Angular Fundamentals You Need to Learn (with Real-Time Examples)

Font: Tahoma

Angular Architecture Basics

What is Angular? Angular is a TypeScript-based open-source front-end web application framework developed by Google. It's used to build Single Page Applications (SPAs).

Real-time example: Consider an HR management system dashboard — Angular allows dynamic updates (like changing views or loading employee data) without reloading the page.

Core Building Blocks:

- **Components:** UI building blocks.
- **Templates:** HTML part of the component.
- **Modules:** Logical grouping of components/services.
- **Services:** Business logic and reusable code.

Angular CLI (Command Line Interface): Tool to scaffold and manage Angular apps.

Commands:

```
ng new hr-system
ng serve
ng generate component employee
```

Components

Creating Components:

```
ng generate component employee
```

@Component Decorator: Defines metadata like selector, template URL, style URL.

```
@Component({
  selector: 'app-employee',
  templateUrl: './employee.component.html',
  styleUrls: ['./employee.component.css']
})
```

Class and Template Binding:

```
<h2>{{ employee.name }}</h2>
<button (click)="promote()">Promote</button>
```

```
export class EmployeeComponent {
  employee = { name: 'Kiran', role: 'Developer' };
  promote() {
    this.employee.role = 'Senior Developer';
  }
}
```

Lifecycle Hooks:

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

Templates & Data Binding

Interpolation:

```
<p>{{ employee.name }}</p>
```

Property Binding:

```
<img [src]="employee.profilePicUrl">
```

Event Binding:

```
<button (click)="editEmployee()">Edit</button>
```

Two-Way Binding:

```
<input [(ngModel)]="employee.name">
```

(Requires `FormsModule`)

Directives:

- *Structural:* `*ngIf`, `*ngFor`, `*ngSwitch`
- *Attribute:* `ngClass`, `ngStyle`

```
<p *ngIf="employee.isActive">Active</p>
<ul>
  <li *ngFor="let emp of employeeList">{{ emp.name }}</li>
</ul>
```

Modules

@NgModule: Used to organize an Angular app.

```
@NgModule({
  declarations: [AppComponent, EmployeeComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
```

Feature Modules: Break large apps into functional areas (e.g., EmployeeModule, HRModule).

Lazy Loading: Load feature modules on demand for performance.

Services and Dependency Injection

Creating a Service:

```
ng generate service employee
```

```
@Injectable({ providedIn: 'root' })
export class EmployeeService {
  getEmployees() {
    return [ { name: 'John' }, { name: 'Jane' } ];
  }
}
```

Injecting in Component:

```
constructor(private empService: EmployeeService) {}
```

Real-time example: Common employee data is fetched using a service instead of duplicating code in every component.

Routing

Setup Routes:

```
const routes: Routes = [
  { path: 'employee', component: EmployeeComponent },
  { path: '', redirectTo: 'home', pathMatch: 'full' }
];
```

Router Links:

```
<a routerLink="/employee">Employee</a>
```

Route Parameters:

```
{ path: 'employee/:id', component: EmployeeDetailComponent }
```

Route Guards: Used to prevent unauthorized access.

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

Forms

Template-driven Forms:

```
<form #empForm="ngForm" (ngSubmit)="onSubmit(empForm)">
  <input name="name" ngModel required>
</form>
```

Reactive Forms:

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

Validation:

```
<input [formControl]="empForm.controls['name']">
<p *ngIf="!empForm.controls['name'].valid">Name required</p>
```

HTTP Client

Import HttpClientModule:

```
import { HttpClientModule } from '@angular/common/http';
```

Service Example:

```
constructor(private http: HttpClient) {}

getEmployees() {
  return this.http.get<Employee[]>('/api/employees');
}
```

Using async pipe:

```
<ul>
  <li *ngFor="let emp of employees$ | async">{{ emp.name }}</li>
</ul>
```

Pipes

Built-in Pipes:

```
<p>{{ emp.name | uppercase }}</p>
<p>{{ emp.salary | currency:'INR' }}</p>
```

Custom Pipe:

```
@Pipe({ name: 'shorten' })
export class ShortenPipe implements PipeTransform {
  transform(value: string, limit: number = 10): string {
    return value.length > limit ? value.substr(0, limit) + '...' : value;
  }
}
```

Angular CLI & Project Structure

Generate Items:

```
ng generate component login
ng generate service auth
```

Angular Structure:

- `app/` → main components & modules
- `assets/` → images, styles
- `environments/` → config per environment

Build & Deploy:

```
ng build --prod
```

Basic State Management

Component State:

```
selectedEmployee: Employee;
```

Service Sharing:

```
@Injectable({ providedIn: 'root' })
export class StateService {
  selectedEmployee = new BehaviorSubject<Employee | null>(null);
}
```

Debugging and Testing (Basic)

Browser Dev Tools:

- Use `console.log` to debug values
- Inspect elements and component data

Unit Testing:

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

Recommended Learning Order

1. Angular architecture & CLI
2. Components & templates
3. Data binding & directives
4. Modules & services
5. Routing basics
6. Forms (template-driven then reactive)
7. HTTP client & observables
8. Pipes
9. Bonus: Lazy loading, guards, custom directives/pipes, testing