**Objects and Internal Representations**

**Objects :**

Objects, in JavaScript, is its most important data-type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive data-types(Number, String, Boolean, null, undefined and symbol) in the sense that 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. 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.

**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**

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).

# **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 :**

Internally, JavaScript objects are implemented using a variety of data structures. The most common internal representation of objects in JavaScript is through a combination of properties and a hidden property called the prototype.

* **Properties and Values:** Each property of an object is stored as a key-value pair. The key is usually a string, and the value can be of any data type. These properties are stored in memory, allowing you to access and manipulate them.
* **Prototype Chain:** JavaScript uses a mechanism called the prototype chain to provide inheritance and shared behavior among objects. Each object has an associated prototype object, which acts as a fallback for properties and methods that are not directly present in the object itself. This allows for a form of inheritance where objects can inherit properties and methods from their prototypes.
* **Hidden Properties:** In addition to the properties you define, JavaScript adds some hidden properties to objects for its internal management. These hidden properties store information about the object's prototype, class, and other details.
* **Memory Allocation:** Objects are allocated memory dynamically as needed. The memory space required for an object is determined by the number of properties it has and the size of those properties.