objects and its internal representation in JavaScript

In JavaScript, objects are a crucial data type that serves as the foundation for modern JavaScript. These objects differ from JavaScript's primitive data types (Number, String, Boolean, null, undefined, and symbol) in that they can contain any combination of primitive and reference data types. An object is a reference data type, which means that when a variable is assigned a reference value, it is given a reference or pointer to that value's location in memory.

Objects in JavaScript can be thought of as an unordered collection of related data, consisting of primitive or reference types in the form of **"key: value"** pairs. These keys can be variables or functions and are referred to as properties and methods, respectively. For example, for a person, it will have properties such as name, age, address, id, etc., and methods such as updateAddress and updateName.

A JavaScript object has properties associated with it. A property of an object can be thought of as a variable attached to the object. Object properties are similar to ordinary JavaScript variables, except that they are attached to objects. The characteristics of an object are defined by its properties. You can access an object's properties using dot notation **(objectName.propertyName)** or bracket notation **(objectName[propertyName])**.

You can define a property by assigning it a value. For example, for an object named **“myCar”** in JS and give it properties named `make`, `model`, and `year`:

```
const myCar = {};

myCar.make = 'Ford';

myCar.model = 'Mustang';

myCar.year = 1969;

console.log(myCar); // { make: 'Ford', model: 'Mustang', year: 1969 }
```

Unassigned properties of an object are undefined. Properties of JavaScript objects can also be accessed or set using bracket notation. Objects are sometimes referred to as associative arrays because each property is associated with a string value that can be used to access it.