Inheritances of object literal notation (using object.create)

```
rectangle = slong, swidth
rec1 = slong, swidth, fillColor, lineColor
```

browser

```
Long : 100; Width :20
Fill Color : red; Line Color : blue
Name : Super Rectangle
Wide : 2000
```

```
<script>
//object using literal notation
var rectangle =
{
        slong : 100,
        swidth : 20,

        getWide : function() // method
        {       return this.slong*this.swidth;      }
};

var rec1 = Object.create(rectangle); // rec1 inherit the properties from rectangle
        rec1.fillColor = "red"; // rec1 have it's own property
        rec1.lineColor = "blue"; // rec1 have it's own property
        rec1.getName = function() // rec1 have it's own method
        {       return "Super Rectangle";       }

document.write("Long : " + rec1.slong  + "; " + "Width :" +  rec1.swidth); // print it's parent properties
document.write("<br>");
document.write("Fill Color : " + rec1.fillColor + "; " + "Line Color : " + rec1.lineColor ); // print
document.write("<br>");
document.write("Name : " + rec1.getName()); // print it's method
document.write("<br>");
document.write("Wide : " + rec1.getWide()); // print it's parent method
</script>
```

Inherit object change property of object inside object

browser

```
Old Account : SuperRec 123
New Account : grigo sert21
```

```
<script>
var rectangle = //object using literal notation
{
        slong : 100,
        swidth : 20,

        setConfig : function(newUser, newPassword) // method
        {
                this.config.setUser = newUser;
                this.config.setPassword = newPassword;
        },

        config :  // object inside object rectangle
        { setUser : "SuperRec", setPassword : "123" }
};

var rec1 = Object.create(rectangle); // rec1 inherit the properties from rectangle
        rec1.fillColor = "red"; // rec1 have it's own property

document.write("Old Account :  " + rec1.config.setUser + " " + rec1.config.setPassword);
document.write("<br>");
rec1.setConfig("grigo", "sert21");
document.write("New Account : " + rec1.config.setUser + " " + rec1.config.setPassword);
</script>
```

object inside object

inherit object

Make more than one instance of object with custom constructor

```
<script>
var Shape = function(itsName, itsLong, itsWidth, itsSide, itsFillcolor, itsLinecolor)
{
            this.name = itsName;
            this.slong = itsLong;
            this.swidth = itsWidth;                          properties
            this.side = itsSide;
            this.fillColor = itsFillcolor;
            this.lineColor = itsLinecolor;

            this.getName = function()
            {           return "Name : " + this.name;       }
                                                                                   custom
            this.getRectangleWide = function()                                     constructor
            {
                        this.wide = this.slong*this.swidth;
                        return "Wide is : " + this.wide;                methods
            }

            this.getSquareWide = function()
            {
                        this.wide = this.side*this.side;
                        return "Wide is : " + this.wide;
            }
};

var rectangle     = new Shape("rectangle 1", 100, 50, " ", ["red","green","blue"], "black");   new instance
var square        = new Shape("square 1", " ", " ", 75,  ["yellow", "orange", "pink"], "blue");  from  object

//print (methods and properties)
document.write(rectangle.getName()); //methods
document.write("<br>");
document.write("Long : " + rectangle['slong'] + ", " + "Width : " + rectangle['swidth']); //properties
document.write("<br>");
document.write(rectangle.getRectangleWide()); //methods
document.write("<br>");
document.write(" Fill Color : " + rectangle["fillColor"]); //properties

document.write("<br>");
document.write("<br>");

document.write(square.getName()); //methods
document.write("<br>");
document.write("Side : " + square['side']); //properties
document.write("<br>");
document.write(square.getSquareWide()); //methods
document.write("<br>");
document.write(" Fill Color : " + square["fillColor"]); //properties
</script>
```

browser

```
Name : rectangle 1
Long : 100, Width : 50
Wide is : 5000
Fill Color : red,green,blue

Name : square 1
Side : 75
Wide is : 5625
Fill Color : yellow,orange,pink
```

Add property to cutom constructor with prototype property

```
<script>
//object using custom constructor
var Rectangle = function (itsLong, itsWidth)
{
        this.slong = itsLong;      } properties
        this.swidth = itsWidth;    }
};

var rec1 = new Rectangle(100, 20); // new instance of object Rectangle

Rectangle.prototype.fillColor = null;  // add property name to object constructor

rec1.fillColor = "Red"; // fill value of property name to new instance rec1

document.write("Long : " + rec1.slong); // print
document.write("<br>");
document.write("Width : " + rec1.swidth); // print
document.write("<br>");
document.write("Fill Color : " + rec1.fillColor); // print
</script>
```

browser

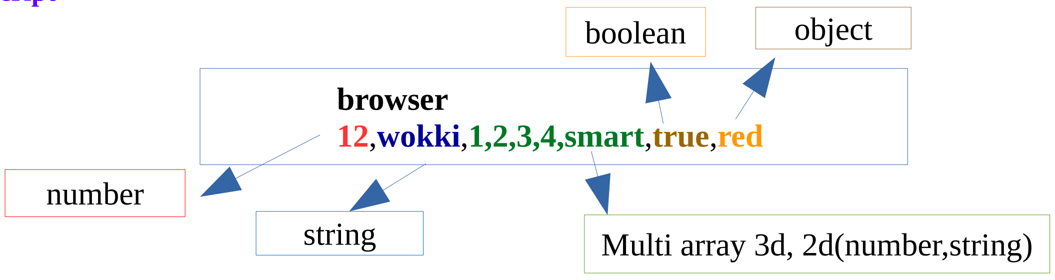
```
Long : 100
Width : 20
Fill Color : Red
```

Function will work as property of object using keyword with

```
<script>
function itsFillColor(getFillColor) // function (not method)
{
        with(this)
        {       fillColor = getFillColor;       }
};

var Rectangle = function (itsLong, itsWidth) //object using custom constructor
{
        this.slong = itsLong;
        this.swidth = itsWidth;
        this.fillColor = " ";
        this.addFillColor = itsFillColor; //function as property of this object
};

var rec1 = new Rectangle(100, 20); // new instance of object
rec1.addFillColor("Red"); // fill value of property name to new instance

document.write("Long : " + rec1.slong + "<br>");
document.write("Width : " + rec1.swidth + "<br>");
document.write("Fill Color : " + rec1.fillColor);
</script>
```

Insert object into arrays

```
<script>
var man = true; //boolean
var paper = { color : 'red' }; //object
var myArrayTest = [12, "wokki", [1,2,3], [4, "smart"], man, paper['color'] ]; //arrays
//print arrays
document.write(myArrayTest);
</script>
```



```
browser
12,wokki,1,2,3,4,smart,true,red
```

Insert object into arrays containing arrays (nested array)

```
<script>
var wall = { color : "red", height :12 }; //object
var myWall = [wall['color']]; //array contains objects wall with property color
var newWall = [[1,2,3], [myWall]]; //array contains array
document.write(newWall); //print array
</script>
```

```
browser
1,2,3,red
```

Insert arrays into object

```
<script>
var newObject = { hmm : "yumi", tyu : ["yumi",3] }; //object contains array
//print
document.write(newObject['hmm'] + "<br>");
document.write(newObject['tyu']);
</script>
```

```
browser
yumi
yumi,3
```

Insert array into object containing object (nested object)

```
<script>
var myFriends = {}; //object
myFriends.naruto = {
  firstName: "Naruto",
  lastName: "Uzumaki",
  number: "(206) 555-4444",
  address: ['Japan','TV','00000'] //arrays inside object
};
var list = function(listFriends)
{
  for(var prop in listFriends)
  {    document.write(prop +  " : " + listFriends[prop] + "<br>");  }
}
list(myFriends.naruto) //call function
</script>
```

Object inside object

Function Not method

```
browser
firstName : Naruto
lastName : Uzumaki
number : (206) 555-4444
address : Japan,TV,00000
```

Built in operator : in, typeof, delete & built in method : hasOwnProperty

```
<script>
//object using literal notation with properties outside using dot notation  & square bracket
var rectangle = {};
        //object properties
        rectangle.slong = "100";
        rectangle.swidth = "50";
        rectangle["fillColor"] = "red";
        rectangle["lineColor"]= "black";

document.write("<br>");
document.write("slong" in rectangle); // check if islong is property of  object rectangle
document.write("<br>");

document.write(typeof rectangle); // checking tipe of variable
document.write("<br>");

document.write(rectangle.hasOwnProperty('fillColor')); // check if object naruto have property fillColor
document.write("<br><br>");

for (var prop in rectangle) // print all property name & value of object rectangle
{
        document.write(prop + " : " + rectangle[prop]);
        document.write("<br>");
}

delete rectangle.slong; // delete one property of object rectangle
for (var prop in rectangle) // print all property name & value of object rectangle
{
        document.write(prop + " : " + rectangle[prop]);
        document.write("<br>");
}
document.write("<br><br>");
</script>
```

browser

```
true
object
true

slong : 100
swidth : 50
fillColor : red
lineColor : black

swidth : 50
fillColor : red
lineColor : black
```

Built in method : toString, built in property : length

```
<script>
var Rectangle = function (itsName, itsLong, itsWidth) // object using custom constructor
{
        this.name = itsName;
        this.slong = itsLong;
        this.swidth = itsWidth;

        this.getNameLength = function() // method
        {       return this.name.toString().length;
        }
};

var rec1 = new Rectangle("Rectangle Super", 100, 2000); // new instance of object

document.write("Length of " + rec1.name + " = " + rec1.getNameLength() + " Character");
</script>
```

browser

```
Length of Rectangle Super = 15 Character
```

toString = conver to string
length = amount of character

Built in operator : instanceof

```
<script>
var Rectangle = function (itsName, itsLong, itsWidth) //object using custom constructor
{
        this.name = itsName;
        this.slong = itsLong;
        this.swidth = itsWidth;
};

var rec1 = new Rectangle("Rectangle Super", 100, 2000); // new instance of object

var test = rec1 instanceof Rectangle // check if rec1 is instance of constructor Rectangle
// var test = rec1.constructor == Rectangle;
document.write(test);
</script>
```

browser

```
true
```

Don't use keyword new to create instance of object

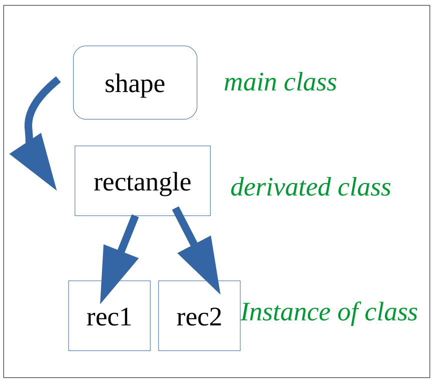
```
<script>
var Rectangle = function (itsName, itsLong, itsWidth) //object using custom constructor
{
        this.name = itsName;
        this.slong = itsLong;
        this.swidth = itsWidth;

        if(!(this instanceof Rectangle)) //  if we don't use new when create instance below
        {
                return new Rectangle(itsName, itsLong, itsWidth);
        }
};

var rec1 = new  Rectangle("Rectangle Super", 100, 2000); // new instance of object
var rec2 = Rectangle("Rectangle Hard", 40, 200); // without keyword new

var test1 = rec1 instanceof Rectangle;
var test2 = rec2 instanceof Rectangle;

document.write(test1  + "  & " + test2);
</script>
```

browser

```
true & true
```

Inheritances of object custom constructor (Class)
(main class, derivated class, inheritances properties & methods using prototype)

```
<script>
var Shape = function(itsDefaultName) // object with custom constructor / we can said class shape
{
        this.defaultName = itsDefaultName;          main class
};

        Shape.prototype.fillColor = "red";  // property class shape
        Shape.prototype.lineColor = "blue";   // property class shape
        Shape.prototype.getRectangleWide = function(recLong, recWidth) // method class shape
        {
                var wide = recLong * recWidth;
                return "Wide is : " + wide;
        };

// object with custom constructor / we said class rectangle now
var Rectangle  = function(itsLong, itsWidth)
{
        this.slong = itsLong;
        this.swidth = itsWidth;          derivated class
};

// set rectangle as derivated of class shape
Rectangle.prototype = new Shape();          inheritance

// new instance of derivated class rectangle
var rec1 = new Rectangle();
var rec2 = new Rectangle();          new instance

var areaRec1 = rec1.getRectangleWide(100, 30); // call method of main class shape
var areaRec2 = rec2.getRectangleWide(10, 20);  // call method of  main class shape
document.write("Fill Color : " + rec1.fillColor + " ; " + "Line Color : " + rec1.lineColor + "<br>");
document.write("Rectangle 1, " + areaRec1); // print
document.write("<br>");
document.write("Rectangle 2, " + areaRec2); // print
</script>
```



browser

```
Fill Color : red ; Line Color : blue
Rectangle 1, Wide is : 3000
Rectangle 2, Wide is : 200
```

```
        Shape.prototype = // method class shape
        {
                getRectangleWide : function(recLong, recWidth) // method 1 class shape
                {
                        var wide = recLong * recWidth;
                        return "Wide is : " + wide;
                },
                getRectanglePerimeter : function(recLong, recWidth) // method 2 class shape
                {
                        var perimeter = 2*recLong + 2*recWidth;
                        return "Perimeter is : " + perimeter;     other tehnique
                }
        };
        Shape.prototype.fillColor = "red";  // property class shape
        Shape.prototype.lineColor = "blue";   // property class shape
```

Private Variable

```
<script>
var rectangle = function (itsLong, itsWidth) // object with custom constructor / class
{
        //object properties
        this.slong = itsLong;
        this.swidth = itsWidth;
        var fillColor = "Red"; // private variable

        this.getFillColor  = function(pass)
        {
                if(pass==123)
                  return "RIGHT" + "<br> " + "Fill Color : " + fillColor;
                else
                  return "WRONG";
        };
};

var rec1 = new rectangle(100, 200); // new instance from object

document.write("Long : " + rec1.slong + “<br>”); // print
document.write("Width : " + rec1.swidth  + “<br><br>”);  // print
var fillColor = rec1.getFillColor(123); // call function
document.write(fillColor);  // print
</script>
```

Function to make private variable accessible

browser

```
Long : 100
Width : 200

RIGHT
Fill Color : Red
```