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

Objects, in JavaScript, is its most important datatype and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive datatypes (Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive datatypes all store a single value each (depending on their types).

Objects are more complex, and each object may contain any combination of these primitive datatypes as well as reference datatypes.  
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 store the value.

Loosely speaking, 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.

# **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 “name1”and give it properties named name, Age, and class as follows:

var name1 = new Object ();  
name1.name = ‘abhishek';  
name.1age = '16';  
name1.class = 10th;

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

Name1.color; // undefined

Properties of JavaScript objects can also be accessed or set using a bracket notation (for more details see [property accessors](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_Accessors)). 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 name object as follows:

name1['name'] = 'abhishek';  
name1['age'] = '16';  
name1['class'] = 10th;