# Objects and Its Internal Representation in JavaScript

## Objects in JavaScript are a collection of related data, of primitive types or reference types, in the form of “Key: Value” pairs.

Objects are the most important data-type in the modern JavaScript. These are quite different from the primitive data types and also very different from objects in Java/ C++.

Objects are a data-type which can contain multiple data types in it. So, for eg. A JavaScript object can describe an employee in an organization as follows:

```
Let employee = {
name: ‘employeeName’, //Samir Suri
photo:’employeePhoto’, //Samir.png
salary:’empSalary’, //$120,000
function:’functionInWhichTheEmployeeWorks’, //Sales or Product Development
manager:’reportingManager’, //Abhinav Mishra
changemanager: function(‘newManagerName’){
this.manager = ‘newManagerName’ }, // function which updates the new manager name
changeSalary: function(‘newSalary’){
this.salary = ‘newSalary’} // function which updates the salary
```

Above is a literal notation of an object “employee”.

## Object Properties and Methods

The properties of the object define the characteristics of the object. Object employee has properties i.e. name which is a string data type, photo which has the path of the photo of employee, salary which is number data type, function which is a string data type, manager which is also a string data type. The properties are accessed by dot notation i.e.

```
employee.name = ‘samir suri’;
employee.photo = ‘../img/samir.png’;
employee.salary = 120,000;
```

If a property is not assigned a value then it will be “undefined” (and not “null”). The properties can also be accessed using bracket notation.

```
employee[name] = ‘samir suri’;

employee[photo] = ‘../img/samir.png’;

employee[salary] = 120,000;
```

When property names are non valid JavaScript Identifiers use of bracket notation is preferred. In case of dynamic property name determination it is recommended to use bracket notation.

Properties can also be accessed using expression

```
objectName[expression] // x = ‘manager’; employee[x]
```

An object method is an object property containing a function definition. Object employee has methods like changemanager() which is a function which updates the manager name; and changeSalary() which updates the salary of the employee.

```
employee.changemanager(‘Rajat Mishra’);
```

will update the object employee’s property manager with new value – ‘Rajat Mishra’.

## Create Objects in JavaScript

### Using Object Literal

```
let bike = {name: 'SuperSport', maker:'Ducati', engine:'937cc'};
```

Enclose the properties and its values in { }.

### Using Object Constructor

Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavour as shown below

```
function Vehicle(name, maker) {
this.name = name;
this.maker = maker; }
let car1 = new Vehicle(’Fiesta’, 'Ford’);
let car2 = new Vehicle(’Santa Fe’, 'Hyundai’)
console.log(car1.name); //Output: Fiesta
console.log(car2.name); //Output: Santa Fe
```

## Using new Keyword

```
var person = new Object();
person.firstName = “John”;
person.lastName = “Doe”;
person.age = 50;
person.eyeColor = “blue”;
```

## Using the Object.create() method

Objects can also be created using the Object.create() method. This method can be very useful, because it allows you to choose the prototype object for the object you want to create, without having to define a constructor function.

```
// Animal properties and method encapsulation
var Animal = {
  type: 'Invertebrates', // Default value of properties
  displayType: function() {  // Method which will display type of Animal
    console.log(this.type);
  }
};
// Create new animal type called animal1
var animal1 = Object.create(Animal);
animal1.displayType(); // Output:Invertebrates
// Create new animal type called Fishes
var fish = Object.create(Animal);
fish.type = 'Fishes';
fish.displayType(); // Output:Fishes
```