**OBJECTS AND ITS INTERNAL REPRESENTATION IN JAVASCRIPT**

**OBJECT:**

Objects are important data types in javascript. Objects are different than primitive datatypes (i.e. number, string, boolean, etc.).

Primitive data types contain one value but Objects can hold many values in form of Key: value pair.

These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

var myCar = new Object();

myCar.maker = 'Suzuki';

myCar.model = 'Altros';

myCar.year = 1978;

myCar.wheels = 2;

**OBJECT PROPERTIES:**

Properties are the values associated with a JavaScript object.A JavaScript object is a collection of unordered properties.Properties can usually be changed, added, and deleted, but some are read only.

The syntax for accessing the property of an object is:

objectName.property      // car.color

or

objectName["property"]   // car[“color]

or

objectName[expression]   // x = "color"; car[color]

**CREATE JAVASCRIPT OBJECT WITH OBJECT LITERAL**

One of easiest way to create a javascript object is object literal, simply define the property and values inside curly braces as shown below

let car = {name: 'Fiesta’, maker:'Ford', color:’Red’};

**CREATE JAVASCRIPT OBJECT WITH CONSTRUCTOR**

Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor 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 THE JAVASCRIPT KEYWORD NEW**

The following example also creates a new JavaScript object with four properties:

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