**Objects And Its Internal Representation In JavaScript:**

## **Objects in JavaScript:**

* Objects are one of the most fundamental and versatile data types in JavaScript. They are used to store and manipulate various kinds of information, such as properties, methods, events, prototypes, and more. In this blog post, we will explore how objects are represented internally in JavaScript, and how we can access and modify their properties and behavior.

# ****Objects and 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 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 example, 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;**

# ****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'};** |

**Internal Representation of Objects:**

* In JavaScript, an object is a collection of key-value pairs, where each key is either a string or a symbol, and each value can be any data type. Internally, an object is represented by a data structure called a \*\*hash table\*\*, which is a way of storing and retrieving data efficiently using a \*\*hash function\*\*. A hash function is a mathematical function that maps any input to a fixed-size output, called a \*\*hash\*\* or a \*\*hash code\*\*. For example, the hash function `h(x) = x mod 10` maps any integer `x` to a hash code between 0 and 9.
* A hash table consists of an array of \*\*buckets\*\*, where each bucket can store one or more key-value pairs. The hash function determines which bucket to store or retrieve a key-value pair from, by computing the hash code of the key and using it as an index in the array. For example, if we have a hash table with 10 buckets, and we want to store the key-value pair `{"name": "Alice"}`, we can use the hash function `h(x) = x mod 10`, where `x` is the ASCII value of the first character of the key. Since the ASCII value of `"n"` is 110, we get `h("name") = 110 mod 10 = 0`, so we store the key-value pair in the bucket at index 0.

# 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:

**Example**

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()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/create" \t "_blank) 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**