[Decorator Design Pattern]

The **Decorator Pattern** is a **structural design pattern** that dynamically adds behavior or responsibilities to an object at runtime without modifying its code. It's a flexible alternative to subclassing, as it allows you to add features to individual objects rather than entire classes.

Real-World Analogy:

Consider a coffee shop:

- You start with a basic coffee.
- You can add extra features like milk, sugar, whipped cream, or caramel syrup.
- Each addition builds upon the base coffee and other previous additions.

Key Concepts:

1. **Component**: The base interface or abstract class that defines common operations.
2. **Concrete Component**: The basic object to which additional responsibilities can be attached.
3. **Decorator**: An abstract class that wraps the component and adds new behavior.
4. **Concrete Decorators**: Specific implementations of decorators that add functionality to the component.

* Decorator in JS @ are example of this pattern