**Objects And Its Internal Representation In JavaScript**

* Objects, in JavaScript, is the most important data-type
* It forms the building blocks for modern JavaScript.
* These objects are quite different from JavaScript’s primitive data-types

**Primitive data-types**

* Number
* String
* Boolean
* Null
* Undefined
* symbol
* While these primitive data-types all store a single value each (depending on their types).
* Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.
* An object is a reference data type.

**Reference data type.**

* Variables that are assigned a reference value are given a reference or a pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don’t actually store the value.
* Objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs.
* These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

**For Eg.** If your object is a student, it will have properties like name, age, address, id, etc and methods like updateAddress, updateNam, etc.

**Objects and its properties**

* A JavaScript object has properties associated with it.
* A property of an object can be explained as a variable that is attached to the object.
* Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects.
* The properties of an object define the characteristics of the object.
* You can access the properties of an object with a simple dot-notation:

*objectName.propertyName*

* Like all JavaScript variables, both the object name (which could be a normal variable) and property name are case sensitive.
* You can define a property by assigning it a value.

**For Eg.** let’s create an object named myCar and give it properties named make, model, and year as follows:

*var myCar = new Object();  
myCar.make = 'Ford';  
myCar.model = 'Mustang';  
myCar.year = 1969;*

* Unassigned properties of an object are undefined (and not null).

*myCar.color;* // undefined

* Properties of JavaScript objects can also be accessed or set using a bracket notation.
* Objects are sometimes called associative arrays, since each property is associated with a string value that can be used to access it.

**For Eg.** you could access the properties of the myCar object as follows:

*myCar['make'] = 'Ford';  
myCar['model'] = 'Mustang';  
myCar['year'] = 1969;*

* An object property name can be any valid JavaScript string, or anything that can be converted to a string, including the empty string.
* However, any property name that is not a valid JavaScript identifier (for example, a property name that has a space or a hyphen, or that starts with a number) can only be accessed using the square bracket notation.
* This notation is also very useful when property names are to be dynamically determined (when the property name is not determined until runtime).

**For Eg.**

// four variables are created and assigned in a single go,   
// separated by commas  
*var myObj = new Object(),  
 str = 'myString',  
 rand = Math.random(),  
 obj = new Object();  
myObj.type = 'Dot syntax';  
myObj['date created'] = 'String with space';  
myObj[str] = 'String value';  
myObj[rand] = 'Random Number';  
myObj[obj] = 'Object';  
myObj[''] = 'Even an empty string';console.log(myObj);*

* You can also access properties by using a string value that is stored in a variable:

*var propertyName = 'make';  
myCar[propertyName] = 'Ford';propertyName = 'model';  
myCar[propertyName] = 'Mustang';*

* You can use the bracket notation with for...in to iterate over all the enumerable properties of an object. Creating Objects In JavaScript :

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

**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 type
* This is also known as constructor function

*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**

**For Eg.**

*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