Here’s a detailed answer set for the 50 Angular interview questions you listed:

**1. What is the main purpose of Angular?**

Angular is a TypeScript-based open-source framework developed by Google for building dynamic single-page web applications (SPAs). It provides tools for developing scalable and maintainable apps with features like data binding, dependency injection, routing, and reusable components.

**2. What are directives in Angular and how many types of directives exist?**

Directives are classes that add behavior to elements in Angular applications.  
**Types of directives:**

* **Component**: A directive with a template.
* **Structural directives**: Change the DOM layout (e.g., \*ngIf, \*ngFor).
* **Attribute directives**: Change the appearance or behavior of an element (e.g., ngClass, ngStyle).

**3. Explain data bindings and the different types.**

Data binding connects component logic with the view.  
**Types:**

* **Interpolation**: {{ value }}
* **Property binding**: [property]="value"
* **Event binding**: (event)="handlerFunction()"
* **Two-way binding**: [(ngModel)]="value"

**4. What are the basic components involved in Angular?**

* **Modules** (@NgModule)
* **Components** (@Component)
* **Templates** (HTML)
* **Metadata**
* **Directives**
* **Services** (@Injectable)
* **Routing**

**5. What’s the difference between AngularJS and Angular?**

| **Feature** | **AngularJS** | **Angular** |
| --- | --- | --- |
| Language | JavaScript | TypeScript |
| Architecture | MVC | Component-based |
| Mobile support | Limited | Good |
| Speed | Slower | Faster with AoT |
| Dependency Injection | Limited | Built-in |

**6. What are components and modules in Angular?**

* **Components**: Core building blocks controlling views using @Component.
* **Modules**: Containers for components, directives, and services using @NgModule.

**7. What are decorators in Angular?**

Decorators are functions that add metadata to classes and properties. Examples include:

* @Component
* @NgModule
* @Injectable
* @Input / @Output

**8. What is metadata or annotations in Angular?**

Metadata provides Angular information about how to process a class. It is added using decorators. E.g., @Component({...}) defines a component’s behavior and configuration.

**9. What are templates in Angular?**

Templates define the HTML view of a component. They can include Angular directives, bindings, and expressions.

**10. What is SPA and how do you implement it in Angular?**

SPA (Single Page Application) loads a single HTML page and updates the view dynamically using JavaScript. Angular implements SPAs using the RouterModule and components to load views based on routes.

**11. Explain the importance of routing in Angular & how to implement it.**

Routing enables navigation between views.  
**Steps to implement:**

1. Import RouterModule in app module.
2. Define routes using Routes array.
3. Use <router-outlet> in the template.
4. Use routerLink for navigation.

**12. What is lazy loading in Angular?**

Lazy loading loads feature modules only when needed, improving performance and load time of large apps.

**13. How do you implement lazy loading in Angular?**

1. Create a feature module with routing.
2. Add a route using loadChildren:
3. { path: 'admin', loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule) }

**14. What is Node.js?**

Node.js is a runtime environment that executes JavaScript outside the browser, used for building scalable server-side applications.

**15. What is NPM?**

NPM (Node Package Manager) is the package manager for Node.js. It manages packages and their dependencies.

**16. Why is the node\_modules folder important?**

It stores all the installed packages and their dependencies required to run a project.

**17. What is package.json?**

It contains metadata about the project, scripts, and dependencies needed to run and build the application.

**18. What is TypeScript?**

TypeScript is a superset of JavaScript that adds static types. Angular is written in TypeScript.

**19. What is the need for Angular CLI?**

Angular CLI automates project setup, development, testing, and builds with commands like ng new, ng serve, ng generate, etc.

**20. What are services in Angular?**

Services are used for business logic and data access. They can be injected into components using Angular’s dependency injection system.

**21. When would you use content projection?**

When you want to insert external HTML or components into a component’s template using <ng-content>.

**22. Explain content projection slots in Angular.**

Named slots allow multiple ng-content areas:

<ng-content select="[header]"></ng-content>

<ng-content select="[footer]"></ng-content>

**23. Why do we need ViewChild and ViewChildren in Angular?**

They allow access to child component instances, DOM elements, or directives from the parent component.

**24. What is a template reference variable?**

A variable declared in a template using # that provides a reference to a DOM element or Angular component.

**25. Explain ContentChild and ContentChildren.**

They access projected content inside a component from a parent:

* ContentChild – single instance
* ContentChildren – list of elements

**26. Differentiate between ViewChild, ViewChildren, ContentChild, and ContentChildren.**

| **Directive** | **Access** | **Type** | **Scope** |
| --- | --- | --- | --- |
| ViewChild | One element | Element/Directive | Component's view |
| ViewChildren | Multiple | QueryList | Component's view |
| ContentChild | One | Element/Directive | Projected content |
| ContentChildren | Multiple | QueryList | Projected content |

**27. What is { static: true } in ViewChild?**

Determines when the query is resolved:

* true: before change detection
* false: after view initialization

**28. What is the importance of Angular component hooks/life cycles?**

They allow developers to tap into key moments of a component’s creation, rendering, and destruction.

**29. Explain Angular lifecycle hooks in detail.**

* ngOnInit() – called once after component initialized
* ngOnChanges() – called on input-bound property changes
* ngDoCheck() – custom change detection
* ngAfterViewInit() – after component’s view initialized
* ngAfterContentInit() – after projected content initialized
* ngOnDestroy() – cleanup before component destroyed

**30. Differentiate between constructor and ngOnInit().**

* **Constructor**: Initializes class members; runs before Angular sets inputs.
* **ngOnInit()**: Used for component initialization logic, runs after inputs are set.

**31. How do you implement lazy loading in Angular?**

Same as Q13 above.

**32. How do you implement HTTP in Angular?**

1. Import HttpClientModule
2. Inject HttpClient in service
3. Use methods like get(), post(), etc.
4. this.http.get('api/url').subscribe(data => { });

**33. How do you pass data between components?**

* **Parent to child**: @Input()
* **Child to parent**: @Output() + EventEmitter
* **Between unrelated components**: Shared service with RxJS or BehaviorSubject

**34. What are pipes in Angular?**

Pipes transform data in the view. Example: {{ price | currency }}

**35. Can you give examples of inbuilt Angular pipes?**

* DatePipe
* CurrencyPipe
* UpperCasePipe
* LowerCasePipe
* DecimalPipe

**36. How do you write a custom pipe?**

@Pipe({ name: 'custom' })

export class CustomPipe implements PipeTransform {

transform(value: string): string {

return value.toUpperCase();

}

}

**37. What is RxJs and why is it needed?**

RxJS is a library for reactive programming using observables, useful for handling asynchronous operations like HTTP requests and event streams.

**38. What are observables and observers?**

* **Observable**: Emits data over time.
* **Observer**: Consumes the data via subscribe().

**39. What is a stream in RxJs?**

A stream is a sequence of ongoing data/events over time.

**40. What is the use of subscribe in RxJs?**

It initiates the execution of an observable and allows handling of emitted data.

**41. How do you unsubscribe from a stream?**

Use Subscription.unsubscribe() or takeUntil, async pipe.

**42. What are operators in RxJs?**

Operators are functions used to manipulate data streams (e.g., map, filter, mergeMap, switchMap).

**43. Where have you used RxJs in Angular?**

Common uses:

* HTTP requests
* Form input debouncing
* Component communication via services

**44. Differentiate between RxJs and Promises.**

| **Feature** | **RxJS** | **Promise** |
| --- | --- | --- |
| Emits | Multiple values | Single value |
| Lazy | Yes | Yes |
| Cancelable | Yes | No |
| Operators | Yes | No |

**45. How do you install RxJs?**

npm install rxjs

**46. Why is RxJs called push/reactive and not pull/imperative?**

RxJS pushes data to subscribers reactively, whereas imperative code pulls data as needed.

**47. Name some RxJs operators.**

* map
* filter
* switchMap
* mergeMap
* concatMap
* take
* debounceTime

**48. What are interceptors in Angular?**

Interceptors intercept and modify HTTP requests/responses. Useful for adding headers, logging, error handling.

**49. How do you implement interceptors?**

1. Create a service implementing HttpInterceptor
2. Register it in the providers array using HTTP\_INTERCEPTORS

{ provide: HTTP\_INTERCEPTORS, useClass: AuthInterceptor, multi: true }

**50. What are some uses of interceptors, and can we provide multi-interceptors?**

**Uses:**

* Add tokens
* Handle errors globally
* Log requests

Yes, you can provide multiple interceptors using multi: true.

Let me know if you want this in PDF or need mock interview questions based on these!