

**Object
Oriented
JavaScript**



Object Oriented JavaScript

OOP

Object-oriented programming (OOP)

Creating reusable software objects

Object

Programming code and data that can be treated as an individual unit or component

Data

Information contained within variables

Object-oriented principles

Encapsulation

Inheritance

Polymorphism

INSTANTIATING AN OBJECT

Creating an object from existing class

- Using new Operator

```
Var today=new Date();  
Var objname = new Object();
```

To create an array you have two ways

```
Var myarr=new Array[3];  
    Or  
Var myarr=[1,2,3];  
//both create array object
```

FROM ARRAYS TO OBJECTS

In Arrays :

```
Var myarr=[1,2,3];           //using [] for array
```

In Objects :

```
Var myobj={id:10};          //using {} for object
```

This is an object that has a property called id with value = 10.

```
Var a={};                   //a is an empty object
```

ENCAPSULATION

Each object contain Properties & Methods to access these Properties.

```
Var Car = { model: 'Toyota',           //object has properties  
            color: 'Red',  
            move: function() {....}  
            beep: function(r){ alert(r) } }; //and methods
```

ACCESSING OBJECT MEMBERS

There are two ways to access object members

- Using **[]** Square brackets.

```
Car['model'];           //read value from property
Car['color'] = "blue";  //assign value to property
Car['move']();          //call a function
```

- Using **dot** operator.

```
Car.color="black";      //like C++
Car.move();
```

ALTERING OBJECT MEMBERS

Because JavaScript code is parsed not compiled you can add or remove properties of the object.

- Add property to the object.

```
Car.motor="1300 cc";           //adding motor to Car
```

- Remove property from the object.

```
Delete Car.model;             //deleting model attribute
```

```
Car.model;                     //"undefined"
```

typeof OPERATOR

Now we have the object Car but from which class we instantiated this object.

```
Typeof Car;                //Object
```

Each object has a property called **Constructor**

```
Car.constructor;           // Object()
```

The object created using `{}` it's constructor is **Object()**

CLASSES IN JAVASCRIPT

- JavaScript doesn't support the notion of *classes as typical OOP languages* do.
- In JavaScript, you create *functions that can behave—in many cases—just* like classes called **Constructor Function**.
- For example, you can call a function, or you can create an instance of that function supplying those parameters.

CONSTRUCTOR FUNCTION

```
Function Human()  
{  
    this.name;           // this refers to the caller object  
    this.age;  
    this.sayName=function() // sayName function in human  
        {alert(this.name);}  
}
```

To take an object from Human

```
Var obj = new Human();  
obj.name="Mohamed";  
obj.age=22;  
obj.sayName();           // alert Mohamed
```

GLOBAL OBJECT

If we called Human function without new operator the **this** refers to the global object **Window**

When you say this.age you add age property to the window object.

```
Human();
```

```
Window.age=30;           //window has no age property
```

C++ & JAVASCRIPT CLASSES

```
Class Table
{
    public:
    int rows,cols;
    Table(int rows,int cols)
    {
        this.rows = rows;
        this.cols = cols;
    }
    int getCellCount()
    {
        return rows * cols;
    }
};
```

```
function Table(rows,cols)
{
    //constructor
    this.rows=rows;
    this.cols=cols;
    //method
    this.getCellCount=function()
    {
        return this.rows * this.cols;
    }
}
```

PRIVATE MEMBERS

```
function Table(rows,cols)
{
    //constructor
    var _rows = rows;
    var _cols = cols;
    //method
    this.getCellCount=function()
    {
        return _rows * _cols;
    }
}
```

Now if you tries to access

```
Var myTable=newTable(3,2);
```

```
myTable.getCellCount();
//it works and returns 6
```

```
mytable._rows;
//undefined
```

Thank You