Understanding Angular 19 Directives: A Step-by-Step Guide

Foundational Topics

1. **Introduction to Angular**
   - What is Angular?
   - Key features of Angular
   - Comparison with other frameworks (e.g., React, Vue.js)
2. **Setting Up the Development Environment**
   - Installing Node.js and npm
   - Using Angular CLI (`ng` commands)
   - Creating a new Angular project
3. **Angular Architecture**
   - Components, Modules, Templates, and Metadata
   - Services and Dependency Injection
   - Angular's modular structure (NgModule)
4. **Components**
   - Creating and registering components
   - Component lifecycle hooks (e.g., `ngOnInit`, `ngOnDestroy`)
   - Data binding (interpolation, property binding, event binding, two-way binding)

Templates and Directives

- Structural directives (`*ngIf`, `*ngFor`, `*ngSwitch`)
- Attribute directives (`ngClass`, `ngStyle`)
- Custom directives

Data Binding

- One-way binding (interpolation, property binding)
- Two-way binding (`[(ngModel)]`)
- Event binding

Modules

- Root module (`AppModule`)
- Feature modules
- Shared modules

Services and Dependency Injection

- Creating services
- Injecting services into components
- Singleton services

Routing and Navigation

- Setting up routes
- Router outlet and router links
- Route parameters and query parameters
- Lazy loading

Forms

- Template-driven forms
- Reactive forms
- Form validation (built-in and custom validators)

Intermediate Topics

1. **HTTP Client**
    - Making HTTP requests (GET, POST, PUT, DELETE)
    - Handling responses and errors
    - Interceptors for modifying requests/responses
2. **Pipes**
    - Built-in pipes (`date`, `currency`, `uppercase`, etc.)
    - Custom pipes
    - Pure vs. impure pipes
3. **State Management**
    - Using services for state management
    - Introduction to NgRx (Redux pattern in Angular)
4. **Change Detection**

- How Angular detects changes
- Optimizing performance with `ChangeDetectionStrategy.OnPush`

5. Animations

- Angular animation basics
- Triggering animations on events

6 Testing

- Unit testing with Jasmine and Karma
- Testing components, services, and pipes
- End-to-end testing with Protractor or Cypress

Internationalization (i18n)

- Adding translations
- Using Angular's i18n tools

Security

- Preventing XSS (Cross-Site Scripting)
- Using Angular's built-in sanitization
- Secure API calls and authentication

Lazy Loading

- Configuring lazy-loaded modules
- Benefits of lazy loading

Dynamic Components

- Loading components dynamically
- Using `ComponentFactoryResolver`

Advanced Topics

1. **NgRx (State Management Library)**
   - Actions, Reducers, Selectors, and Effects
   - Managing complex application states
2. **Server-Side Rendering (SSR)**
   - Using Angular Universal for SSR
   - Benefits of SSR for SEO and performance
3. **Performance Optimization**
   - Ahead-of-Time (AOT) compilation
   - Tree shaking and bundle optimization
   - Lazy loading and code splitting
4. **Custom Decorators**
   - Creating custom decorators
   - Use cases for custom decorators
5. **Web Workers**
   - Offloading heavy computations to Web Workers
   - Integrating Web Workers in Angular
6. **Micro Frontends**
   - Building micro frontend architectures with Angular
   - Module Federation with Webpack

Angular Elements

- Packaging Angular components as custom elements
- Using Angular components in non-Angular applications

Advanced Routing

- Nested routes
- Route guards (`CanActivate`, `CanDeactivate`, etc.)
- Resolvers for pre-fetching data

Custom Validators

- Creating reusable form validators
- Cross-field validation

Third-Party Libraries

- Integrating libraries like RxJS, Lodash, Moment.js, etc.
- Using Angular Material for UI components

Best Practices

1. Writing clean and maintainable code
2. Following Angular style guide
3. Modularizing code effectively
4. Managing side effects with RxJS operators
5. Keeping components small and focused

Tools and Resources

1. **Angular CLI** (Command Line Interface)
2. **RxJS** (Reactive Extensions for JavaScript)
3. **Angular Material** (UI component library)
4. **Augury** (Debugging tool for Angular)
5. **StackBlitz** (Online IDE for Angular development)

Step 1: Installing Node.js and npm

Why Node.js?

Angular requires Node.js because it uses npm (Node Package Manager) to manage dependencies and tools like Angular CLI.

Steps to Install Node.js and npm

1. **Check if Node.js is Already Installed**
    - Open your terminal or command prompt.
    - Run the following commands to check if Node.js and npm are already installed:

```
node -v

npm -v
```

- If you see version numbers (e.g., `v18.17.1` for Node.js and `9.6.7` for npm), skip to Step 2. Otherwise, proceed with the installation.

Download Node.js

- Go to the official Node.js website: https://nodejs.org/ .
- Download the **LTS (Long-Term Support)** version (recommended for stability) or the **Current** version (latest features).

Install Node.js

- Run the downloaded installer.
- Follow the installation wizard:
    - Accept the license agreement.
    - Choose the default installation path.
    - Ensure that the option to install npm is checked (it's enabled by default).
- Complete the installation.

Verify Installation

- After installation, open a new terminal or command prompt and run:

```
node -v

npm -v
```

You should see the installed versions of Node.js and npm.

Step 2: Using Angular CLI (ng Commands)

What is Angular CLI?

Angular CLI (Command Line Interface) is a powerful tool that helps you create, develop, scaffold, and maintain Angular applications. It automates tasks like generating components, services, modules, and more.

Steps to Install Angular CLI

1. **Install Angular CLI Globally**
    - In your terminal or command prompt, run the following command:

```
npm install -g @angular/cli
```

- The `-g` flag installs Angular CLI globally, so you can use it in any project.

Verify Angular CLI Installation

- After installation, verify that Angular CLI is installed correctly by running:

```
ng version
```

You should see output similar to this:

```
Angular CLI: 16.2.0

Node: 18.17.1

Package Manager: npm 9.6.7

OS: win32 x64
```

Update Angular CLI (Optional)

- If you already have Angular CLI installed but want to update to the latest version, run:

```
npm install -g @angular/cli@latest
```

Step 3: Creating a New Angular Project

Steps to Create a New Angular Project

1. **Generate a New Angular Project**
    - Use the `ng new` command to create a new Angular project. Replace `my-angular-app` with your desired project name:

```
ng new my-angular-app
```

- This command will prompt you with a few questions:
    - **Would you like to add Angular routing?**
        - Type `y` (yes) if you plan to use routing in your app, or `n` (no) if not.
    - **Which stylesheet format would you like to use?**
        - Choose from options like CSS, SCSS, SASS, LESS, etc. Press Enter to select the default (CSS).

Navigate to the Project Directory

- Once the project is created, navigate into the project folder:

```
cd my-angular-app
```

Start the Development Server

- Use the following command to start the Angular development server:

```
ng serve
```

By default, the app will run on `http://localhost:4200`

Open the App in Your Browser

- Open your browser and go to `http://localhost:4200`.
- You should see the default Angular welcome page:

```
Welcome to my-angular-app!
```

Step 4: Understanding the Generated Project Structure

```
my-angular-app/

├── e2e/                # End-to-end tests (Protractor or Cypress)

├── node_modules/             # Installed dependencies

├── src/                # Source code of the application

│   ├── app/            # Main application folder

│   │   ├── app.component.ts    # Root component

│   │   ├── app.module.ts       # Root module

│   │   └── ...         # Other components, services, etc.

│   ├── assets/         # Static files (images, JSON, etc.)

│   ├── environments/   # Environment-specific configuration

│   ├── index.html      # Main HTML file

│   ├── main.ts         # Entry point of the application

│   └── styles.css      # Global styles

├── .angular-cli.json   # Angular CLI configuration (deprecated in newer versions)

├── angular.json        # Angular workspace configuration

├── package.json        # Project metadata and dependencies

├── tsconfig.json       # TypeScript configuration

└── README.md                 # Project documentation
```

Step 5: Running Common Angular CLI Commands

Here are some useful Angular CLI commands to get started:

1. **Generate Components**
    - To generate a new component:

```
ng generate component my-component
```

Shortcut:

```
ng g c my-component
```

Generate Services

- To generate a new service:

```
ng generate service my-service
```

Shortcut:

```
ng g s my-service
```

Generate Modules

- To generate a new module:

```
ng generate module my-module
```

Shortcut:

```
ng g m my-module
```

Build the Application

- To build the app for production:

```
ng build --prod
```

Run Tests

- To run unit tests:

```
ng test
```

To run end-to-end tests:

```
ng e2e
```

Troubleshooting Common Issues

1. **Error: "ng: command not found"**
   - Ensure Angular CLI is installed globally:

```
npm install -g @angular/cli
```

- Add npm's global binaries to your system PATH if necessary.

Error: "Port 4200 is already in use"

- Stop the process using port 4200:

```
npx kill-port 4200
```

Alternatively, specify a different port:

```
ng serve --port 4300
```

Slow Installation

- Use a faster npm registry like Yarn or pnpm:

```
npm install -g yarn

yarn install
```