**2.Write a blog about objects and its internal representation in Javascript**

JavaScript Object Internal Representation:

Image of JavaScript object internal representation as a key-value map

* **Key-Value Pairs**: At their core, JavaScript objects are essentially key-value maps. Think of them as treasure chests where keys (property names) unlock specific values (data).
* **Handles to Memory**: Each object holds a reference to a hidden memory location where its properties and methods reside. This means that multiple variables can point to the same object, creating shared access.
* **Properties**: These are the named slots within an object that store values of various data types (strings, numbers, arrays, even other objects!) Think of them as the labeled drawers within the storage unit.
* **Methods**: Functions attached to objects bring them to life, allowing them to perform actions. They're like the handy tools within the box, ready to be used for specific tasks.

**Creating Objects:**

const MyDetails = {

Name: "Murali",

Age: 24

Role: "Full-Stack Developer" ,

Output(){

console.log("Future Full-Stack Developer");

}

};

**Internal Representation of Objects**

Data Properties:

Data properties hold actual data values. They include attributes such as [[Value]], [[Writable]], [[Enumerable]], and [[Configurable]]. For instance:

Example:

const obj = {

key: 'value'

};

// Internal representation of the 'key' property

console.log(Object.getOwnPropertyDescriptor(obj, 'key'));

Accessor Properties:

Accessor properties are functions that execute when a property is read, written, or deleted. They consist of [[Get]] and [[Set]] functions. Here's an example:

Example:

const obj = {

get greeting() {

return 'Hello!';

}

};

// Internal representation of the 'greeting' accessor property

console.log(Object.getOwnPropertyDescriptor(obj, 'greeting'));