# Write a blog about objects and its internal representation in Java script

# INTRODUCTION

* JavaScript is designed on a simple object-based paradigm.
* An object is a collection of properties, and a property is an association between a name (or key) and a value
* A property’s value can be a function, in which case the property is known as a method.

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.
* We can 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:

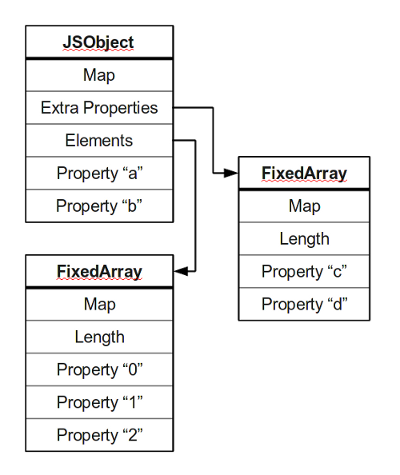
|  |
| --- |
| *var myCar = new Object(); myCar.make = ‘Ford’; myCar.model = ‘Mustang’; myCar.year = 1969;* |

The above example could also be written using an **object initializer**, which is a comma-delimited list of zero or more pairs of property names and associated values of an object, enclosed in curly braces ({}):

|  |
| --- |
| *var myCar = { make: ‘Ford’, model: ‘Mustang’, year: 1969 };* |

**JavaScript’s internal representation of Objects:**

A simple diagram is probably the best way to give a quick overview of the object representation in Javascript.



Most objects contain all their properties in a single block of memory *(‘a’ and ‘b’)*. All blocks of memory have a pointer to a map, which describes their structure.

Named properties that don’t fit in an object are usually stored in an overflow array *(‘c’ and ‘d’)*.

Numbered properties are stored separately, usually in a contiguous array.

The JavaScript standard allows developers to define objects in a very flexible way, and it is hard to come up with an efficient representation that works for everything. An object is essentially a collection of *properties*: basically key-value pairs. We can access properties using two different kinds of expressions:

* obj.prop
* obj[“prop”]

According to the spec, property names are always strings. If we use a name that is not a string, it is implicitly converted to a string. This may be a little surprising: if we use a number as a property name, it gets converted to a string as well. So a JavaScript object is basically a map from strings to values.