**Objects and its internal representation**

Define ;

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.

**Indroduction ;**

Objects are important data types in javascript. Objects are different than primitive data types such as Number, String, Boolean, null, undefined and symbol. 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.

Every object has some property associated with some value. These values can be accessed using these properties associated with them.

Example ;

var myBike = new Object();

myBike.maker = 'Royal Enfield';

myBike.model = 'Classic 350';

myBike.year = 2022;

myBike.wheels = 2;

After creating myBike object, the value inside the object can be accessed using keys.

**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 myBike and give it properties named make, model, and year as follows:

var myBike = new Object();  
myBike.maker = 'Royal Enfiled';  
myBike.model = 'Classic';  
myBike.year = 2022;

Unassigned properties of an object are undefined (and not [null](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/null)).

myBike.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. So, for example, you could access the properties of the myBike object as follows:

myBike['maker'] = 'Royal Enfiled';  
myBike['model'] = 'Classic';  
myBike['year'] = 2022;

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). Examples are as follows:

// 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';  
myBike[propertyName] = 'Royal Enfiled';

propertyName = 'model';  
myBike[propertyName] = 'Classic 350';

# ****Creating Objects :****

# Create 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: 'R15 V4.0', maker:'Yamaha', engine:'155cc'};

# Create 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 bike1 = new Vehicle(’Splendor Plus’, 'Hero’);  
let bike2 = new Vehicle(’Shine’, 'Honda’)  
console.log(bike1.name); //Output: Splendor Plus  
console.log(bike2.name); //Output: Shine

# Using the JavaScript Keyword new ;

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

Example

var person = new Object();  
person.firstName = “Sethu”;  
person.lastName = “Ram”;  
person.age = 20;  
person.eyeColor = “Brown”;

# 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

Thank You