Core Concepts (Angular and .NET)

Angular Core Concepts

1. **What is Angular and how does it differ from other frameworks like React or Vue?**
    - Angular is a TypeScript-based open-source framework for building single-page web applications (SPAs). It uses two-way data binding, dependency injection, and a component-based architecture.
2. **Explain the difference between `ngOnInit` and `constructor` in Angular components.**
    - The `constructor` is used for dependency injection and is called before the `ngOnInit` lifecycle hook.
    - `ngOnInit` is part of Angular's lifecycle hooks, called after the component's constructor, and is typically used for initialization logic.
3. **What are directives in Angular?**
    - Directives are special markers in the DOM that extend HTML's capabilities by adding behavior to elements. There are three types:
        - **Structural directives** (like `*ngFor`, `*ngIf`) modify the DOM structure.
        - **Attribute directives** (like `ngClass`, `ngStyle`) modify the appearance or behavior of elements.
        - **Component directives**, which define a component in Angular.
4. **What is the purpose of services in Angular, and how do they work?**
    - Services in Angular provide a way to share data and logic across components. They are typically singleton instances created using Angular's dependency injection system.
5. **What are Observables in Angular?**
    - Observables are a part of RxJS and are used for handling asynchronous events in Angular. They allow you to subscribe to asynchronous data streams and manage events like HTTP requests, user inputs, etc.
6. **Explain how dependency injection works in Angular.**
    - Dependency Injection (DI) is a design pattern used to implement IoC (Inversion of Control). Angular's DI system provides instances of services to components or other services, which promotes loose coupling and easier testing.

.NET Core Concepts

1. **What is the difference between `IEnumerable` and `IQueryable` in .NET?**
    - `IEnumerable` represents a collection of objects that can be enumerated (iterated) but only executes in-memory filtering, whereas `IQueryable` is used for querying collections that are data-source-bound (e.g., databases) and supports deferred execution with filtering happening on the server-side.
2. **What is dependency injection in .NET Core, and how is it configured?**
    - Dependency Injection (DI) in .NET Core allows services to be injected into classes (like controllers or services). It is configured in the `ConfigureServices` method of the `Startup.cs` file using

`services.AddScoped()`, `services.AddSingleton()`, or `services.AddTransient()`.

3. **What is Entity Framework (EF) and how does it work?**
    - Entity Framework (EF) is an ORM (Object-Relational Mapping) tool that helps developers interact with a database using object-oriented programming (OOP) principles. EF Core is the cross-platform version of EF used in .NET Core.
4. **What is the difference between `async` and `await` in C#?**
    - `async` is used to declare a method that will run asynchronously, and `await` is used to indicate that the program should wait for the asynchronous task to complete before continuing execution.

SOLID Principles (in Object-Oriented Programming)

1. What are the SOLID principles, and why are they important?

SOLID is an acronym for five principles that improve the design and maintainability of object-oriented software:

- **S - Single Responsibility Principle**: A class should have only one reason to change, i.e., it should only have one responsibility.
- **O - Open/Closed Principle**: Software entities (classes, modules, functions) should be open for extension but closed for modification.
- **L - Liskov Substitution Principle**: Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.
- **I - Interface Segregation Principle**: Clients should not be forced to implement interfaces they don't use. Prefer small, specific interfaces over large, general ones.
- **D - Dependency Inversion Principle**: High-level modules should not depend on low-level modules. Both should depend on abstractions.

2. Explain the Single Responsibility Principle with an example in C# or Angular.

- A class should only have one reason to change. For example, if a class is responsible for both handling user authentication and logging errors, it violates this principle. You should separate these into two classes, each responsible for one task.

3. What is the Open/Closed Principle?

- The Open/Closed Principle suggests that a class should be open for extension (you should be able to add functionality to it) but closed for modification (you shouldn't change the existing code). In C#, this can be implemented using inheritance or interfaces to extend the functionality of a class without changing its original implementation.

4. How do you implement the Dependency Inversion Principle in .NET?

- This principle can be implemented in .NET Core by using Dependency Injection. Instead of classes depending directly on other concrete classes, they depend on interfaces, and concrete implementations are injected at runtime.

Object-Oriented Programming (OOP) Concepts

1. What are the four pillars of OOP?

- **Encapsulation**: Bundling data and methods that operate on the data within one unit (class), and restricting access to some of the object's components.
- **Abstraction**: Hiding complex implementation details and showing only the necessary functionality.
- **Inheritance**: A mechanism where one class inherits properties and behaviors (methods) from another class.
- **Polymorphism**: The ability to take many forms; allows objects of different types to be treated as instances of a common base type.

2. What is the difference between `interface` and `abstract class` in C#?

- **Interface**: Defines a contract with method signatures, without any implementation. A class can implement multiple interfaces.
- **Abstract Class**: A class that can have both implemented methods and abstract methods (methods without implementation). A class can inherit only one abstract class.

3. Explain the concept of method overloading and method overriding.

- **Method Overloading**: Involves defining multiple methods with the same name but with different parameters (different number or type of parameters).
- **Method Overriding**: Allows a subclass to provide a specific implementation of a method that is already defined in its superclass.

4. What is the difference between a value type and a reference type in C#?

- **Value Types** (e.g., `int`, `char`): Hold data directly and are stored in the stack. Assignment of value types copies the value.
- **Reference Types** (e.g., `string`, `class`): Hold a reference to the data stored in the heap. Assignment of reference types copies the reference, not the actual object.

5. What is polymorphism in OOP, and how does it work in C#?

- Polymorphism allows a class to define methods that are implemented differently in subclasses. In C#, you can achieve polymorphism using method overriding or interface implementation.

Additional Common Questions

1. **How do you handle error handling in Angular?**
    - In Angular, error handling can be done by using `try-catch` blocks, error handling services, and `HttpInterceptor` for global error handling for HTTP requests.
2. **Explain lazy loading in Angular.**
    - Lazy loading is a technique where modules are loaded only when they are needed, reducing the initial loading time of the application.
3. **What are middleware and routing in .NET Core?**
    - Middleware in .NET Core refers to components that handle HTTP requests and responses in the application's pipeline. Routing determines how an incoming request is mapped to a controller action.

SHORT QUESTIONS

Angular Core Concepts

1. **What is Angular?**
    - A TypeScript-based framework for building single-page web applications (SPAs).
2. **What is two-way data binding in Angular?**
    - Synchronization between model and view.
3. **What is the role of `ngOnInit` in Angular?**
    - Initializes component data after Angular first displays the component.
4. **What is a component in Angular?**
    - A building block of Angular apps, consisting of HTML, CSS, and TypeScript code.
5. **What is a service in Angular?**
    - A class used for data sharing and business logic.
6. **What is dependency injection in Angular?**
    - A design pattern that allows services to be injected into components or other services.
7. **What are directives in Angular?**
    - Special markers in the DOM to add behavior to elements.
8. **What is the difference between `ngFor` and `ngIf`?**
    - `ngFor`: Loops over data; `ngIf`: Conditionally displays content.
9. **What is RxJS in Angular?**
    - A library for reactive programming using Observables.
10. **What is an observable in Angular?**
    - A stream of asynchronous events that can be observed and acted upon.
11. **What is Angular CLI?**
    - A command-line interface tool for initializing, developing, scaffolding, and managing Angular applications.
12. **What are pipes in Angular?**
    - Transform data in templates, e.g., `date`, `currency`.
13. **What is the difference between Angular and React?**
    - Angular is a full framework, React is a library focused on the UI.

14. **What is a module in Angular?**
    - A container for related components, services, and other code.
15. **What is routing in Angular?**
    - The mechanism to navigate between views or components.

.NET Core Concepts

1. **What is .NET Core?**
    - A cross-platform, open-source framework for building applications.
2. **What is dependency injection in .NET?**
    - A design pattern used to provide services to classes without hard dependencies.
3. **What is Entity Framework?**
    - An Object-Relational Mapping (ORM) framework for .NET that allows interaction with databases using objects.
4. **What is the purpose of `ConfigureServices` in .NET?**
    - Configures services for dependency injection.
5. **What is an asynchronous task in C#?**
    - A task that runs independently of the main thread, allowing other operations to run concurrently.
6. **What is `IEnumerable` in .NET?**
    - Represents a collection of objects that can be iterated.
7. **What is `IQueryable` in .NET?**
    - Represents a collection of objects that can be queried and supports deferred execution.
8. **What is `async` and `await` in C#?**
    - `async`: Marks a method as asynchronous. `await`: Pauses the execution until the awaited task completes.
9. **What is a delegate in C#?**
    - A type that references a method, allowing methods to be passed as parameters.
10. **What is the difference between `new` and `new()` in C#?**
    - `new`: Creates an instance of a class. `new()`: Used with generics to specify a type constraint.
11. **What is middleware in .NET Core?**
    - Software components that process HTTP requests and responses.
12. **What is a controller in .NET?**
    - A class that handles HTTP requests and returns responses in a web application.
13. **What is the difference between .NET Framework and .NET Core?**
    - .NET Core is cross-platform and open-source, while .NET Framework is Windows-only.

SOLID Principles

1. **What does SOLID stand for?**

    - Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion.
2. **What is the Single Responsibility Principle?**
    - A class should have only one reason to change.
3. **What is the Open/Closed Principle?**
    - Classes should be open for extension, but closed for modification.
4. **What is the Liskov Substitution Principle?**
    - Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness.
5. **What is the Interface Segregation Principle?**
    - Clients should not be forced to implement interfaces they don't use.
6. **What is the Dependency Inversion Principle?**
    - High-level modules should not depend on low-level modules. Both should depend on abstractions.
7. **What is encapsulation?**
    - Hiding the internal state and requiring all interactions to be performed through well-defined methods.
8. **What is abstraction in OOP?**
    - Hiding complex implementation details and showing only the essential features.
9. **What is inheritance in OOP?**
    - A mechanism where one class inherits the properties and methods of another class.
10. **What is polymorphism?**
    - The ability to treat objects of different types through a common interface.
11. **What is method overloading?**
    - Defining multiple methods with the same name but different parameters.
12. **What is method overriding?**
    - Redefining a method in a subclass that is already defined in the parent class.
13. **What is constructor injection in DI?**
    - Injecting dependencies via a class constructor.
14. **What is the difference between interface and abstract class?**
    - Interface defines a contract with no implementation. Abstract class can have both defined and undefined methods.
15. **What is tight coupling?**
    - When classes are highly dependent on each other, making changes difficult.
16. **What is loose coupling?**
    - When classes have minimal dependency on each other, promoting flexibility.

OOP Concepts

1. **What is Object-Oriented Programming?**
    - A programming paradigm based on objects, which contain data and methods.
2. **What are the four pillars of OOP?**
    - Encapsulation, Abstraction, Inheritance, Polymorphism.
3. **What is a class in OOP?**
    - A blueprint for creating objects that define data and behavior.
4. **What is an object in OOP?**

    - An instance of a class.
5. **What is inheritance in OOP?**
    - A mechanism to derive a new class from an existing class.
6. **What is encapsulation in OOP?**
    - Bundling data and methods inside a class and restricting direct access.
7. **What is polymorphism in OOP?**
    - The ability for different classes to be treated as instances of the same class through inheritance.
8. **What is abstraction in OOP?**
    - Simplifying complex systems by exposing only essential components.
9. **What is a constructor in OOP?**
    - A method used to initialize objects when they are created.
10. **What is a destructor in OOP?**
    - A method used to clean up or release resources when an object is destroyed.
11. **What is a static method?**
    - A method that belongs to the class, not instances, and can be called without creating an object.
12. **What is method overriding in C#?**
    - Redefining a method in a subclass that has the same signature as the one in the superclass.
13. **What is an interface in OOP?**
    - A contract that defines a set of methods without providing implementation.
14. **What is an abstract class in OOP?**
    - A class that cannot be instantiated and may have abstract methods that need to be implemented by derived classes.
15. **What is a virtual method?**
    - A method in a base class that can be overridden in a derived class.