**Objects and its internal representation in Javascript**

In JavaScript, objects are fundamental data types that represent collections of key-value pairs. They are used to store various data types and can also contain functions, making them versatile for representing complex data structures.

The internal representation of objects in JavaScript can vary depending on the JavaScript engine being used (e.g., V8 in Chrome, SpiderMonkey in Firefox). However, conceptually, objects typically consist of properties and methods.

Breakdown of the internal representation and key concepts:

**Properties:**

* Properties are the key-value pairs that define the characteristics of an object. Each property has a name (or key) and a corresponding value. Properties can hold various data types, including strings, numbers, booleans, arrays, other objects, and even functions.

let person = {

name: "John",

age: 30,

isStudent: false

};

**Methods:**

* Methods are functions that are stored as property values. They define behavior for objects.

let person = {

name: "John",

age: 30,

greet: function() {

return "Hello, my name is " + this.name + " and I'm " + this.age + " years old.";

}

};

console.log(person.greet()); // Output: Hello, my name is John and I'm 30 years old.

**Prototype:**

* Each object in JavaScript has an internal property called [[Prototype]], which refers to another object (the prototype) that the current object inherits properties and methods from. This forms the prototype chain.

let person = {

name: "John",

age: 30

};

console.log(person.toString()); // Output: [object Object]

**Object Model:**

* JavaScript follows a prototype-based object model rather than a class-based model like some other languages. This means objects inherit properties and methods directly from other objects.

let person = {

name: "John",

age: 30

};

let student = {

major: "Computer Science"

};

// Inheriting properties from person

student.\_\_proto\_\_ = person;

console.log(student.name); // Output: John

**Property Descriptor:**

* Each property of an object has a property descriptor, which defines attributes such as value, writable, enumerable, and configurable.

let obj = {};

Object.defineProperty(obj, 'property1', {

value: 42,

writable: false // Cannot be changed

});

console.log(obj.property1); // Output: 42

obj.property1 = 100; // Ignored in strict mode

console.log(obj.property1); // Output: 42