



One framework. Mobile and desktop

A black and white photograph of a man with dark hair and glasses, wearing a dark jacket over a light-colored shirt. He is looking slightly to his left. The background is a plain, light-colored wall.

AKASH

CTO and Co-founder, GetSetGo Fitness.

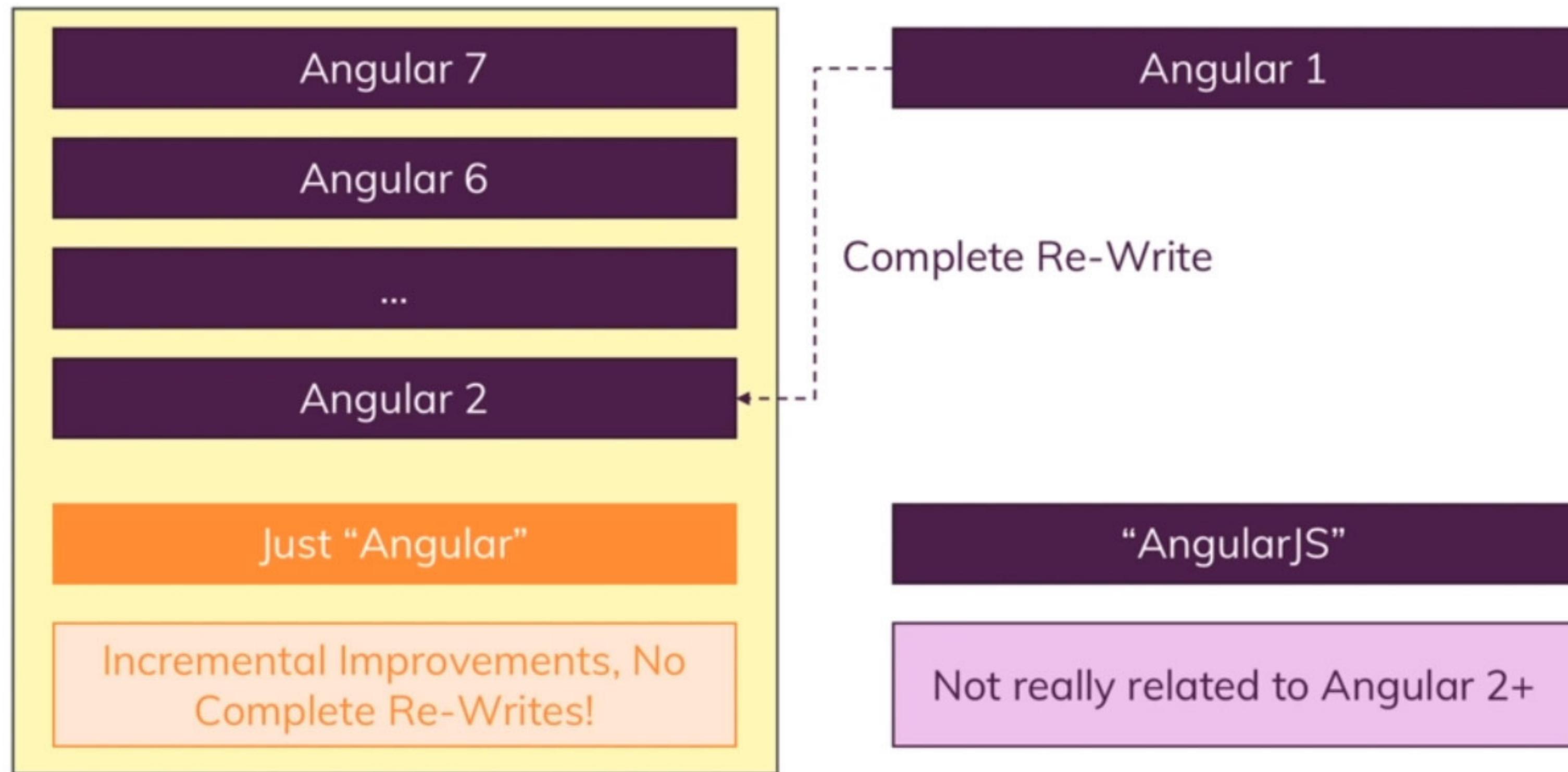
At my core, I am still a software architect who likes to write 'good code'



**SO,
WHAT IS ANGULAR ANYWAY?**

MV-VM FRAMEWORK?

Angular 7 vs Angular 6 vs Angular 2 vs Angular 1



LET US START AN APP REAL QUICK

<https://angular.io/guide/quickstart>

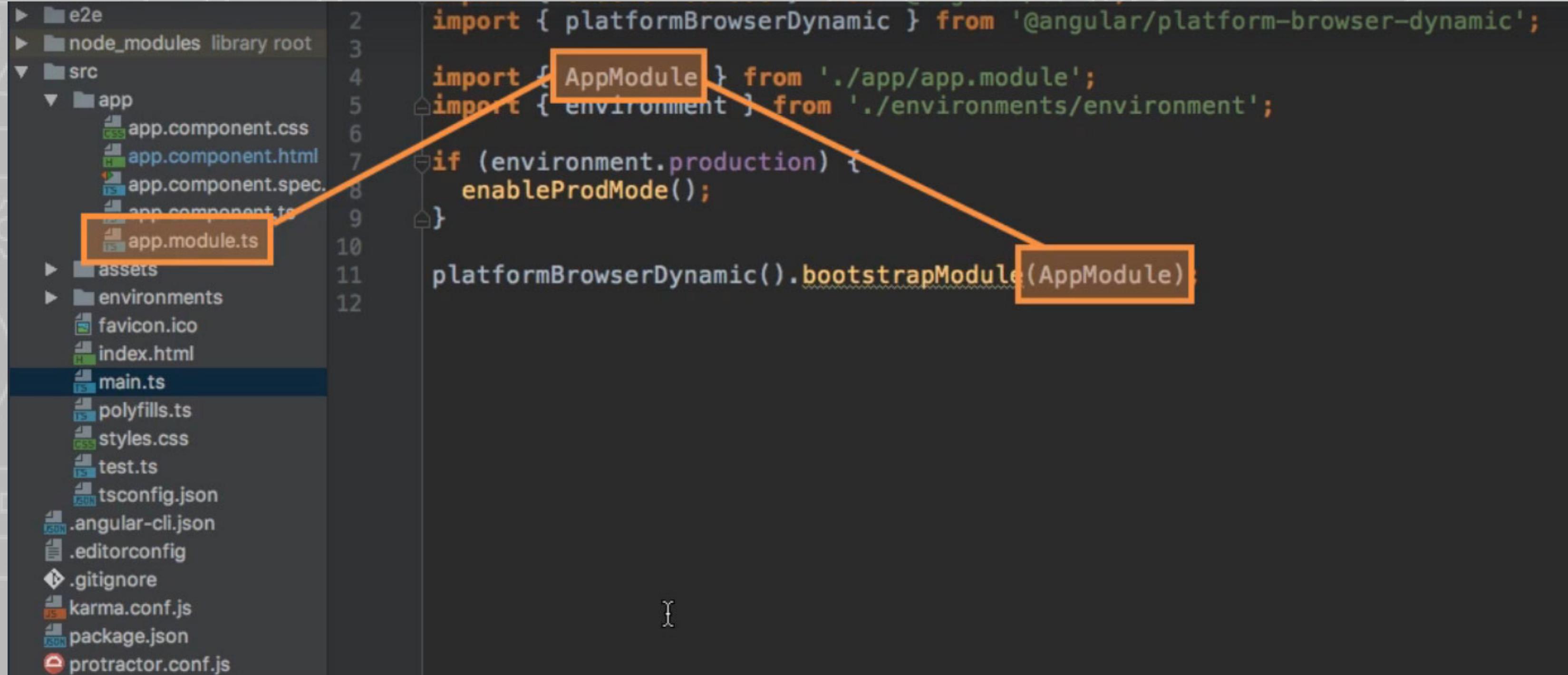
SOME MAGIC - NGMODEL

Let the code do the talking ...

ANGULAR MATERIAL - THE PRESENTATION LIBRARY

<https://material.angular.io/>

A WORD ON SPA AND A PEEK BEHIND THE SCENE



```
2 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4 import { AppModule } from './app/app.module';
5 import { environment } from './environments/environment';
6
7 if (environment.production) {
8   enableProdMode();
9 }
10
11 platformBrowserDynamic().bootstrapModule(AppModule);
```

The screenshot shows a code editor with the following file structure:

- e2e
- node_modules library root
- src
 - app
 - app.component.css
 - app.component.html
 - app.component.spec.ts
 - app.component.ts
 - app.module.ts
 - assets
 - environments
 - favicon.ico
 - index.html
 - main.ts
 - polyfills.ts
 - styles.css
 - test.ts
 - tsconfig.json
 - .angular-cli.json
 - .editorconfig
 - .gitignore
 - karma.conf.js
 - package.json
 - protractor.conf.js

LET'S TALK COMPONENTS



ENOUGH TALKING ... LET'S ADD ONE

Err ... one small step before we move ahead - let's discuss some JS and ES6 ...

... Okay, now, the component

... And now for something cool - create components using CLI

... and while we're at it, let's look at template (over templateUrl) with template strings

... oh, and some styling is in order, right? A discussion on styles and stylesUrl

Component selectors - attribute selectors, class selectors

LET'S MAKE THINGS INTERESTING WITH DATABINDING

TypeScript Code
(Business Logic)

Databinding = Communication

Output Data

String Interpolation ({{ data }})

Property Binding ([property]="data")

React to (User) Events

Event Binding ((event)="expression")

Template (HTML)

Combination of Both: **Two-Way-Binding** ([(ngModel)]="data")

DIRECTIVES - MAKE THINGS HAPPEN

Directives are Instructions in the DOM!

```
<p appTurnGreen>Receives a green background!</p>
```

```
@Directive({  
    selector: '[appTurnGreen]'  
})  
export class TurnGreenDirective {  
    ...  
}
```

TYPES OF DIRECTIVES

Structural directives - ngIf, ngFor
attribute directives - ngStyle, ngClass

Getting index in ngFor

LET'S CATCH THEM WHEN THEY'RE YOUNG - ERRORS

Understanding the error messages

Debugging code in browser using sourcemaps

Using Augury to dive into Angular apps

PROPERTY AND EVENT BINDING

HTML Elements

Directives

Components

Native Properties &
Events

Custom Properties &
Events

Custom Properties &
Events

COMPONENTS AND PROPERTIES - DEEP DIVE

Splitting an app into components

Binding to custom properties

Assigning an alias to custom properties

Binding to custom events

Assigning an alias to custom events

Understanding view encapsulation

Using local references in templates

Projecting contents in components using
ng-content

Explore ContentChild

COMPONENT LIFECYCLE

ngOnChanges

Called after a bound input property changes

ngOnInit

Called once the component is initialized

ngDoCheck

Called during every change detection run

ngAfterContentInit

Called after content (ng-content) has been projected into view

ngAfterContentChecked

Called every time the projected content has been checked

ngAfterViewInit

Called after the component's view (and child views) has been initialized

ngAfterViewChecked

Called every time the view (and child views) have been checked

ngOnDestroy

Called once the component is about to be destroyed

DIRECTIVES - DEEP DIVE

A basic directive

Improving with renderer

Using the host listener to bind events

Using the host binding to bind properties

Binding to directive properties with @Input

Behind the scenes of structural directives

Let's build a structural directive

ngSwitch

STRUCTURAL DIRECTIVE - BEHIND THE SCENE

```
[ngClass]="{odd: odd % 2 !== 0}"
[ngStyle]="{backgroundColor: odd % 2 !== 0 ? 'yellow' : 'transparent'}"
*ngFor="let odd of oddNumbers">
{{ odd }}
</li>
</div>
<div *ngIf="!onlyOdd">
<li
  class="list-group-item"
  [ngClass]="{even: even % 2 === 0}"
  [ngStyle]="{backgroundColor: even % 2 === 0 ? 'yellow' : 'transparent'}"
*ngFor="let even of evenNumbers">
{{ even }}
</li>
</div>
<ng-template [ngIf]="!onlyOdd">
<div>
<li
  class="list-group-item"
  [ngClass]="{odd: even % 2 !== 0}"
  [ngStyle]="{backgroundColor: even % 2 !== 0 ? 'yellow' : 'transparent'}"
*ngFor="let even of evenNumbers">
{{ even }}
</li>
</div>
```

S E R V I C E S

Why do we need them?

Creating a logging service

Injecting a service in a component

Data services

Hierarchical injection of services

Inject service in module

Inject service in service

Cross component communication with services

HIERARCHICAL SERVICES

AppModule

Same Instance of Service is available **Application-wide**

AppComponent

Same Instance of Service is available for **all Components (but not for other Services)**

Any other Component

Same Instance of Service is available for **the Component and all its child components**

ROUTER

Setting up and loading routes

Navigating with router links

Understanding links - absolute vs relative path

Active class for active route

Navigating programmatically

Navigating programmatically - Issue in relative path

Adding parameters to routes

Fetching routes reactively - caching issues and observables

Query parameters and fragments

Nesting of routes

ROUTER CONT'D...

Redirection and wildcard routes

Protecting routes with canActivate

Protecting child with canActivateChild

Using a fake auth service

FORMS

Reactive vs template-driven

Form-control

Form group

setvalue

Nesting form group

setValue vs patchValue

Form-builder

Form validation

Form status