
✅ 1. What is Angular and how is it different from AngularJS?

AngularJS (Old)	Angular (Modern)
Version 1.x	Version 2+ (Current: Angular 17+)
Based on **JavaScript**	Based on **TypeScript** (Super powerful JS)
Uses Controllers & $scope	Uses Components & Services
Two-way binding by default	Two-way binding with better performance
Less performance, not mobile-friendly	Fast, optimized for mobile and web

✅ Example:

AngularJS: Old scooter
Angular: Modern electric bike (Fast + New Tech)

✅ 2. What are the key features of Angular?

- **Component-based architecture**
- **Two-way Data Binding**
- **Dependency Injection (DI)**
- **Directives** (for DOM behavior)
- **Routing** (for navigation between pages)
- **Reactive Programming with RxJS**
- **Built-in Testing Support**

✅ 3. What is a component in Angular?

A **component** is like a building block of an Angular app.

✅ Example:

If your app is a house:

- Kitchen → One component
- Bedroom → Another component

Each component has:

1. **HTML** (Template)

2. **CSS** (Style)
3. **TypeScript** (Logic)

```
@Component({
  selector: 'app-hello',
  template: `<h1>Hello!</h1>`
})
export class HelloComponent {}
```

✅ 4. What is a module in Angular?

A **module** is a container to group related **components**, **services**, and other modules together.

It organizes the app and makes it easier to manage.

```
@NgModule({
  declarations: [HelloComponent],
  imports: [BrowserModule],
  bootstrap: [HelloComponent]
})
export class AppModule {}
```

✅ 5. What is the role of NgModules?

- Organize your app into small, manageable parts.
- Help in **Lazy Loading** (load only when needed).
- Allow **sharing** components and services across modules.

✅ 6. What is a service in Angular?

A **service** contains logic that you can use in multiple components (for sharing data, APIs, etc.)

✅ Example:

```
@Injectable({ providedIn: 'root' })
export class DataService {
  getData() {
    return 'Data from Service';
  }
}
```

✅ 7. What is Dependency Injection in Angular?

Dependency Injection (DI) means giving components their needed objects (services) automatically.

✅ Example:

```
constructor(private dataService: DataService) {}
```

Angular **injects** the service, and you can use it without creating it manually.
It improves **reusability** & **testability**.

✅ 8. What is the difference between a component and a directive?

Feature	Component	Directive
Purpose	UI building block	Add behavior to existing elements
Has Template?	YES	NO
Usage	To create views/pages	To modify or manipulate DOM elements

✅ Example:

- **Component** → `<app-header></app-header>`
- **Directive** → `*ngIf`, `*ngFor`

✅ 9. What are lifecycle hooks in Angular?

Lifecycle Hooks are special methods that Angular calls at specific stages of a component's life (like birth, growth, and death).

Hook	When It's Called
`ngOnInit`	After component is created and inputs set (init phase)
`ngOnChanges`	When input values change
`ngAfterViewInit`	After component's view (DOM) is initialized
`ngOnDestroy`	Before component is removed (cleanup time)

✅ 10. What is the purpose of ngOnInit()?

It runs when the component is created and ready.

✅ Use it for:

- API calls
- Initializing variables
- Fetching data

✅ Example:

```
ngOnInit() {
  console.log('Component initialized');
}
```

✅ 🎯 Simple Summary:

Concept	Meaning (Simple Words)
Angular	Modern app framework (Fast & Scalable)
Component	Small piece of UI (like Lego block)
Module	Group of components/services
Service	Code to share between components (like helper)
Dependency Injection	Auto-giving service to components
Directive	Adds extra behavior to elements
Lifecycle Hooks	Special events in component's life
ngOnInit	Runs when component is born (start things)

Here is a **simple, clear explanation** with **examples** for your Angular questions on **Data Binding** & **Directives** in easy language:

🔷 2. Data Binding in Angular

✅ 11. What are the types of data binding in Angular?

There are **4 main types** of data binding in Angular:

Type	Direction	Usage
Interpolation	Component → Template	Show data in HTML
Property Binding	Component → DOM Element	Set element properties
Event Binding	DOM → Component	Handle user actions
Two-Way Binding	Both ways	Sync data in both directions

✅ 12. What is two-way data binding?

- **Two-way data binding** means data flows both from **component → template** and **template → component**.
- Changes in the UI automatically update the component & vice-versa.

✅ Example using `[(ngModel)]`:

```
<input [(ngModel)]="userName" />
<p>Hello, {{ userName }}!</p>
```

✅ 13. How is one-way data binding implemented?

One-way data binding means data flows in **only one direction**.

✔️ *Example:*

```
<!-- Interpolation -->
<p>{{ userName }}</p>

<!-- Property Binding -->
<img [src]="profilePicUrl" />

<!-- Event Binding -->
<button (click)="sayHello()">Click Me</button>
```

✅ 14. What is [(ngModel)]?

- `[(ngModel)]` is a **directive** used for two-way data binding in Angular.
- It **syncs** the component data with the input field.

✔️ *Example:*

```
<input [(ngModel)]="email" placeholder="Enter Email" />
<p>Your email: {{ email }}</p>
```

i Important: You must import `FormsModule` in your module to use `ngModel`.

✅ 15. How does event binding work in Angular?

- Event binding listens to events (like `click`, `keyup`) and triggers a method in the component.

✔️ *Example:*

```
<button (click)="submitForm()">Submit</button>
submitForm() {
  console.log('Button Clicked!');
}
```

🔷 3. Directives in Angular

✅ 16. What is the difference between structural and attribute directives?

Feature	Structural Directive	Attribute Directive
Purpose	Change DOM structure (Add/Remove elements)	Change element's appearance or behavior
Syntax Example	`*ngIf`, `*ngFor`, `*ngSwitch`	`ngClass`, `ngStyle`, `hidden`, `disabled`
Symbol Used	Starts with `*` (asterisk)	No `*` needed

✅ *17. What does ngIf do?

- Conditionally shows or hides elements based on **true/false** value.

✔️ *Example:*

```
<div *ngIf="isLoggedIn">Welcome, User!</div>
```

✅ *18. What does ngFor do?

- It repeats the DOM element for every item in an array (for loops).

✔️ *Example:*

```
<ul>
  <li *ngFor="let product of products">{{ product.name }}</li>
</ul>
```

✅ 19. What is ngSwitch used for?

- Used for **multiple conditions** (like switch-case).

✔️ *Example:*

```
<div [ngSwitch]="userRole">
  <p *ngSwitchCase="'admin'">Admin Panel</p>
  <p *ngSwitchCase="'user'">User Dashboard</p>
  <p *ngSwitchDefault>Unknown Role</p>
</div>
```

✅ 20. How do you create a custom directive?

Steps:

1. Create a directive with `@Directive`.
2. Use `ElementRef` to access the DOM element.
3. Use `Renderer2` for safe DOM changes.

Example: (Change text color directive)

```
@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
  constructor(el: ElementRef) {
    el.nativeElement.style.color = 'red';
  }
}
```

Usage:

```
<p appHighlight>Highlighted Text!</p>
```

Simple Summary:

Concept	Meaning (Simple Words)
Interpolation	Display value inside HTML (`{{ value }}`)
Property Binding	Bind property (`[src]`, `[disabled]`)
Event Binding	Handle user actions (`(click)`)
Two-way Binding	Sync data both ways (`[(ngModel)]`)
Structural Directive	Add/Remove elements in DOM (`*ngIf`, `*ngFor`)
Attribute Directive	Change look or behavior (`ngClass`, `ngStyle`)
*ngIf	Show/hide element based on condition
*ngFor	Loop through data
ngSwitch	Multiple conditions (switch-case)
Custom Directive	Your own directive to customize DOM

---

4. Components and Templates

---

21. What is ViewEncapsulation?

ViewEncapsulation controls whether the **CSS** styles of a component affect only its template or also leak outside.

3 Types:

Type	Meaning
Emulated *(default)*	Styles apply only inside component (safe & scoped).
None	Styles apply globally (dangerous; affects all).
ShadowDom	Uses browser's native Shadow DOM (advanced).

Example:

```
@Component({
  selector: 'app-demo',
  templateUrl: './demo.component.html',
  styleUrls: ['./demo.component.css'],
  encapsulation: ViewEncapsulation.Emulated // Default
})
```

✅ 22. What is the use of @Input() and @Output()?

- **@Input()** → Pass data **from parent to child**.
- **@Output()** → Send events/data **from child to parent**.

Example:

```
// Child Component
@Input() productName: string;
@Output() notify = new EventEmitter<string>();
```

✅ 23. How do you pass data from child to parent component?

- Use **@Output()** with **EventEmitter**.

Example:

```
// Child Component
@Output() message = new EventEmitter<string>();
sendMessage() {
  this.message.emit('Hello Parent!');
}

// Parent Component
<app-child (message)="receiveMessage($event)"></app-child>
```

✅ 24. What is content projection?

It means placing content inside a component from the **outside**.

Example:

```
// Child Component Template
<ng-content></ng-content>
```

✔️ *Usage:*

```
<app-card>
  <h1>Welcome!</h1>
</app-card>
```

✔️ 25. How do you communicate between two unrelated components?

→ Use a **Shared Service** with **Subject/BehaviorSubject**.

✔️ *Example:*

```
@Injectable({providedIn: 'root'})
export class DataService {
  private messageSource = new BehaviorSubject<string>('Default');
  currentMessage = this.messageSource.asObservable();

  changeMessage(msg: string) {
    this.messageSource.next(msg);
  }
}
```

Both components can subscribe and emit messages via this service.

🔷 5. Forms in Angular

✔️ 26. Difference between Template-driven and Reactive Forms?

Feature	Template-driven	Reactive Forms
Approach	Simple, Easy, HTML-based	Complex, Controlled by TS code
Validation	In Template	In Component (TS)
Flexibility	Less Flexible	Highly Flexible
Use Case	Simple Forms	Complex Forms (Dynamic Forms)

✔️ 27. What are FormControl and FormGroup?

- **FormControl** → Represents a single form input.
- **FormGroup** → A group of FormControls.

✔️ *Example:*

```
this.form = new FormGroup({
  name: new FormControl(''),
  email: new FormControl(''),
```

```
});
```

✅ 28. How do you add validation in reactive forms?

By passing validators to FormControl:

```
this.form = new FormGroup({
  name: new FormControl('', Validators.required),
});
```

✅ 29. What is FormBuilder?

It is a shortcut service to create forms quickly.

✔️ *Example:*

```
this.form = this.fb.group({
  name: ['', Validators.required],
  email: [''],
});
```

✅ 30. How do you show error messages dynamically in forms?

✔️ *Example:*

```
<input formControlName="name" />
<div *ngIf="form.get('name')?.errors?.required">
  Name is required!
</div>
```

🔷 6. Pipes

✅ 31. What is a pipe in Angular?

Pipes transform data in the template.

✔️ *Example:*

```
{{ 12345.6789 | number:'1.2-2' }}
```

✅ 32. What are built-in pipes in Angular?

Pipe	Use
date	Format dates

Pipe	Use
currency	Currency formatting
number	Number formatting
percent	Percent formatting
json	JSON representation
uppercase	Convert text to UPPERCASE
lowercase	Convert text to lowercase

✅ 33. How do you create a custom pipe?

✔️ *Example:*

```
@Pipe({name: 'reverse'})
export class ReversePipe implements PipeTransform {
  transform(value: string): string {
    return value.split('').reverse().join('');
  }
}
```

✔️ *Usage:*

```
{{ 'hello' | reverse }}
```

✅ 34. What is a pure and impure pipe?

Pipe Type	Behavior
Pure	Runs only if input changes (default)
Impure	Runs on every change detection cycle

✔️ *Example:*

```
@Pipe({ name: 'myPipe', pure: false })
```

✅ 35. How do you use async pipe?

`async` pipe automatically subscribes to **Observable/Promise** and unsubscribes too.

✔️ *Example:*

```
<p>{{ myObservable$ | async }}</p>
```

✅ Simple Summary:

Feature	Explanation
@Input()	Receive data from parent
@Output()	Send data to parent
Content Projection	Inject content inside component
Reactive Forms	Controlled by TypeScript (form builder approach)
Template Forms	Defined in HTML (simple)
Pipe	Format data in the template
Async Pipe	Auto subscribe to Observables/Promises

🔷 7. Routing and Navigation

✅ 36. How does Angular routing work?

Angular uses a **Router Module** to navigate between components (pages) without reloading the page.

✔️ *Example:*

```
const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'about', component: AboutComponent },
];
<a routerLink="/home">Go to Home</a>
```

✅ 37. What are route guards?

Guards control whether navigation to a route should be allowed or not.

✅ 38. Difference between CanActivate and CanDeactivate?

Guard Type	Purpose
CanActivate	Prevent entering a route (before navigating).

Guard Type	Purpose
CanDeactivate	Prevent leaving a route/component.

✔️ *Example:*

```
canActivate(): boolean {
  return isLoggedIn;
}
```

✅ 39. How do you pass parameters in routes?

✔️ *Example:*

```
{ path: 'user/:id', component: UserComponent }
```

→ Navigate: `/user/101`

→ Inside Component:

```
this.route.snapshot.paramMap.get('id');
```

✅ 40. What is lazy loading?

Lazy loading means loading modules **only when needed** (on-demand), improving app speed.

✔️ *Example:*

```
{ path: 'admin', loadChildren: () => import('./admin/admin.module').then(m
=> m.AdminModule) }
```

🔷 8. Observables and RxJS

✅ 41. What is an Observable?

Observable is a way to **manage async data** like HTTP calls, user inputs, etc.

✔️ *Example:*

```
of(1, 2, 3).subscribe(data => console.log(data));
```

✅ 42. Difference between Observable and Promise?

Feature	Observable	Promise
Data	Multiple values	Single value

Feature	Observable	Promise
Lazy?	Yes (runs only when subscribed)	No
Cancelable?	Yes (unsubscribe)	No
Operators	Yes (map, filter, etc.)	No

✅ 43. What is Subject and BehaviorSubject?

Term	Meaning
Subject	Broadcast to multiple subscribers, no initial value.
BehaviorSubject	Same as Subject but holds **latest value**.

✔️ *Example:*

```
const subject = new BehaviorSubject(0);
subject.subscribe(data => console.log(data));
subject.next(1);  // emits 1
```

✅ 44. Common RxJS Operators:

Operator	Purpose
map	Transform data
filter	Filter values
take	Take limited values
switchMap	Cancel previous & start new observable
debounceTime	Delay execution (for search etc.)

✅ 45. How do you unsubscribe from an observable?

✔️ *Example:*

```
this.subscription = this.service.getData().subscribe();
ngOnDestroy() {
  this.subscription.unsubscribe();
}
```

🔷 9. HTTP & Interceptors

✅ 46. How do you make HTTP calls in Angular?

✅ Example:

```
this.http.get('api/data').subscribe(data => console.log(data));
```

✅ 47. What is an HTTP Interceptor?

Interceptor allows you to **modify all HTTP requests or responses globally**.

✅ Example:

- Add token, logs, error handling, etc.

✅ 48. How do you handle errors globally?

Use Interceptor + `catchError` operator.

✅ Example:

```
return next.handle(req).pipe(
  catchError(error => {
    console.error('Error:', error);
    return throwError(() => error);
  })
);
```

✅ 49. How do you add headers to every HTTP request?

In **Interceptor**:

```
const modifiedReq = req.clone({
  headers: req.headers.set('Authorization', 'Bearer TOKEN')
});
```

✅ 50. What is HttpClientModule?

`HttpClientModule` provides Angular's **HTTP client** to make API calls.

✅ Usage:

```
import { HttpClientModule } from '@angular/common/http';
@NgModule({
  imports: [HttpClientModule]
})
export class AppModule { }
```

✅ Simple Summary:

Concept	Meaning
Routing	Navigate between pages
Guards	Protect routes
Observables	Manage async data streams
Subject/BehaviorSubject	Broadcast data between components
RxJS Operators	Transform or filter data streams
HTTP Client	Make API calls
Interceptor	Modify requests/responses globally (ex: auth tokens)

🔷 10. Angular CLI & Build Tools

✅ 51. What is Angular CLI?

Angular CLI (**Command Line Interface**) is a tool to:

- Create Angular apps.
- Generate components, services, modules, etc.
- Build, run & test Angular projects.

✅ *Example:*

```
ng new my-app  # create new project
ng g component header  # generate component
```

✅ 52. What is difference between ng serve & ng build?

Command	Purpose
ng serve	Starts local dev server + rebuilds automatically.
ng build	Builds app for **production** (creates `/dist` folder).

✅ 53. What are environments in Angular?

Environment files store **different settings** for different environments.

✔️ *Example:*

- `environment.ts` → for development
- `environment.prod.ts` → for production

```
export const environment = {
  production: false,
  apiUrl: 'http://localhost:3000'
};
```

✔️ 54. How do you create a new module using CLI?

```
ng g module user
```

✔️ 55. What is AOT Compilation?

AOT (**Ahead-Of-Time**) Compilation compiles Angular templates **before** browser runs them.
→ Faster load, smaller bundle size.

🔷 11. Change Detection

✔️ 56. What is change detection in Angular?

It checks for changes in data and updates the view (HTML) automatically.

✔️ 57. Difference between Default and OnPush Strategy:

Strategy	Meaning
Default	Checks for all changes in entire component tree.
OnPush	Only checks if @Input() changes or event triggers it.

✔️ 58. How does ChangeDetectorRef work?

It manually triggers or stops change detection in components.

✔️ *Example:*

```
constructor(private cd: ChangeDetectorRef) {}
this.cd.detectChanges();  // manually trigger detection
```

✅ 59. What causes performance issues in change detection?

- Large DOM with complex bindings.
- Many event listeners.
- Too many nested components.
- Heavy operations inside templates.

✅ 60. When should you use OnPush strategy?

- When component's data doesn't change often.
- For performance optimization in large apps.
- When using Immutable data or Observables.

🔷 12. State Management

✅ 61. What is state management?

Managing app data (state) centrally for easy sharing between components.

✅ 62. What is NgRx?

NgRx is a **state management library** for Angular, based on **Redux** pattern.

✅ 63. What are actions, reducers, effects in NgRx?

Term	Meaning
Actions	Events that describe what to do.
Reducers	Functions that change state based on actions.
Effects	Handle async tasks (API calls) and dispatch actions.

✅ 64. What is a store in NgRx?

A single object that holds the app's entire state.

✅ 65. How does NgRx compare with Services & BehaviorSubject?

Feature	NgRx	Services + BehaviorSubject
Data Flow	Strict (actions, reducers, effects)	Simple, direct
Learning Curve	High	Low
Debugging	Easy with DevTools	Harder for complex apps

🔷 13. Advanced Topics

✅ 66. What is Module Federation?

Allows loading remote modules from other apps at runtime → useful for **micro frontends**.

✅ 67. What are Micro Frontends?

Micro Frontends break large apps into **small, independent apps** that work together.

✅ 68. What is SSR (Server-Side Rendering)?

Rendering Angular app on **server** to send fully rendered page to the browser → improves SEO & load speed.

✅ 69. How does Angular Universal work?

Angular Universal enables SSR in Angular.
It runs Angular code on the server & generates static HTML for faster load.

✅ 70. What is the purpose of zone.js?

`zone.js` tracks async operations (like HTTP calls, setTimeout) to:

- Auto-detect changes.
- Trigger Angular change detection automatically.

☑ Summary Table:

Section	Key Idea
CLI & Build	Commands for building, serving & generating code
Change Detection	Auto UI updates when data changes
State Management	Manage shared app data with NgRx or Services
Advanced Topics	Modern Angular concepts like Micro Frontends, SSR

Here's a **detailed explanation** for these Angular **performance optimization** questions with examples in **simple language**:

🔷 14. Performance Optimization

71. How do you optimize an Angular app?

Here are some **common optimization techniques**:

1. **Lazy Loading**
   Load feature modules only when needed instead of loading all at once.
2. **Use OnPush Change Detection**
   Reduces unnecessary change detection cycles.

```
@Component({
  selector: 'app-example',
  templateUrl: './example.component.html',
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class ExampleComponent { }
```

3. **Avoid Memory Leaks**
   Unsubscribe from Observables to prevent memory issues:

```
ngOnDestroy() {
  this.subscription.unsubscribe();
}
```

4. **Use TrackBy with ngFor*
   Improves rendering of large lists:

```
<li *ngFor="let item of items; trackBy: trackById">
```

```
  {{ item.name }}
</li>
trackById(index: number, item: any) {
  return item.id;
}
```

5. **Minimize Bundle Size (via Lazy Loading, Tree Shaking, etc.)**

---

72. What is Lazy Loading?

Lazy Loading means loading feature modules **only when required** (not at app start).

→ This reduces the **initial load time** of the app.

Example:

```
// app-routing.module.ts
const routes: Routes = [
  {
    path: 'products',
    loadChildren: () => import('./products/products.module').then(m =>
m.ProductsModule)
  }
];
```

When users visit `/products`, Angular will load the **ProductsModule**.

---

73. How do you reduce bundle size?

Here are practical ways to reduce bundle size:

1. **Lazy Loading** (load modules on demand).
2. **Tree Shaking** (removes unused code automatically during production builds).
3. **Minification & Compression** (done by Angular CLI during build).
4. **Use Lightweight Libraries** (avoid heavy, unused libraries).
5. **Remove Unused Features or Polyfills**.
6. **Optimize Images & Assets**.

```
ng build --prod --optimization
```

---

74. What is Tree Shaking?

Tree Shaking automatically removes **unused code** from the final bundle during build.

→ It's like "shaking the tree" to drop dead leaves (unused code).

Example:
If you import a library but only use 1 function:

```
import { usefulFunction } from 'my-lib';
```

Angular will **remove** other unused parts of `my-lib` from the bundle.

✅ It happens automatically during `ng build --prod`.

---

75. What are best practices for Angular performance?

Here's a list of **best practices**:

Best Practice	Description
Lazy Loading	Load modules on demand
OnPush Change Detection	Detect changes only when necessary
TrackBy with *ngFor	Optimize rendering of large lists
Minify Images	Compress image sizes
Avoid Memory Leaks	Always unsubscribe from Observables
Avoid Complex Logic in Templates	Move heavy logic to TypeScript component code
Bundle Analyzer Tools	Analyze bundle size using tools like `webpack-bundle-analyzer`
Avoid Using Many Global Variables	Keep components self-contained and small
Use Angular CLI Production Builds	Builds optimized bundles automatically
Prefer Pure Pipes	Pure pipes are faster as they re-execute only when inputs change

---

✅ Simple Summary:

- Lazy Loading → Load on demand.
- Tree Shaking → Remove unused code.
- OnPush → Less change detection.
- Bundle optimization → Smaller, faster app.
- TrackBy → Faster list rendering.

Here's the **detailed explanation** for Angular questions on **Testing & Security** with **examples** in simple language:

🔷 15. Testing

76. What is Jasmine?

Jasmine is a **testing framework** used to write unit tests in Angular.
It provides:

- Functions like `describe`, `it`, `expect`.
- Easy syntax for testing logic.

✅ **Example:**

```
describe('Math test', () => {
  it('should add numbers', () => {
    expect(1 + 2).toBe(3);
  });
});
```

---

77. What is Karma?

Karma is a **test runner** used to run your Jasmine tests automatically in a browser environment.

- Executes your test cases.
- Shows results in terminal or browser.
- Continues watching code changes if required.

✅ Command:

```
ng test
```

---

78. What is a TestBed in Angular?

`TestBed` is the **main utility** to create a test environment for Angular components, services, and directives.

It allows:

- Component testing in isolation.
- Mocking dependencies.

✅ **Example:**

```
beforeEach(() => {
  TestBed.configureTestingModule({
    declarations: [MyComponent],
  }).compileComponents();
});
```

79. How do you write a unit test for a component?

✅ **Example:**

```
import { TestBed } from '@angular/core/testing';
import { MyComponent } from './my.component';

describe('MyComponent', () => {
  let component: MyComponent;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [MyComponent]
    });
    const fixture = TestBed.createComponent(MyComponent);
    component = fixture.componentInstance;
  });

  it('should create component', () => {
    expect(component).toBeTruthy();
  });
});
```

---

80. What is the difference between unit testing and e2e testing?

Feature	Unit Testing	End-to-End (E2E) Testing
Scope	Tests individual functions/components	Tests entire app behavior
Tools	Jasmine, Karma	Protractor, Cypress
Speed	Very fast	Slower (needs browser interaction)
Example	Test a service method	Test login flow through browser
Usage	Validate internal logic	Validate UI, routing, APIs together

---

🔐 16. Security

81. What is XSS and how does Angular prevent it?

XSS (Cross Site Scripting):
Attack where malicious scripts are injected into web pages.

🔒 **Angular Protection:**
Angular **auto-sanitizes** values in templates:

```
<div>{{ userInput }}</div> <!-- Safe -->
```

It automatically escapes dangerous code.

---

82. What is CSP?

CSP (Content Security Policy) is a browser security feature to prevent XSS attacks.

- Restricts where scripts, styles, etc. can be loaded from.
- You set it via HTTP headers:

```
Content-Security-Policy: default-src 'self';
```

---

83. What is DOM sanitization?

Sanitization means removing dangerous parts from HTML.

➡️ Example in Angular:

```
import { DomSanitizer } from '@angular/platform-browser';

safeHtml = this.sanitizer.bypassSecurityTrustHtml(dirtyHtml);
```

Angular auto-sanitizes most bindings, but in special cases, you can control it manually.

---

84. How to securely bind HTML in Angular?

By default, Angular sanitizes HTML using `innerHTML`:

```
<div [innerHTML]="userHtml"></div>
```

If you're sure the HTML is safe (not recommended unless absolutely necessary):

```
this.safeHtml = this.sanitizer.bypassSecurityTrustHtml(userHtml);
```

---

85. How do you handle authentication and authorization?

✅ **Authentication:** Verifying user identity (login).
✅ **Authorization:** Granting user access to specific parts (roles, permissions).

Common Methods in Angular:

- Use **JWT tokens** for login.
- Store tokens in `localStorage` or `sessionStorage`.
- Use route guards for protecting routes.

✅ Example Route Guard:

```
canActivate(): boolean {
  return !!localStorage.getItem('authToken');
}
```

✅ Simple Summary:

- Jasmine → Testing framework.
- Karma → Test runner.
- TestBed → Test environment.
- XSS → Auto-handled by Angular.
- CSP → Browser policy to block risky content.
- Always sanitize HTML inputs before binding!

Here are the detailed, simple explanations with examples for **Deployment, Miscellaneous, and Real-time Features** in Angular:

🔷 17. Deployment

86. How do you deploy an Angular application?

1. Run:

```
ng build --prod
```

This creates `dist/` folder with production files.

2. Deploy `dist/your-project` folder to:

- Apache/Nginx
- Firebase Hosting
- AWS S3
- Any web server

---

87. What is the purpose of baseHref?

It tells Angular the **root URL** of the app.

✅ Example (in `index.html`):

```
<base href="/">
```

If app is deployed in a subfolder like `/myapp/`:

```
<base href="/myapp/">
```

---

88. How do you use HashLocationStrategy vs PathLocationStrategy?

Strategy	Explanation	Example URL
PathLocationStrategy (default)	Clean URLs using History API	`https://example.com/home`
HashLocationStrategy	Uses `#` in URL (no server config needed)	`https://example.com/#/home`

✅ Example:

```
providers: [{ provide: LocationStrategy, useClass: HashLocationStrategy }]
```

---

89. What is Angular Universal?

Angular Universal is **Server-Side Rendering (SSR)** for Angular apps.

- Renders app on server → sends HTML to client.
- Improves SEO & faster first page load.

---

90. How do you handle 404s in Angular?

You can add a **wildcard route**:

```
{ path: '**', component: NotFoundComponent }
```

This route catches all unknown URLs and shows a **404 page**.

🔷 18. Miscellaneous

91. What are decorators in Angular?

Decorators add extra info to classes, methods, or properties.

✅ Examples:

- `@Component` → For components.
- `@Injectable` → For services.
- `@Input` / `@Output` → For data binding.

---

92. Difference: ng-content vs ng-template vs ng-container?

Directive	Purpose	Example Use
ng-content	Content projection (slot-like)	Inject parent content inside child
ng-template	Defines template that's rendered later	Conditional or dynamic templates
ng-container	Group elements without adding extra DOM	Wrap multiple directives without DOM

✅ Simple Example:

```
<ng-container *ngIf="show">
  <div>Visible</div>
</ng-container>
```

---

*93. What is the role of trackBy in ngFor?

`trackBy` helps Angular track items in a loop by unique IDs to **avoid re-rendering**.

✅ Example:

```
<li *ngFor="let item of items; trackBy: trackById">{{ item.name }}</li>
trackById(index: number, item: any) {
  return item.id;
}
```

---

94. What is differential loading?

Angular creates **two bundles**:

- Modern JS (`ES2015+`) for new browsers.
- Legacy JS (`ES5`) for old browsers.

✅ It improves loading speed for modern browsers automatically.

---

95. What is the purpose of APP_INITIALIZER?

`APP_INITIALIZER` runs **before the app starts**—usually to load config or settings.

✅ Example:

```
providers: [
  {
    provide: APP_INITIALIZER,
    useFactory: loadConfig,
    deps: [ConfigService],
    multi: true
```

```
  }
]
```

🔷 19. Real-time Features

96. How do you integrate WebSocket in Angular?

Use `WebSocket` API or libraries like `socket.io-client`.

✅ Example:

```
const socket = new WebSocket('ws://example.com/socket');
socket.onmessage = (event) => console.log(event.data);
```

---

97. How do you handle real-time updates in Angular?

- Use **WebSockets** for instant updates.
- Use **RxJS Subjects** to share updates across components.

---

98. How do you implement polling or long polling?

Polling: Call API repeatedly at intervals.

✅ Example:

```
interval(5000).pipe(
  switchMap(() => this.api.getData())
).subscribe(console.log);
```

---

99. How do you integrate third-party libraries like Chart.js, Leaflet, etc.?

Steps:

1. Install:

```
npm install chart.js
```

2. Import & Use:

```
import Chart from 'chart.js/auto';
```

---

100. How do you handle large data sets in Angular (virtual scroll, pagination)?

Virtual Scrolling: Load visible items only.
✅ Use Angular CDK:

```
<cdk-virtual-scroll-viewport itemSize="50">
  <div *cdkVirtualFor="let item of items">{{item}}</div>
</cdk-virtual-scroll-viewport>
```

Pagination: Load data page by page from API.

✅ Summary (in Simple Terms):

- `ng-content`: Slot content.
- `ng-template`: Reusable invisible templates.
- `ng-container`: Logical wrapper (no DOM).
- `trackBy`: Prevents unnecessary re-rendering.
- `APP_INITIALIZER`: Pre-load app config.
- Real-time features: WebSocket, Polling, RxJS.
- Virtual Scroll + Pagination = Large Data Handling.
