Good evening, my name is Elvis Leyva Sardon and today I'm going to talk about:

Dependency Inversion Principle in a JavaScript Example

Let's start by defining the Dependency Inversion Principle (DIP). This principle states that high-level modules should not depend on low-level modules. Instead, both should depend on abstractions.

That means the core logic of a system should not be tightly coupled to specific details like PayPal, Stripe, or Bank Transfer, but rather rely on a general abstraction.

- PaymentGateway.js

The first file is PaymentGateway.js. Here we define the abstraction or interface.

It's a base class with two methods: processPayment and refundPayment, which throw an error if they are not implemented.

This forces any class that extends it to provide its own implementation, ensuring consistency across all payment methods.

- /PaymentMethods/

Inside the PaymentMethods folder, we store the actual implementations for different payment options.

- PayPal.js

In PayPal.js, we create a concrete class that extends PaymentGateway.

It simulates a payment and a refund using PayPal, and it generates unique transaction IDs to mimic API calls.

- Other Methods

The same is done for other payment methods like Stripe and BankTransfer.

They also implement PaymentGateway, showing how we can easily switch or add payment providers without changing the core logic of our application.

- OrderService.js

OrderService.js represents the high-level module in our design.

It is responsible for handling orders and processing payments or refunds.

The key point is that it does not depend on PayPal, Stripe, or BankTransfer directly. It only depends on the PaymentGateway abstraction.

- index.js

Finally, in index.js, everything comes together.

We create instances of each payment method and inject them into the OrderService.

Then we simulate payments and refunds using each one.

- Final Output

As a result, we can see payments being processed with different payment methods, each generating its own unique transaction ID.

Then we perform a refund for each one, again generating a unique refund ID.

This clearly demonstrates that our system relies on a general abstraction, not on specific payment providers.

Thank you very much for your attention.