**Objects and it is internal representation in Javascript:**

Objects are non-primitive data type in javascript. 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. 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.

Each and every object possesses a property that corresponds with a specific value. The values can be retrieved by accessing the respective properties.

Once the object is created, its contents can be retrieved using keys.

For Example:

var obj = {

name: abc,

address:xyz }

console.log(obj.name) 🡪 Output: abc .

It is also possible to access these values using the bracket notation.

Console.log(obj[address]) 🡪 Output: xyz

We can add, retrieve, delete and edit a property to an object

The syntax for adding a property to an object is :

🡪ObjectName.ObjectProperty = propertyValue;

The syntax for removing an object's property is as follows:

🡪delete ObjectName.ObjectProperty;

One can use the following syntax to retrieve(access) a property from an object:

🡪objectName.property (dot notation)

//or

🡪objectName["property”] (bracket notation)

//or

🡪objectName[expression]

**Inherited Properties**

Inherited properties of an object are those properties that have been inherited from the object’s prototype, as opposed to being defined for the object itself, which is known as the object’s Own property. To verify if a property is an object’s Own property, we can use the hasOwnProperty method. Property Attributes Data properties in JavaScript have four attributes.

value: The property’s value.

writable: When true, the property’s value can be changed

enumerable: When true, the property can be iterated over by “for-in” enumeration. Otherwise, the property is said to be non-enumerable.

configurable: If false, attempts to delete the property, change the property to be an access-or property, or change its attributes (other than [[Value]], or changing [[Writable]] to false) will fail.