Write a blog about objects and its internal representation in JavaScript

Introduction:

JavaScript is a versatile and widely used programming language that relies heavily on objects. Objects play a fundamental role in JavaScript, serving as the building blocks for constructing complex data structures and enabling the creation of dynamic and flexible code. In this blog post, we will dive into the world of objects in JavaScript and explore their internal representation.

Objects in JavaScript:

In JavaScript, an object is a composite data type that encapsulates related data and functionality. Unlike other primitive data types like numbers and strings, objects are more complex and can hold multiple values and methods. Objects are essentially collections of key-value pairs, where the keys are called properties, and the corresponding values can be of any type, including other objects.

Objects, in JavaScript, is it’s 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, Array, Boolean, Symbol, null and undefined) 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.

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. For Eg. If your object is a student, it will have properties like name, age, address, id, etc

A constructor is a function that creates and initializes an **object**. **JavaScript** provides a special constructor function called **Object**() to build the **object**. The return value of the **Object**() constructor is assigned to a variable.

Internal Representation of Objects:

Internally, JavaScript implements objects using a mechanism called "hash tables" or "hash maps." A hash table is a data structure that allows efficient storage and retrieval of key-value pairs. In the case of JavaScript objects, the keys are the property names, and the values are the associated data or functions.

When an object is created in JavaScript, memory is allocated to store the object's properties and methods. Each property is internally represented as a key-value pair, where the key is stored as a string and the value can be of any type. The JavaScript engine uses a hash function to convert the property name into an index, which is then used to store the key-value pair in the underlying hash table.

In JavaScript, objects are key components that enable developers to create complex data structures and dynamic applications. Understanding how objects are internally represented is crucial for effectively working with them. In this blog post, we will explore the internal representation of objects in JavaScript.

Object Creation:

In JavaScript, objects can be created in several ways. The most common method is using object literal notation.

Property Storage:

Internally, object properties are stored as key-value pairs. The property names are stored as strings, and the corresponding values can be of any data type, including other objects. The object's properties are stored in a structure called the "hidden class" or "shape" of the object. This hidden class helps optimize property access and improves performance.

Hidden Class:

When an object is created, the JavaScript engine assigns it a hidden class, which represents the object's structure and the properties it contains. The hidden class defines the order and layout of the properties in memory. As properties are added or removed, the hidden class may change dynamically.

Property Access:

To access an object's property, the JavaScript engine looks up the property using the hidden class. It computes a hash value for the property name and checks the corresponding memory location for the property's value. If the property exists, its value is returned; otherwise, undefined is returned.

Property Addition and Deletion:

Adding properties to an object is straightforward. We can simply assign a new key-value pair. When a new property is added, the hidden class may change, potentially impacting performance. It's worth noting that adding properties in a consistent order can help optimize the hidden class transitions.

To delete a property from an object, the delete operator is used

Property Enumeration:

Properties of an object can be enumerated using loops, such as for...in or Object.keys(): The order of property enumeration is not guaranteed in JavaScript, so it's important not to rely on a specific order.

Object Methods:

Objects in JavaScript can have methods, which are functions stored as properties. When a method is accessed, it is invoked as a regular function with the object as its context (represented by the 'this' keyword).

Understanding how objects are internally represented in JavaScript is crucial for writing efficient and optimized code. Objects store properties as key-value pairs, utilize hidden classes to optimize property access, and allow for the addition and deletion of properties. By grasping these concepts, developers can leverage the power of objects and create robust applications in JavaScript.