OBJECTS AND ITS INTERNAL REPRESENTATION IN JAVASCRIPT

- In JavaScript, objects are fundamental data structures that represent collections of key-value pairs. These key-value pairs are often referred to as properties and methods. Objects can be used to model and store complex data structures, and they play a central role in the language. Here are the key points regarding object representation in JavaScript

1. Key-Value Pairs: JavaScript objects are essentially collections of key-value pairs. Each key is a unique string (or symbol in ES6 and later), and its associated value can be of any data type, including other objects, functions, or primitive values like numbers and strings.

2. Properties: The keys in an object are referred to as properties. You can access object properties using dot notation (e.g., objectName.propertyName) or bracket notation (e.g., objectName['propertyName']).

3. Methods: In addition to properties, objects can also have methods, which are functions associated with the object. Methods are simply properties with function values.

4. Literal Notation: You can create objects using object literal notation, which is a concise way to define an object

5. Constructor Functions: You can create objects using constructor functions, which are functions designed to create and initialize objects. ES6 introduced the class syntax, which makes it easier to define constructor functions.

6.Dynamic and Mutable: JavaScript objects are dynamic and mutable, meaning you can add, update, or delete properties and methods after the object is created.

Utilizing object methods for enhanced functionality in javascript

- Object methods in JavaScript are functions that are associated with objects. They can be used to enhance the functionality and behavior of those objects. Object methods are a fundamental part of object-oriented programming in JavaScript. Here's how you can utilize object methods for enhanced functionality in JavaScript

1. You can add methods to an object by defining functions as the values of properties within the object

2. You can call object methods using the dot notation

3. Object methods can provide enhanced functionality to objects. They allow you to encapsulate behavior and logic within an object, making it easier to work with and maintain.

4. Encapsulation and Abstraction

Object methods enable encapsulation and abstraction, two key principles of object-oriented programming. Encapsulation allows you to hide implementation details and expose only the necessary functionality. Abstraction allows you to work with objects at a higher level of abstraction, simplifying complex operations

Object inheritance and prototypes

- Object inheritance and prototypes are fundamental concepts in JavaScript that underlie the language's approach to object-oriented programming (OOP). These concepts allow you to create and share behavior between objects, promoting code reusability and maintaining a hierarchical structure. Let's explore object inheritance and prototypes in JavaScript.

- Objects and Prototypes:

1. In JavaScript, every object has a prototype, which is another object that it inherits properties and methods from.

2. An object's prototype can be accessed using the __proto__ property or the Object.getPrototypeOf() method.

- Object Creation

1. Objects can be created using object literal notation, constructor functions, or classes (introduced in ECMAScript 6).

2. Constructors and classes define a blueprint for creating objects with specific properties and methods.

Prototype Chain

1. When you access a property or method on an object, JavaScript searches for it in the object itself.

2. If it's not found in the object, JavaScript looks in the object's prototype (its parent or "super" object), and this process continues up the prototype chain until the property or method is found or the end of the chain is reached.

The prototype chain allows for property and method inheritance.

Prototype Object

1. In JavaScript, the prototype of an object is another object.
2. Functions (constructor functions or class methods) have a prototype property, which is an object used as the prototype for instances created with that function.

Inheritance via Prototypes

1. You can add properties and methods to an object's prototype, and these will be inherited by all objects created from that prototype.
2. This is a powerful way to achieve code reuse and implement inheritance in JavaScript.