Day 23 of Training

JavaScript: Object-Oriented Programming - Classes, Objects, and Inheritance

Today's session introduced the fundamental concepts of **Object-Oriented Programming (OOP)** in JavaScript, specifically focusing on **Classes**, **Objects**, and **Inheritance**. This is a powerful paradigm for structuring code and managing complex applications.

Revisiting Objects

We began by revisiting JavaScript objects, understanding them as entities that possess both **state (properties/variables)** and **behavior (methods/functions)**. We practiced creating simple objects with properties like name and marks, and methods such as printMarks. A key takeaway was the use of the this keyword within an object's methods, which refers to the object itself, allowing access to its internal properties (e.g., this.marks).

Understanding Prototypes

A critical concept introduced was the **Prototype** property, which is automatically associated with every JavaScript object. The prototype is itself an object, containing default properties and methods that the main object can inherit. This explains how built-in JavaScript objects like Arrays have methods like push() and pop() even when we don't explicitly define them. It's a mechanism for objects to inherit features from a "parent" prototype. If an object and its prototype both have a method with the same name, the object's own method takes precedence (Method Overriding).

Classes: Blueprints for Objects

We moved on to **Classes**, which serve as blueprints or templates for creating objects. Classes allow us to define a common structure for multiple similar objects, preventing code duplication when creating many instances of the same type. Key aspects of classes:

- **Syntax:** Classes are defined using the class keyword, followed by the class name.
- **constructor Method:** This is a special method within a class that is automatically invoked when a new object (instance) is created using the new keyword. The constructor is responsible for initializing the object's properties. If no constructor is explicitly defined, JavaScript provides a default one.
- **Creating Objects (Instances):** Objects are created from a class using the new keyword (e.g., let myCar = new Car();).

We practiced defining classes with properties and methods, observing how objects created from these classes automatically inherit these attributes and behaviors.