**Basic Angular Questions**

**Q1. What is Angular?**  
**A:** Angular is a **TypeScript-based, front-end framework** developed by Google for building **single-page applications (SPAs)**. It provides features like data binding, dependency injection, directives, components, and RxJS-based reactive programming.

**Q2. Difference between AngularJS and Angular?**

* AngularJS → based on JavaScript (ES5/ES6), uses controllers & $scope.
* Angular (2+) → based on TypeScript, uses components & modules, better performance.

**Q3. What are Components in Angular?**  
**A:** Components are the **basic building blocks** of Angular applications. They consist of:

* **HTML Template** (view)
* **TypeScript Class** (logic)
* **CSS/SCSS** (styling)  
  Defined using @Component decorator.

**Q4. What is a Module in Angular?**  
**A:** A module groups components, directives, pipes, and services. Every Angular app has at least one **root module (AppModule)**. Declared with @NgModule.(old version)

**Q5. What is Data Binding in Angular?**  
**A:** It allows communication between component and template:

* **Interpolation** → {{ expression }}
* **Property Binding** → [property]="value"
* **Event Binding** → (event)="handler()"
* **Two-way Binding** → [(ngModel)]="property"

Intermediate Angular Questions

**Q6. What are Directives in Angular?**  
**A:** Directives modify DOM behavior.

* **Component Directives** → have a template.
* **Structural Directives** → modify DOM structure (\*ngIf, \*ngFor).
* **Attribute Directives** → modify DOM element appearance/behavior (ngClass, ngStyle).

**Q7. What is Dependency Injection (DI)?**  
**A:** A design pattern where dependencies (services) are **injected** into a class instead of creating them manually. Angular’s DI system makes code reusable and testable.

**Q8. What is RxJS in Angular?**  
**A:** Reactive Extensions for JavaScript (RxJS) is a library for **reactive programming** using Observables. Angular uses it for handling async operations like HTTP calls and event streams.

**Q9. Difference between Observable and Promise?**

* **Promise** → handles one async event, eager, not cancellable.
* **Observable** → handles multiple values over time, lazy, cancellable, supports operators like map, filter.

**Q10. What are Lifecycle Hooks in Angular?**  
**A:** Methods that allow you to tap into key events in a component’s lifecycle.

* ngOnInit() – runs after component initialized
* ngOnChanges() – runs when input properties change
* ngDoCheck() – custom change detection
* ngOnDestroy() – cleanup before component destroyed

**Advanced Angular Questions**

**Q11. What is Change Detection in Angular?**  
**A:** Process where Angular checks if the **model** (component data) has changed and updates the **view** accordingly. Can use **Default** or **OnPush** strategies.

**Q12. What is Angular Routing?**  
**A:** Routing allows navigation between views/components. Defined in app-routing.module.ts with RouterModule.forRoot(routes). Supports lazy loading, route guards, and parameterized routes

**Q13. What is Lazy Loading?**  
**A:** Technique to load modules only when needed (on-demand), improving initial load time. Configured in routes using loadChildren.

**Q14. What are Angular Guards?**  
**A:** Guards control navigation:

* **CanActivate** → check before navigating
* **CanDeactivate** → check before leaving
* **Resolve** → fetch data before navigating
* **CanLoad** → check before lazy loading a module

**Q15. What is Ahead-of-Time (AOT) Compilation?**  
**A:** Angular compiles templates **at build time** instead of runtime (JIT). This improves performance, security, and smaller bundle size.

**Q16. What are Angular Pipes?**  
**A:** Pipes **transform** data in templates. Example:

* Built-in: date, uppercase, async
* Custom: by creating @Pipe({ name: 'myPipe' })

**Q17. How does Angular handle forms?**

* **Template-driven forms** → simple, declarative (ngModel)
* **Reactive forms** → more control, uses FormGroup, FormControl, FormBuilder.

**Q18. What is NgZone in Angular?**  
**A:** NgZone helps Angular detect async operations (like setTimeout, HTTP requests) and trigger change detection automatically.

**Q19. What are Angular Interceptors?**  
**A:** Interceptors sit between the app and backend. Common uses:

* Add authentication tokens
* Logging requests/responses
* Error handling

**Q20. How do you optimize Angular app performance?**

* Use **OnPush change detection**
* Use **trackBy** with \*ngFor
* Implement **Lazy Loading**
* Minimize bundle size with **AOT & optimization flags**
* Use **pure pipes** instead of functions in templates

**Constructor VS ngOnInit**

**Constructor**

* A **TypeScript/JavaScript feature**, not Angular-specific.
* Used for **class instantiation** and **dependency injection**.
* Called **before** Angular initializes the component lifecycle.
* Good for: injecting services or initializing simple variables.
* ❌ Not good for: calling Angular-specific APIs (like fetching input values) because bindings are not yet ready.

**ngOnInit()**

* An **Angular lifecycle hook** (OnInit interface).
* Called **once after the component is initialized** and Angular sets input properties.
* Good for: fetching data from APIs, initializing values that depend on @Input(), setting up subscriptions.

*How do you handle exceptions in Angular?*

→ Answer:

* Local try-catch
* catchError with RxJS
* Global ErrorHandler
* HTTP Interceptors for centralized handling