Prerequisites

HTML/CSS

A strong grasp of HTML for structuring content and CSS for styling is essential before diving into Angular. Angular templates are written in HTML with Angular directives.

JavaScript ES6+

Modern Angular heavily uses ES6+ features like arrow functions, classes, modules, destructuring, and async/await.

TypeScript

Angular is built with TypeScript, a typed superset of JavaScript. Knowing interfaces, types, classes, and decorators will help.

Node.js & npm

Node.js and npm (Node Package Manager) are used to install Angular CLI and manage dependencies.

Phase 1: Foundation (Week 1-2)

1. Angular Architecture & Setup

Angular CLI
Angular CLI is a command-line tool to create, build, and manage Angular projects.

```
bash
CopyEdit
npm install -g @angular/cli
ng new my-angular-app
cd my-angular-app
ng serve
```

Project Structure

- `src/app/` – Main folder for your app's components, modules, services.

- `angular.json` – Project config.
- `package.json` – Dependencies.
- `main.ts` – Bootstrap entry point.

Angular Modules (`NgModule`)

Modules organize your application into cohesive blocks of functionality.

```
typescript
CopyEdit
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent], // Components, directives,
pipes
  imports: [BrowserModule],   // Other modules to import
  providers: [],              // Services
  bootstrap: [AppComponent]   // Root component
})
export class AppModule { }
```

Components

Building blocks of UI. Each has a template, styles, and logic.

```
typescript
CopyEdit
import { Component } from '@angular/core';

@Component({
  selector: 'app-example',
  templateUrl: './example.component.html',
  styleUrls: ['./example.component.css']
})
export class ExampleComponent {
  title = 'Hello Angular';
}
```

Templates & Data Binding

Templates combine HTML and Angular syntax to render dynamic UI.

2. Data Binding & Interpolation

String Interpolation
Bind component properties into HTML.

```
html
CopyEdit
<h1>{{ title }}</h1>
```

Property Binding
Bind element properties like attributes.

```
html
CopyEdit
<img [src]="imageUrl" />
```

Event Binding
Handle user events.

```
html
CopyEdit
<button (click)="onClick()">Click me</button>
```

Two-way Binding
Sync UI and component variables using `ngModel`.

```
html
CopyEdit
<input [(ngModel)]="username" />
<p>Your name is: {{ username }}</p>
```

Phase 2: Core Concepts (Week 3-4)

3. Directives

Structural Directives
Change DOM structure.

```
html
CopyEdit
```

```
<p *ngIf="isVisible">Visible Text</p>
<ul>
  <li *ngFor="let item of items">{{ item }}</li>
</ul>
```

Attribute Directives
Change element appearance.

```
html
CopyEdit
<div [ngClass]="{ 'highlight': isActive }">Content</div>
```

Custom Directives
Create your own directive for custom behavior.

```
typescript
CopyEdit
import { Directive, ElementRef, Renderer2 } from
'@angular/core';

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
  constructor(el: ElementRef, renderer: Renderer2) {
     renderer.setStyle(el.nativeElement, 'background-color',
'yellow');
  }
}
```

Usage:

```
html
CopyEdit
<p appHighlight>Highlighted text</p>
```

4. Component Communication

@Input()
Pass data from parent to child.

```
typescript
CopyEdit
@Component({
  selector: 'child',
  template: `Child message: {{ message }}`
})
export class ChildComponent {
  @Input() message!: string;
}
```

Parent template:

```
html
CopyEdit
<child [message]="'Hello from Parent'"></child>
```

@Output() and EventEmitter

Send events from child to parent.

```
typescript
CopyEdit
@Component({
  selector: 'child',
  template: `<button (click)="notifyParent()">Notify</button>`
})
export class ChildComponent {
  @Output() notify = new EventEmitter<string>();

  notifyParent() {
     this.notify.emit('Child says hi!');
  }
}
```

Parent template:

```
html
CopyEdit
<child (notify)="onNotify($event)"></child>
```

ViewChild & ViewChildren

Access child components or DOM elements inside a parent.

```
typescript
CopyEdit
@ViewChild(ChildComponent) child!: ChildComponent;

ngAfterViewInit() {
  console.log(this.child.message);
}
```

Template Reference Variables

Access DOM elements or components in template.

```
html
CopyEdit
<input #inputRef />
<button (click)="log(inputRef.value)">Log</button>
```

5. Services & Dependency Injection

Creating a Service

```
typescript
CopyEdit
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class DataService {
  getData() {
     return ['Item 1', 'Item 2'];
  }
}
```

Injecting Service

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

ngOnInit() {
  console.log(this.dataService.getData());
```

```
}
```

Service Hierarchy
Services provided at root are singleton. Services provided in a module/component are scoped.

Phase 3: Advanced Features (Week 5-6)

6. Routing & Navigation

Setting up routing

```
typescript
CopyEdit
import { RouterModule, Routes } from '@angular/router';
const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'about', component: AboutComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {}
```

Template:

```
html
CopyEdit
<nav>
  <a routerLink="/">Home</a>
  <a routerLink="/about">About</a>
</nav>
<router-outlet></router-outlet>
```

Route Parameters

```
typescript
CopyEdit
```

```
this.route.params.subscribe(params => {
  console.log(params['id']);
});
```

Route Guards

Implement `CanActivate` guard to control access.

7. Forms

Template-driven Forms

```
html
CopyEdit
<form #form="ngForm" (ngSubmit)="submitForm(form)">
  <input name="username" ngModel required />
  <button type="submit">Submit</button>
</form>
```

Reactive Forms

```
typescript
CopyEdit
form = new FormGroup({
  username: new FormControl('', Validators.required)
});

submitForm() {
  console.log(this.form.value);
}
```

8. HTTP Client

Setup

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

@NgModule({
```

```
  imports: [HttpClientModule]
})
export class AppModule { }
```

Service Example

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

getUsers() {
  return
this.http.get<User[]>('https://api.example.com/users');
}
```

Error Handling

```
typescript
CopyEdit
this.http.get(url).pipe(
  catchError(error => {
     console.error('Error:', error);
     return throwError(error);
  })
).subscribe();
```

Phase 4: Advanced Topics (Week 7-8)

9. RxJS & Observables

Basic Subscription

```
typescript
CopyEdit
this.dataService.getData().subscribe(data => {
  console.log(data);
});
```

Operators

```
typescript
CopyEdit
import { map, filter } from 'rxjs/operators';

this.dataService.getData().pipe(
  map(items => items.filter(item => item.active))
).subscribe();
```

Subjects

```
typescript
CopyEdit
const subject = new BehaviorSubject(0);
subject.next(1);
subject.subscribe(value => console.log(value));
```

10. Pipes

Built-in

```
html
CopyEdit
<p>{{ today | date:'shortDate' }}</p>
```

Custom Pipe

```
typescript
CopyEdit
import { Pipe, PipeTransform } from '@angular/core';

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

Usage:

```
html
CopyEdit
```

```
<p>{{ 'Angular' | reverse }}</p> <!-- ralugnA -->
```

11. Lifecycle Hooks

Hook	Purpose
ıgOnInit	nitialization after inputs set
ıgOnChanges	Detect input property changes
ıgAfterViewIn ıt	/iew and child components loaded
ıgOnDestroy	Cleanup, unsubscribe from streams

Example:

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

Phase 5: Architecture & Best Practices (Week 9-10)

12. Module Organization

- **Core Module**: Singleton services loaded once.
- **Shared Module**: Common components, pipes, directives.
- **Feature Modules**: Group related features.
- **Lazy Loading**: Load feature modules on demand.

13. State Management

- Component-local state stored in variables.
- Shared state via services.
- For complex apps, consider **NgRx**.

14. Testing

- Use Jasmine & Karma for unit testing.
- Configure TestBed to test components and services.
- E2E tests via Protractor or Cypress.

Phase 6: Production & Optimization (Week 11-12)

15. Performance Optimization

- Use `ChangeDetectionStrategy.OnPush` to reduce unnecessary change detection.
- Use `trackBy` with `*ngFor`.
- Lazy load modules and images.

16. Build & Deployment

```
bash
CopyEdit
ng build --prod
```

Manage environment-specific configurations and deploy to hosting platforms like Netlify, Firebase, or your own server.