Write a blog about objects and its internal representation in Javascript

In JavaScript, objects are one of the fundamental data types, and they play a central role in the language's syntax and structure. Objects are collections of key-value pairs, where keys are strings (or symbols in ES6+) and values can be any data type, including other objects, functions, arrays, and primitive values like strings, numbers, and booleans.

**Internal Representation:**

**1. Properties:**

Each key-value pair in an object is called a property. Properties can be accessed using dot notation (**object.property**) or bracket notation (**object['property']**).

**2. Prototypes:**

JavaScript objects have an internal property called **[[Prototype]]** (accessible via **\_\_proto\_\_** in modern JavaScript). This links objects to other objects, forming a prototype chain. When you access a property of an object, JavaScript first looks for that property directly on the object. If it's not found, it looks at the object's prototype, then the prototype's prototype, and so on until it finds the property or reaches the end of the prototype chain.

**3. Descriptors:**

Each property of an object has associated meta-data called a property descriptor. Descriptors control how the property behaves when accessed, modified, or deleted. They include attributes like **writable**, **enumerable**, and **configurable**.

**4. Memory Representation:**

Objects in JavaScript are typically allocated memory on the heap. The actual representation of objects in memory might vary depending on the JavaScript engine being used (e.g., V8 for Chrome, SpiderMonkey for Firefox). Engines might use techniques like dictionary-based storage, hidden classes, or inline caching for performance optimization.

Let person = {

Key: Value

Id: 24,

Name: “keerti”,

designation: “uidesigner”,

status : “single”,

}

Console.log(desifnation);

Output:

Uidesigner