Comprehension Questions

In what ways do decorators support the principle of separation of concerns in application development?

How do decorators interact with TypeScript's runtime and compilation process? Are decorators purely compile-time features?

How do decorators enhance code maintainability and extensibility in large-scale projects?

What is the significance of the `Reflect` API in the context of decorators?

How does the Singleton pattern ensure that only one instance of a class exists throughout an application?

How does the Factory pattern differ from directly instantiating classes using the `new` keyword?

What is the core idea behind the Observer design pattern, and how does it help keep different parts of a system in sync?

How does the Strategy pattern differ from using conditional logic (like `if` or `switch` statements) inside a class?

Why are design patterns important in software engineering, and how do they contribute to code maintainability and scalability?

How does dependency injection contribute to the maintainability and modularity of a TypeScript codebase?

What are the advantages of injecting dependencies through constructors compared to setting them via properties or methods?

What are the advantages of injecting dependencies through constructors compared to setting them via properties or methods?

How can you identify which dependencies should be injected and which can be created internally within a class?