Giridharan K
23-11-2023
Day-1

2. Objects and its internal representation in Java script.

JavaScript is a versatile and dynamic programming language that employs various data types to represent and manipulate information. One of the fundamental data types in JavaScript is the object, a complex and flexible construct that plays a pivotal role in the language. In this blog post, we'll delve into the concept of objects in JavaScript, exploring their internal representation, and understanding how they contribute to the language's power and expressiveness.

The Basics: What is an Object?

In JavaScript, an object is a collection of key-value pairs, where each key is a string (or a Symbol) and each value can be any data type, including other objects. This key-value pairing allows developers to create complex and hierarchical structures to represent real-world entities or abstract concepts.

Let's start with a simple example:

```
// Creating an object

const person = {

  name: "Giridharan",

  age: 22,

  profession: "Software Developer"

};


// Accessing object properties

console.log(person.name); // Output: Giridharan
```

In this example, `person` is an object with three properties: `name`, `age`, and `profession`. The keys are strings, and the values can be of any valid JavaScript data type.

Internal Representation of Objects:

Internally, how does JavaScript represent these objects? Understanding the internal workings helps developers make informed decisions about object usage and optimize their code for better performance.

1. Object Properties:

JavaScript objects use two main mechanisms for storing and accessing properties: **named data properties** and **internal methods**.

Named Data Properties:

These properties hold the actual data values associated with the object. For example, in `person.name`, `name` is a named data property.

Internal Methods:

These methods are built-in functions that enable various operations on objects. For instance, the `[[Get]]` method is invoked when retrieving a property value.

2. Prototypes:

JavaScript objects have a prototype chain that allows them to inherit properties and methods from other objects. This chain is crucial for object-oriented programming in JavaScript.

```
// Creating an object with a prototype

const student = Object.create(person);



console.log(student.name); // Output: Giridharan
```

In this example, `student` inherits properties from `person` through the prototype chain.

3. Hidden Classes:

JavaScript engines use a concept called hidden classes (or shapes) to optimize object property access and assignment. These hidden classes define the layout of an object and help improve performance.

Objects and Immutability:

JavaScript objects are mutable, meaning their properties can be changed. However, in some scenarios, immutability is desirable for better predictability and ease of reasoning about the code. Libraries like Immutable.js provide immutable data structures in JavaScript, enabling the creation of objects that cannot be modified after their initial creation.

Conclusion:

Objects in JavaScript serve as a powerful tool for organizing and representing data. Their internal representation involves a combination of named data properties, internal methods, prototypes, and hidden classes. Understanding these internal mechanisms empowers developers to write efficient and performant code.

As JavaScript evolves, new features like classes and enhanced syntax make working with objects even more intuitive. The dynamic nature of JavaScript, coupled with its robust object system, continues to contribute to the language's popularity and applicability across a wide range of domains. Whether you're building a simple web page or a complex web application, a solid understanding of objects and their internal representation is crucial for leveraging the full potential of JavaScript.