ANGULAR TRAINING

By Jeroen Burgers

CINQ.

TIMETABLE DAY 1

{

09:30 | Welcome
#0 | About me and you
#0 | Introduction
#1 | What/why is Angular?
#2 | Create your first app
#3 | Use modules and components
#4 | Navigate to other pages
#5 | Writing services and fetching data
#6 | Transform data with pipes
16:00 | Recap and closing

}

ABOUT ME JEROEN BURGERS

Work

- Senior front-end developer (started in 2009) and consultant
- Working for DHL (employed by CINQ ICT)
- Worked for companies such as Politie (Dutch Police), Univé, Qbuzz, Suzuki, Aegon, KBC Bank, and ING/NN Investments
- Specialized in: JavaScript, ES6, Angular, Typescript, React, and more

Personal

- Live in Almere (The Netherlands)
- Married to Daniëlle
- Father of Sem and Elise
- Passion for soccer (especially Ajax) and running

ABOUT YOU

Who are you?

- Personal
- Work (experience)
- Skills
- Company

About the training

- What do you want to learn?

INTRODUCTION

Day 1

- What and why is Angular?
- Create your first app
- Use modules and components
- Navigate to other pages
- Writing service and fetching data
- Transform data with pipes

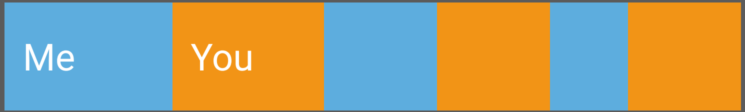
Day 2

- Build a form
- Directives
- Testing and publish your app (in practice)

Day 3

- …same as day 1 and 2
- Build your own todo app
- What do you want to learn more?

Way of working



#1 What/why is Angular?

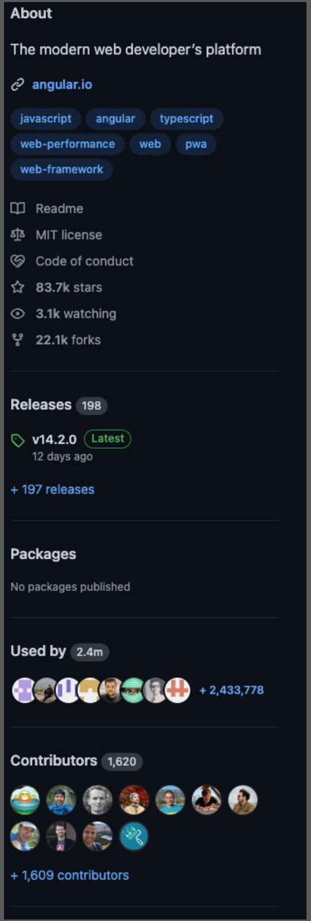
#1 What is Angular?

- The modern web developer's platform
- A platform that develops with Typescript
- Developed by Google
- The first release in 2010
- First named AngularJS and Angular2 since 2016 (formerly known as Angular)

- Essential for Angular:
  - Components
  - Templates
  - Dependency injection

#1 Why Angular?

- Gives you a toolset to develop and learn
  - A lot of build-in functionality (services, pipes, etc)
- Has a well-maintained codebase (included updates)
  - Current version: 14
  - Release management: https://angular.io/guide/releases#support-policy-and-schedule
- Each project has the “same” structure
  - Styleguide: https://angular.io/guide/styleguide
- Angular CLI helps developers to implement
  - See: https://angular.io/cli


About

The modern web developer's platform

angular.io

javascript angular typescript web-performance web pwa web-framework

Readme

MIT license

Code of conduct

83.7k stars

3.1k watching

22.1k forks

Releases 198

v14.2.0 Latest

12 days ago

+ 197 releases

Packages

No packages published

Used by 2.4m

+ 2,433,778

Contributors 1,620

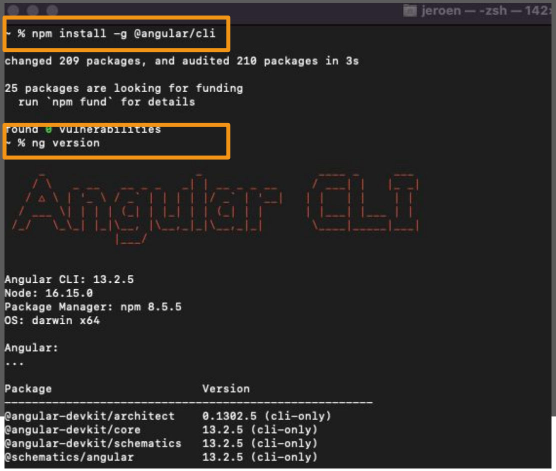
+ 1,609 contributors

#2 Create your first app

#2 Create your first app

1. Install nodejs/npm with: https://nodejs.org/en/
    - Or check your current version in the terminal with the commands
        - npm -v
        - node -v

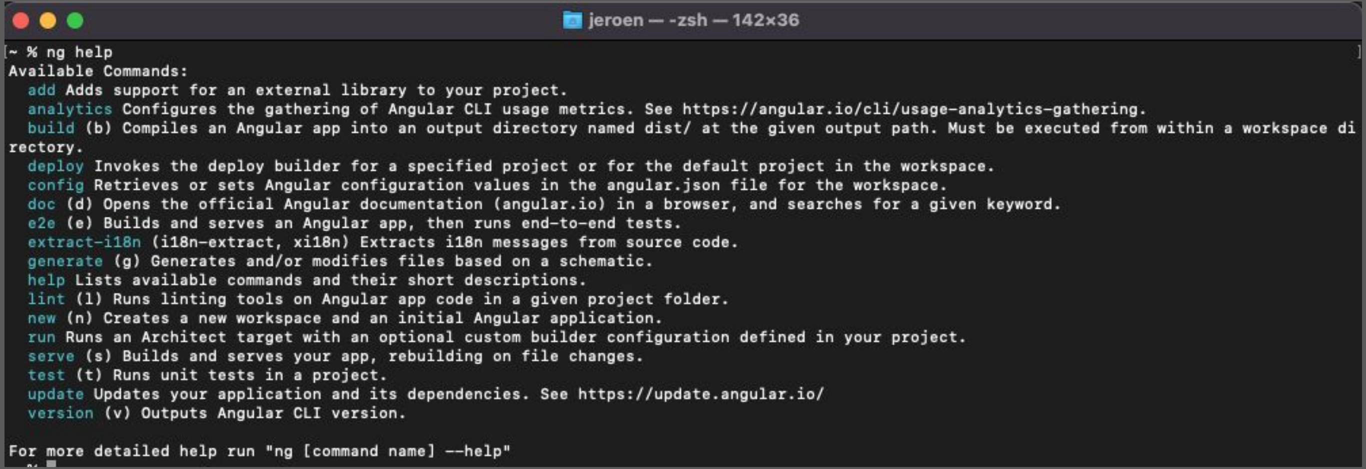
2. Open your terminal and install Angular CLI globally with: npm install -g @angular/cli



```
jeroen — -zsh — 142
~ % npm install -g @angular/cli

changed 209 packages, and audited 210 packages in 3s

25 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
~ % ng version

Angular CLI: 13.2.5
Node: 16.15.0
Package Manager: npm 8.5.5
OS: darwin x64

Angular:
...

Package                      Version
-------------------------------------------------------
@angular-devkit/architect    0.1302.5 (cli-only)
@angular-devkit/core         13.2.5 (cli-only)
@angular-devkit/schematics   13.2.5 (cli-only)
@schematics/angular          13.2.5 (cli-only)
```

#2 What is Angular CLI?

- Command-line interface tool that you use to initialize, develop, scaffold, and maintain

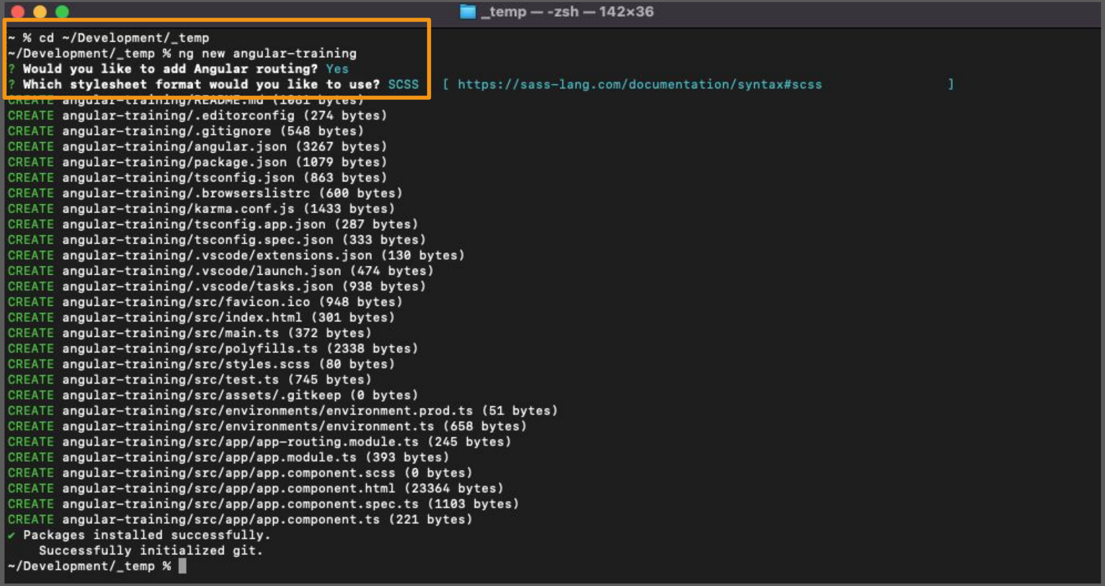


```
~ % ng help
Available Commands:
  add Adds support for an external library to your project.
  analytics Configures the gathering of Angular CLI usage metrics. See https://angular.io/cli/usage-analytics-gathering.
  build (b) Compiles an Angular app into an output directory named dist/ at the given output path. Must be executed from within a workspace di
rectory.
  deploy Invokes the deploy builder for a specified project or for the default project in the workspace.
  config Retrieves or sets Angular configuration values in the angular.json file for the workspace.
  doc (d) Opens the official Angular documentation (angular.io) in a browser, and searches for a given keyword.
  e2e (e) Builds and serves an Angular app, then runs end-to-end tests.
  extract-i18n (i18n-extract, xi18n) Extracts i18n messages from source code.
  generate (g) Generates and/or modifies files based on a schematic.
  help Lists available commands and their short descriptions.
  lint (l) Runs linting tools on Angular app code in a given project folder.
  new (n) Creates a new workspace and an initial Angular application.
  run Runs an Architect target with an optional custom builder configuration defined in your project.
  serve (s) Builds and serves your app, rebuilding on file changes.
  test (t) Runs unit tests in a project.
  update Updates your application and its dependencies. See https://update.angular.io/
  version (v) Outputs Angular CLI version.

For more detailed help run "ng [command name] --help"
```

#2 Create your app with Angular CLI

1. Go to a folder in your terminal where you want to install your app:
   cd ~/[PATH_TO_FOLDER]

2. Install your app:
   ng new angular-training



```
~ % cd ~/Development/_temp
~/Development/_temp % ng new angular-training
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS   [ https://sass-lang.com/documentation/syntax#scss              ]
CREATE angular-training/README.md (1061 bytes)
CREATE angular-training/.editorconfig (274 bytes)
CREATE angular-training/.gitignore (548 bytes)
CREATE angular-training/angular.json (3267 bytes)
CREATE angular-training/package.json (1079 bytes)
CREATE angular-training/tsconfig.json (863 bytes)
CREATE angular-training/.browserslistrc (600 bytes)
CREATE angular-training/karma.conf.js (1433 bytes)
CREATE angular-training/tsconfig.app.json (287 bytes)
CREATE angular-training/tsconfig.spec.json (333 bytes)
CREATE angular-training/.vscode/extensions.json (130 bytes)
CREATE angular-training/.vscode/launch.json (474 bytes)
CREATE angular-training/.vscode/tasks.json (938 bytes)
CREATE angular-training/src/favicon.ico (948 bytes)
CREATE angular-training/src/index.html (301 bytes)
CREATE angular-training/src/main.ts (372 bytes)
CREATE angular-training/src/polyfills.ts (2338 bytes)
CREATE angular-training/src/styles.scss (80 bytes)
CREATE angular-training/src/test.ts (745 bytes)
CREATE angular-training/src/assets/.gitkeep (0 bytes)
CREATE angular-training/src/environments/environment.prod.ts (51 bytes)
CREATE angular-training/src/environments/environment.ts (658 bytes)
CREATE angular-training/src/app/app-routing.module.ts (245 bytes)
CREATE angular-training/src/app/app.module.ts (393 bytes)
CREATE angular-training/src/app/app.component.scss (0 bytes)
CREATE angular-training/src/app/app.component.html (23364 bytes)
CREATE angular-training/src/app/app.component.spec.ts (1103 bytes)
CREATE angular-training/src/app/app.component.ts (221 bytes)
✔ Packages installed successfully.
    Successfully initialized git.
~/Development/_temp % 
```

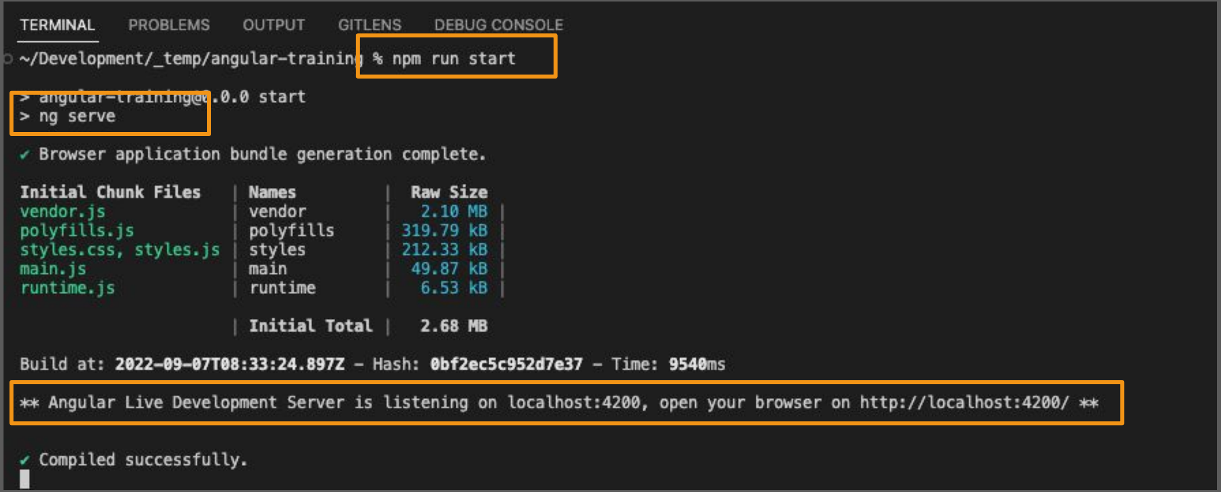
#2 Go to your app

1. Go to the folder of your app:
   cd angular-training

2. Run your app:
   ng serve or npm run start

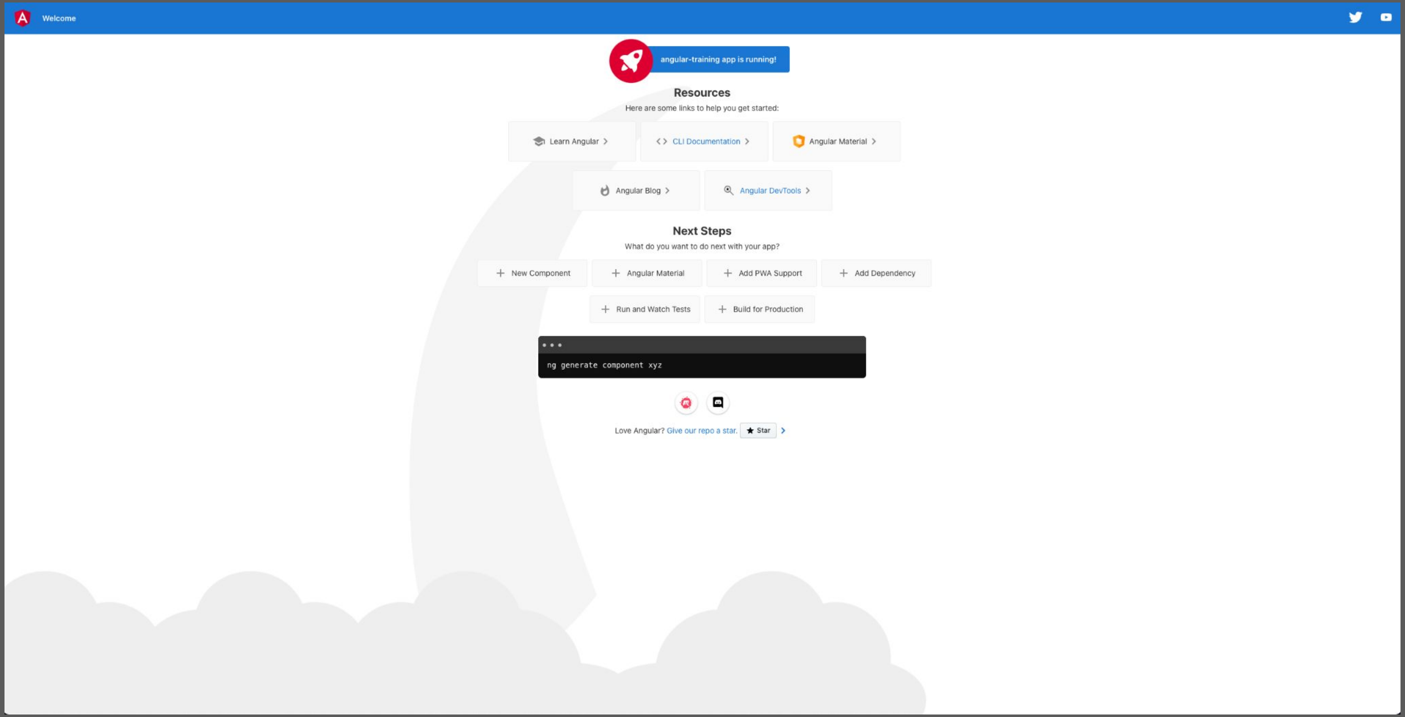
3. Go to http://localhost:4200
   in your browser

4. Open your code editor



```
~/Development/_temp/angular-training % npm run start

> angular-training@0.0.0 start
> ng serve

✔ Browser application bundle generation complete.

Initial Chunk Files          | Names          |  Raw Size
vendor.js                    | vendor         |   2.10 MB |
polyfills.js                 | polyfills      | 319.79 kB |
styles.css, styles.js        | styles         | 212.33 kB |
main.js                      | main           |  49.87 kB |
runtime.js                   | runtime        |   6.53 kB |

                             | Initial Total  |   2.68 MB

Build at: 2022-09-07T08:33:24.897Z - Hash: 0bf2ec5c952d7e37 - Time: 9540ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✔ Compiled successfully.
```

#2 Your first app


Welcome

angular-training app is running!

Resources

Here are some links to help you get started:

Learn Angular >
CLI Documentation >
Angular Material >
Angular Blog >
Angular DevTools >

Next Steps

What do you want to do next with your app?

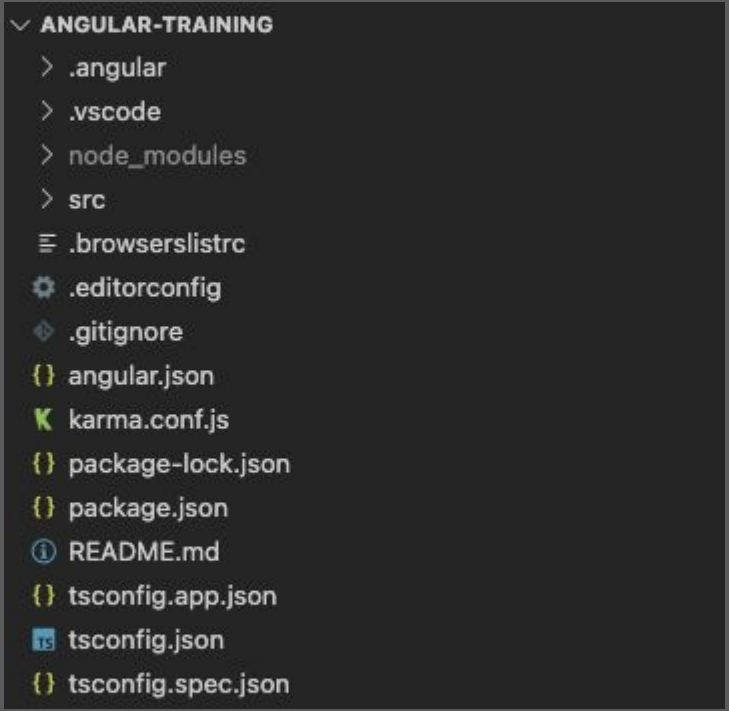
+ New Component
+ Angular Material
+ Add PWA Support
+ Add Dependency
+ Run and Watch Tests
+ Build for Production

```
ng generate component xyz
```

Love Angular? Give our repo a star. ★ Star >

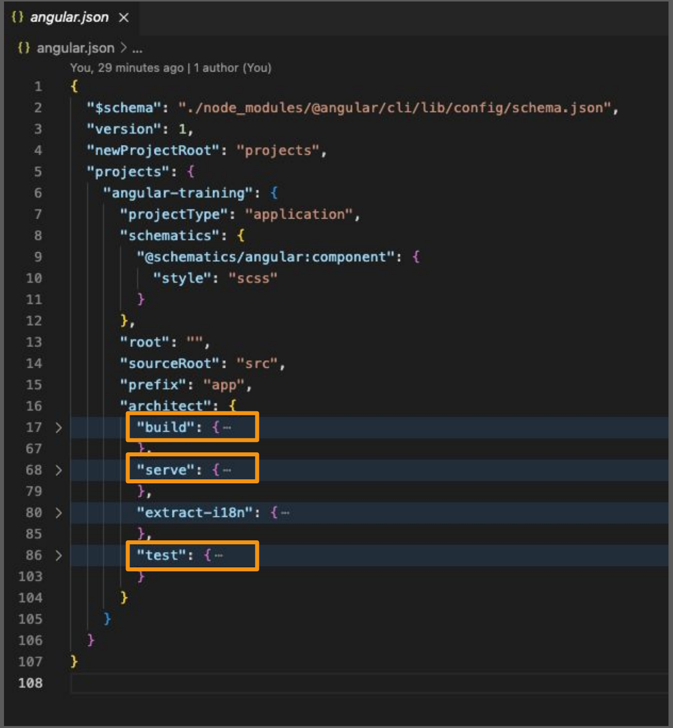
CINQ.

#2 Folder structure



```
ANGULAR-TRAINING
  > .angular
  > .vscode
  > node_modules
  > src
  .browserslistrc
  .editorconfig
  .gitignore
  angular.json
  karma.conf.js
  package-lock.json
  package.json
  README.md
  tsconfig.app.json
  tsconfig.json
  tsconfig.spec.json
```

#2 Folder structure

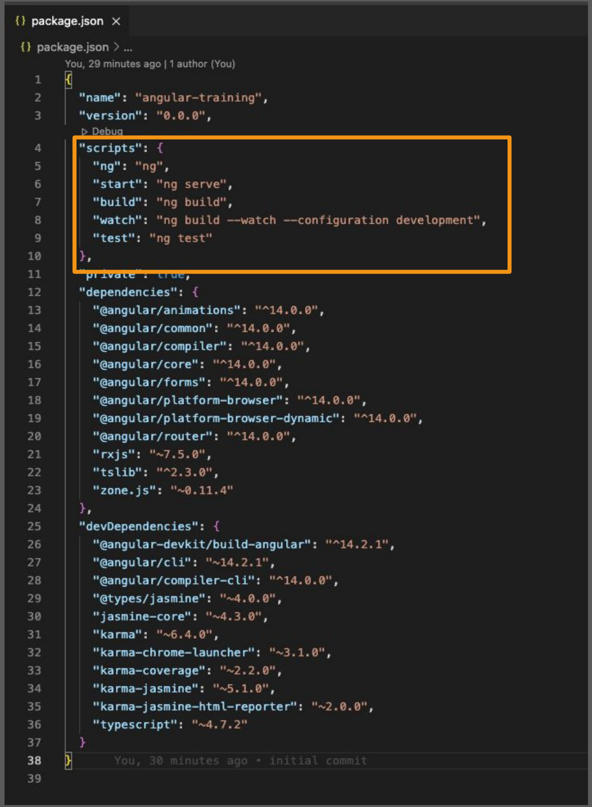


angular.json ×

angular.json > ...

You, 29 minutes ago | 1 author (You)

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "angular-training": {
      "projectType": "application",
      "schematics": {
        "@schematics/angular:component": {
          "style": "scss"
        }
      },
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {…
        },
        "serve": {…
        },
        "extract-i18n": {…
        },
        "test": {…
        }
      }
    }
  }
}
```



package.json ×

package.json > ...

You, 29 minutes ago | 1 author (You)

```
{
  "name": "angular-training",
  "version": "0.0.0",
  ▷ Debug
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^14.0.0",
    "@angular/common": "^14.0.0",
    "@angular/compiler": "^14.0.0",
    "@angular/core": "^14.0.0",
    "@angular/forms": "^14.0.0",
    "@angular/platform-browser": "^14.0.0",
    "@angular/platform-browser-dynamic": "^14.0.0",
    "@angular/router": "^14.0.0",
    "rxjs": "~7.5.0",
    "tslib": "^2.3.0",
    "zone.js": "~0.11.4"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^14.2.1",
    "@angular/cli": "~14.2.1",
    "@angular/compiler-cli": "^14.0.0",
    "@types/jasmine": "~4.0.0",
    "jasmine-core": "~4.3.0",
    "karma": "~6.4.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage": "~2.2.0",
    "karma-jasmine": "~5.1.0",
    "karma-jasmine-html-reporter": "~2.0.0",
    "typescript": "~4.7.2"
  }
}      You, 30 minutes ago • initial commit
```

CINQ.

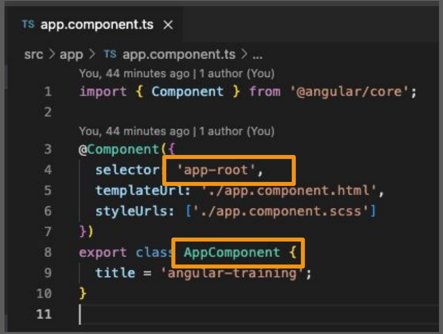
#2 Understanding file structure

- app/app.component.ts
  The root of the application

- app/app.module.ts
  The entry of our Angular application

- index.html
  The page where your app will be rendered in

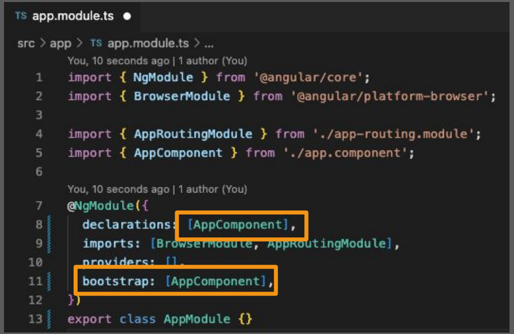
- app/main.ts
  The file where magic happened to connect component and page



app.component.ts

src > app > app.component.ts > ...

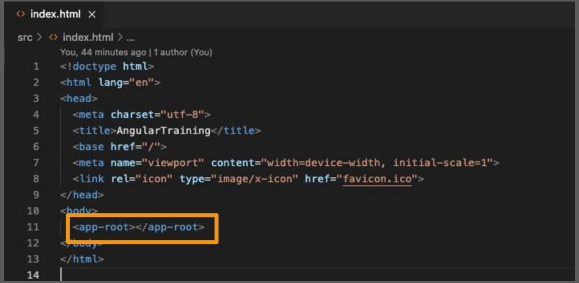
```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'angular-training';
}
```



app.module.ts

src > app > app.module.ts > ...

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, AppRoutingModule],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```



index.html

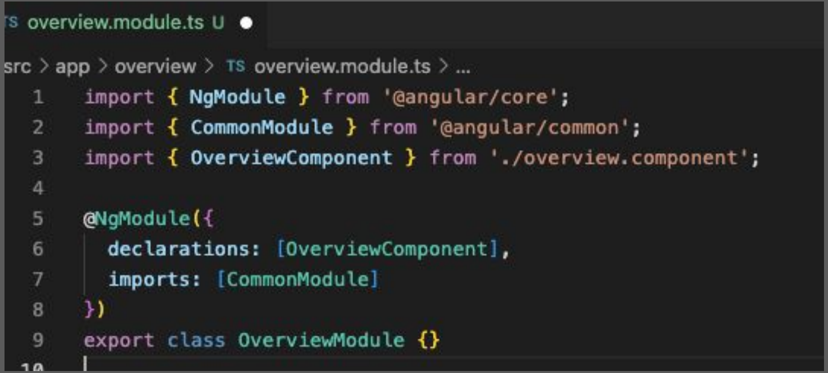
src > index.html > ...

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>AngularTraining</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

#3 Use modules and components

#3 What is a module?

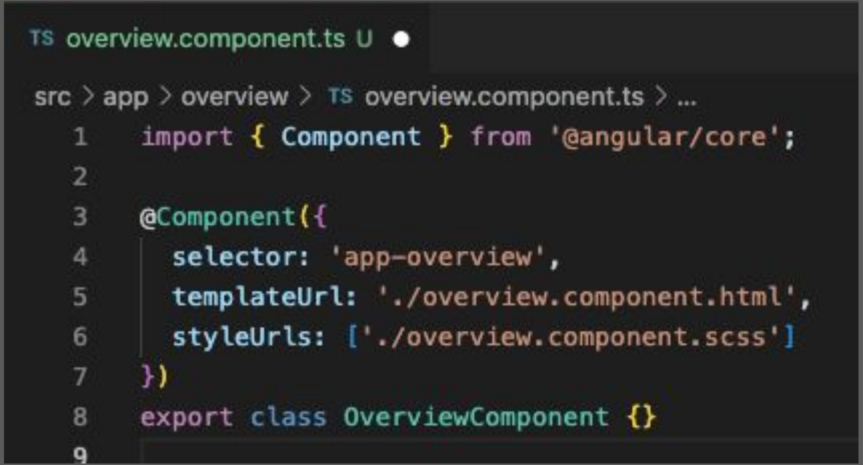
- A mechanism to bundle components, directives, pipes and services
- We should use them as “NgModule” decorator, which is a Typescript feature
- Angular knows two types of modules,
  namely a root module (normally app.module.ts) and feature modules (overview.module.ts)



```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { OverviewComponent } from './overview.component';

@NgModule({
  declarations: [OverviewComponent],
  imports: [CommonModule]
})
export class OverviewModule {}
```

#3 What is a component?

- Components are the most basic UI building block of an Angular app
- An Angular app contains a tree of Angular components
- A component should be register in a module to use them

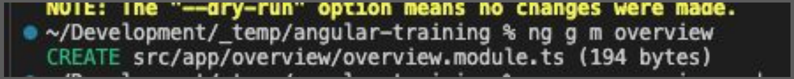

TS overview.component.ts U
src > app > overview > TS overview.component.ts > ...

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

@Component({
  selector: 'app-overview',
  templateUrl: './overview.component.html',
  styleUrls: ['./overview.component.scss']
})
export class OverviewComponent {}
```

CINQ.

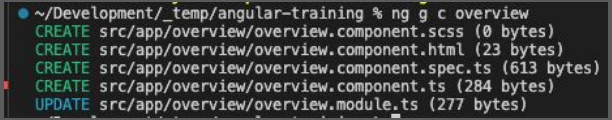
#3 Create a module and a component

1. Create a overview module:
   ng g m overview or ng generate module overview
   https://angular.io/cli/generate#module-command

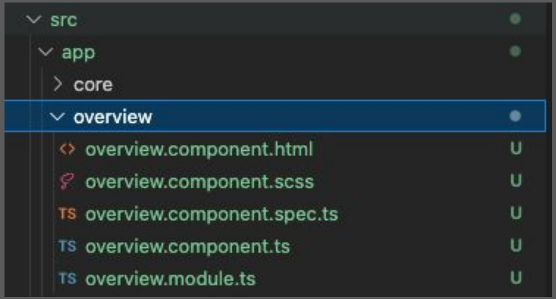


```
  NOTE: The "--dry-run" option means no changes were made.
● ~/Development/_temp/angular-training % ng g m overview
  CREATE src/app/overview/overview.module.ts (194 bytes)
```

2. Create an overview component:
   ng g c overview or ng generate component overview
   https://angular.io/cli/generate#component-command



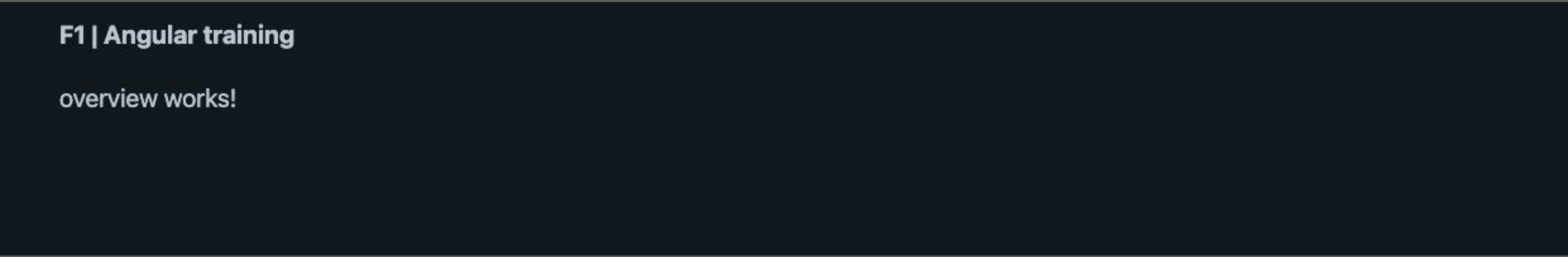
```
● ~/Development/_temp/angular-training % ng g c overview
  CREATE src/app/overview/overview.component.scss (0 bytes)
  CREATE src/app/overview/overview.component.html (23 bytes)
  CREATE src/app/overview/overview.component.spec.ts (613 bytes)
  CREATE src/app/overview/overview.component.ts (284 bytes)
  UPDATE src/app/overview/overview.module.ts (277 bytes)
```



```
∨ src
  ∨ app
    > core
    ∨ overview
        overview.component.html       U
        overview.component.scss       U
        overview.component.spec.ts    U
        overview.component.ts         U
        overview.module.ts            U
```

#3 Connect

1. Export the OverviewComponent in the overview.module.ts
2. Import (add) the OverviewModule in the app.module.ts
3. Use the selector of overview.component.ts in the app.component.html



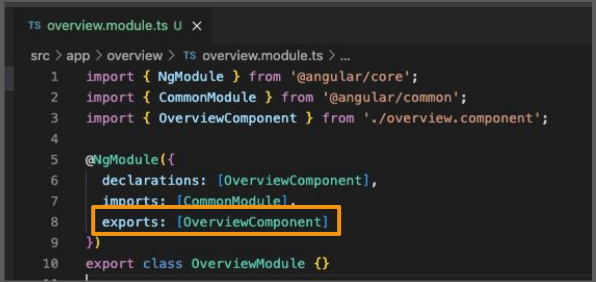
F1 | Angular training

overview works!

CINQ.

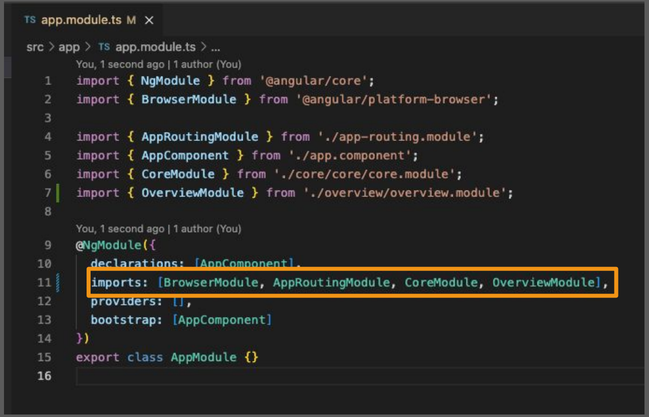
#3 Connect

1. Export the OverviewComponent in the overview.module.ts



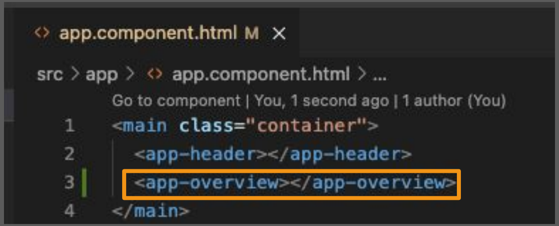
```
TS overview.module.ts U ×
src > app > overview > TS overview.module.ts > ...
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { OverviewComponent } from './overview.component';

@NgModule({
  declarations: [OverviewComponent],
  imports: [CommonModule],
  exports: [OverviewComponent]
})
export class OverviewModule {}
```

2. Import (add) the OverviewModule in the app.module.ts



```
TS app.module.ts M ×
src > app > TS app.module.ts > ...
You, 1 second ago | 1 author (You)
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { CoreModule } from './core/core/core.module';
import { OverviewModule } from './overview/overview.module';

You, 1 second ago | 1 author (You)
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, AppRoutingModule, CoreModule, OverviewModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

3. Use the selector of overview.component.ts in the app.component.html



```
<> app.component.html M ×
src > app > <> app.component.html > ...
Go to component | You, 1 second ago | 1 author (You)
<main class="container">
  <app-header></app-header>
  <app-overview></app-overview>
</main>
```

#3 What are lifecycle hooks

A component has a lifecycle managed by Angular itself. Angular manages creation, rendering, data-bound properties etc. It also offers hooks that allow us to respond to key lifecycle events. Here is the complete lifecycle hook interface inventory:

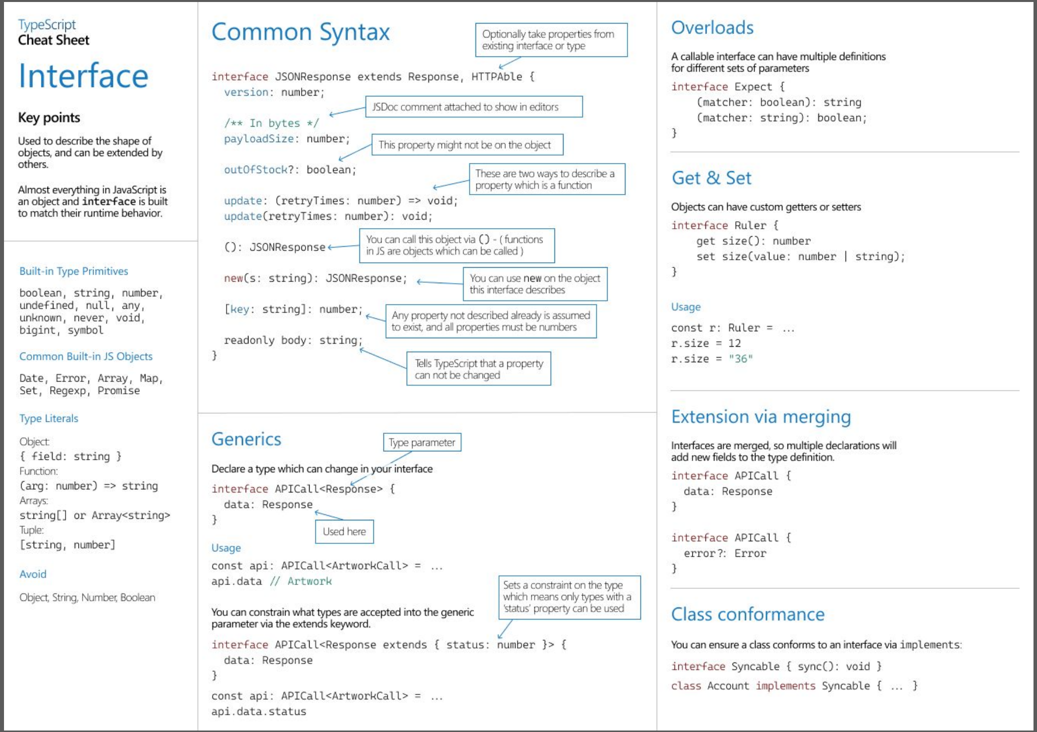
- ngOnChanges - called when an input binding value changes
- ngOnInit - after the first ngOnChanges
- ngDoCheck - after every run of change detection
- ngAfterContentInit - after component content initialized
- ngAfterContentChecked - after every check of component content
- ngAfterViewInit - after component's view(s) are initialized
- ngAfterViewChecked - after every check of a component's view(s)
- ngOnDestroy - just before the component is destroyed

#3 What are interfaces?

Interface is a structure that defines the contract in your application. It defines the syntax for classes to follow. Classes that are derived from an interface must follow the structure provided by their interface.

The TypeScript compiler does not convert interface to JavaScript. It uses interface for type checking.

Read more…


TypeScript
Cheat Sheet

Interface

Key points

Used to describe the shape of objects, and can be extended by others.

Almost everything in JavaScript is an object and `interface` is built to match their runtime behavior.

Built-in Type Primitives

```
boolean, string, number,
undefined, null, any,
unknown, never, void,
bigint, symbol
```

Common Built-in JS Objects

```
Date, Error, Array, Map,
Set, Regexp, Promise
```

Type Literals

Object:
`{ field: string }`
Function:
`(arg: number) => string`
Arrays:
`string[] or Array<string>`
Tuple:
`[string, number]`

Avoid

Object, String, Number, Boolean

Common Syntax

```
interface JSONResponse extends Response, HTTPAble {
  version: number;

  /** In bytes */
  payloadSize: number;

  outOfStock?: boolean;

  update: (retryTimes: number) => void;
  update(retryTimes: number): void;

  (): JSONResponse

  new(s: string): JSONResponse;

  [key: string]: number;

  readonly body: string;
}
```

Optionally take properties from existing interface or type

JSDoc comment attached to show in editors

This property might not be on the object

These are two ways to describe a property which is a function

You can call this object via () - (functions in JS are objects which can be called)

You can use **new** on the object this interface describes

Any property not described already is assumed to exist, and all properties must be numbers

Tells TypeScript that a property can not be changed

Generics

Type parameter

Declare a type which can change in your interface

```
interface APICall<Response> {
  data: Response
}
```

Used here

Usage

```
const api: APICall<ArtworkCall> = ...
api.data // Artwork
```

You can constrain what types are accepted into the generic parameter via the extends keyword.

Sets a constraint on the type which means only types with a 'status' property can be used

```
interface APICall<Response extends { status: number }> {
  data: Response
}

const api: APICall<ArtworkCall> = ...
api.data.status
```

Overloads

A callable interface can have multiple definitions for different sets of parameters

```
interface Expect {
    (matcher: boolean): string
    (matcher: string): boolean;
}
```

Get & Set

Objects can have custom getters or setters

```
interface Ruler {
    get size(): number
    set size(value: number | string);
}
```

Usage

```
const r: Ruler = ...
r.size = 12
r.size = "36"
```

Extension via merging

Interfaces are merged, so multiple declarations will add new fields to the type definition.

```
interface APICall {
  data: Response
}

interface APICall {
  error?: Error
}
```

Class conformance

You can ensure a class conforms to an interface via `implements`:

```
interface Syncable { sync(): void }
class Account implements Syncable { ... }
```

CINQ.

#3 Create an interface

1. Create a “shared” folder in your src/app directory
2. Create a “races” folder in your shared directory
3. Create a races.model.ts in your races directory
4. Create an interface for a race

```
export interface Race {
 name: string;
 location: string;
 date: string;
}
```

#3 Create a property

1. Create an empty races property in the overview.module.ts with the type Race[]

2. Create/use the ngOnInit to fill your static races property with the values

   Don't forget to implement OnInit for the class property, see documentation of lifecycle hooks

```
import { Component, OnInit } from '@angular/core';
import { Race } from '../shared/races/races.model';

@Component({
 selector: 'app-overview',
 templateUrl: './overview.component.html',
 styleUrls: ['./overview.component.scss']
})
export class OverviewComponent implements OnInit {
 public races: Race[] = [];

 public ngOnInit(): void {
   this.races = [
     {
       name: 'Bahrain International Circuit',
       location: 'Bahrain',
       date: '2022-03-20'
     },
    {...},
    {...}
   ];
 }
}
```

CINQ.

#3 Create more static...

1. Create a title property

2. Create a year property

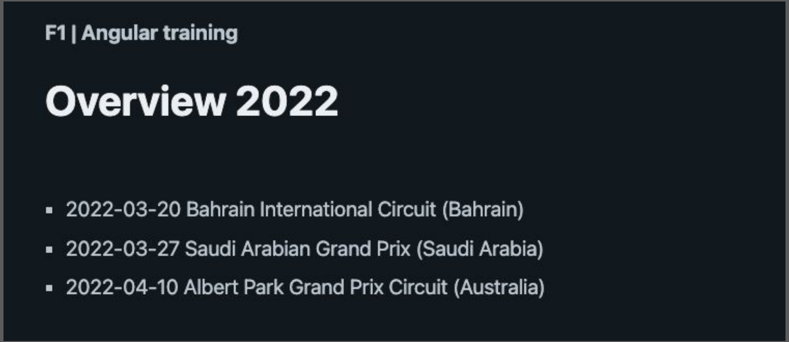
```
import { Component, OnInit } from '@angular/core';
import { Race } from '../shared/races/races.model';

@Component({
 selector: 'app-overview',
 templateUrl: './overview.component.html',
 styleUrls: ['./overview.component.scss']
})
export class OverviewComponent implements OnInit {
 public title: string = 'Overview';
 public year: number = 2022;
 public races: Race[] = [];
}
```

CINQ.

#3 Binding our data

1. Open your overview.component.html
2. Add a <h1> with the title and year property
3. Add a list of races by using a *ngFor loop

```
<h1>{{ title }} {{ year }}</h1>
<ul>
 <li *ngFor="let race of races">
    {{ race.date }} {{ race.name }}
 </li>
</ul>
```



F1 | Angular training

Overview 2022

- 2022-03-20 Bahrain International Circuit (Bahrain)
- 2022-03-27 Saudi Arabian Grand Prix (Saudi Arabia)
- 2022-04-10 Albert Park Grand Prix Circuit (Australia)

#3 Passing data

1. Create a overview-item in our overview folder with the Angular CLI:
   ng g c overview/overview-item

2. Define (add) a @Input() property in the overview-item.component.ts

```
export class OverviewItemComponent {
 @Input() public race: Race | undefined;
}
```

3. Instead of <ul> <li> we should use the <app-overview-item> component

```
<h1>{{ title }} {{ year }}</h1>
<app-overview-item *ngFor="let race of races" [race]="race"></app-overview-item>
```

4. Use the race property in the overview-item.component.html

```
{{ race?.name }}
```

#4 Navigate to other pages

#4 Setup routes

1. Open your app-routing.module.ts

2. Create your first route in the routes variable

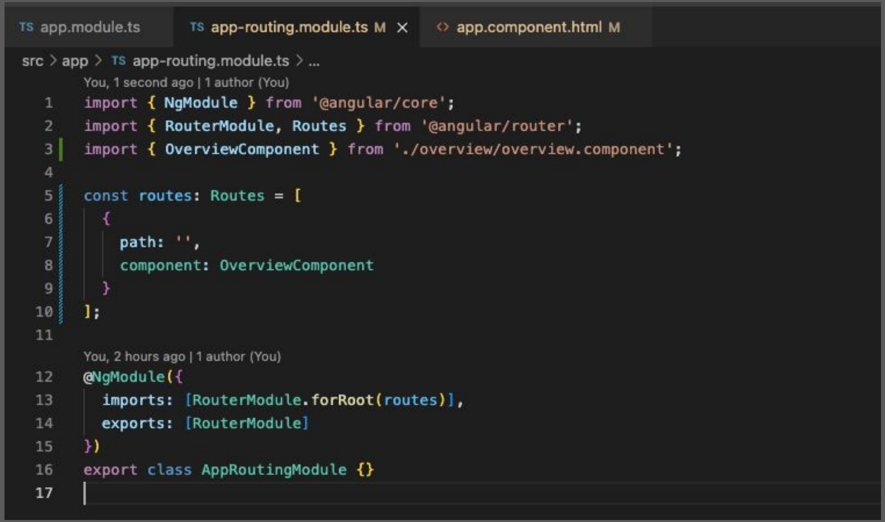
```
const routes: Routes = [
  {
    path: '',
    component: OverviewComponent
  }
];
```

3. Change in app.component.html

```
<app-overview></app-overview>
```

into

```
<router-outlet></router-outlet>
```


TS app.module.ts
TS app-routing.module.ts M
app.component.html M
src > app > TS app-routing.module.ts > ...
You, 1 second ago | 1 author (You)

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { OverviewComponent } from './overview/overview.component';

const routes: Routes = [
  {
    path: '',
    component: OverviewComponent
  }
];

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

You, 2 hours ago | 1 author (You)

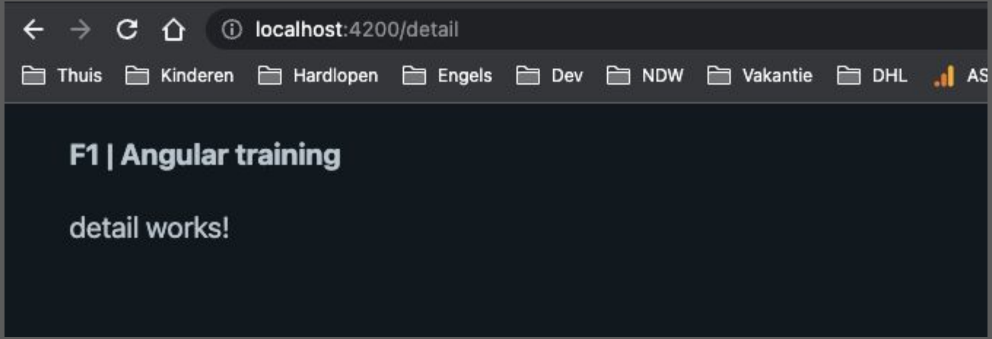
CINQ.

#4 Add a detail module and component

1. Create a detail module and component (same as the previous steps)

2. Add a detail path in app-routing.module.ts

3. Visit http://localhost:4200/detail



F1 | Angular training

detail works!

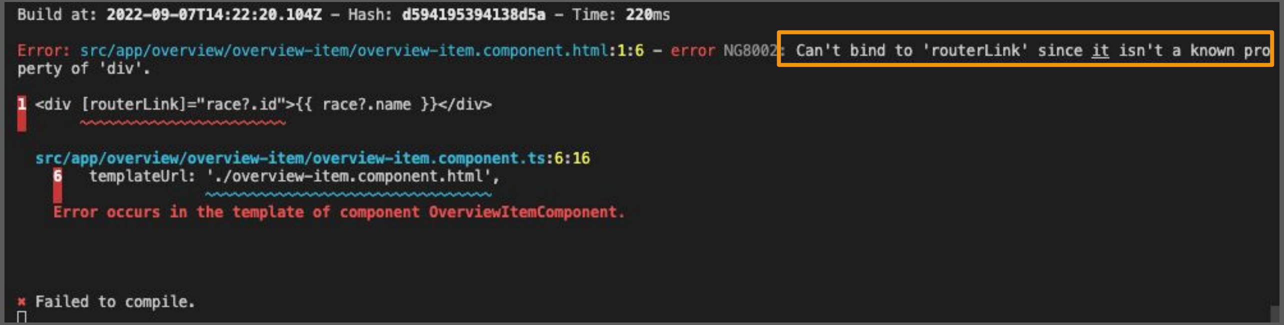
#4 Link to detail

1. Open your race.model.ts and add an “id” property

2. Add a id property in your this.races from overview.component.ts

3. Add a [routerLink] attribute in the overview-item.component.ts

```
<a [routerLink]="['detail', race?.id]">{{ race?.name }}</a>
```



```
Build at: 2022-09-07T14:22:20.104Z - Hash: d594195394138d5a - Time: 220ms

Error: src/app/overview/overview-item/overview-item.component.html:1:6 - error NG8002: Can't bind to 'routerLink' since it isn't a known pro
perty of 'div'.

1 <div [routerLink]="race?.id">{{ race?.name }}</div>
       ~~~~~~~~~~~~~~~~~~~~~~~

  src/app/overview/overview-item/overview-item.component.ts:6:16
    6   templateUrl: './overview-item.component.html',
                     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Error occurs in the template of component OverviewItemComponent.




× Failed to compile.
```

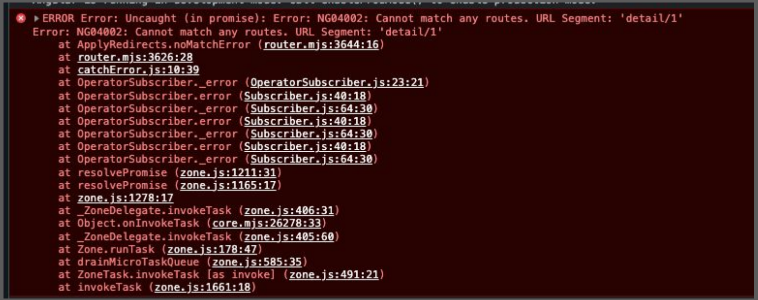
CINQ.

#4 Link to detail

1. Because we use a feature module, we should add a specific import for the BrowserModule in overview.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { OverviewComponent } from './overview.component';
import { OverviewItemComponent } from './overview-item/overview-item.component';
import { RouterModule } from '@angular/router';

@NgModule({
 declarations: [OverviewComponent, OverviewItemComponent],
 imports: [CommonModule, RouterModule],
 exports: [OverviewComponent]
})
export class OverviewModule {}
```

2. When you have clicked on the link there is still an error



```
▸ERROR Error: Uncaught (in promise): Error: NG04002: Cannot match any routes. URL Segment: 'detail/1'
Error: NG04002: Cannot match any routes. URL Segment: 'detail/1'
    at ApplyRedirects.noMatchError (router.mjs:3644:16)
    at router.mjs:3626:28
    at catchError.js:10:39
    at OperatorSubscriber._error (OperatorSubscriber.js:23:21)
    at OperatorSubscriber.error (Subscriber.js:40:18)
    at OperatorSubscriber._error (Subscriber.js:64:30)
    at OperatorSubscriber.error (Subscriber.js:40:18)
    at OperatorSubscriber._error (Subscriber.js:64:30)
    at OperatorSubscriber.error (Subscriber.js:40:18)
    at OperatorSubscriber._error (Subscriber.js:64:30)
    at resolvePromise (zone.js:1211:31)
    at resolvePromise (zone.js:1165:17)
    at zone.js:1278:17
    at _ZoneDelegate.invokeTask (zone.js:406:31)
    at Object.onInvokeTask (core.mjs:26278:33)
    at _ZoneDelegate.invokeTask (zone.js:405:60)
    at Zone.runTask (zone.js:178:47)
    at drainMicroTaskQueue (zone.js:585:35)
    at ZoneTask.invokeTask [as invoke] (zone.js:491:21)
    at invokeTask (zone.js:1661:18)
```

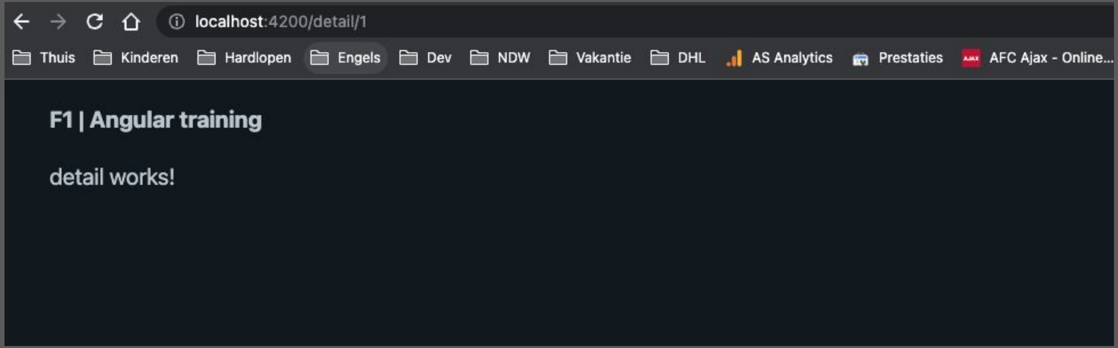
#4 Add param for id

1. Change in app-routing.module.ts

```
{
  path: 'detail',
  component: DetailComponent
}
```

Into

```
{
  path: 'detail/:id',
  component: DetailComponent
}
```


localhost:4200/detail/1

F1 | Angular training

detail works!

#4 But... we use modules

- I always recommend using lazy loading for modules
- By default all routes/modules load, also when not used
- Lazy loading is a best practice in the case of (more significant) projects
- With a few simple steps, we can activate lazy loading, lets try...

#4 Use modules

1. Change the routes const in app-routing.module.ts

```
const routes: Routes = [
  {
    path: '',
    loadChildren: () => import('./overview/overview.module').then((m:
typeofimport('./overview/overview.module')) => m.OverviewModule)
  },
  {
    path: 'detail/:id',
    loadChildren: () => import('./detail/detail.module').then((m: typeof import('./detail/detail.module'))
=> m.DetailModule)
  }
];
```

#4 Add routes in overview

1. Add routes in overview.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { OverviewComponent } from './overview.component';
import { OverviewItemComponent } from './overview-item/overview-item.component';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: '',
    component: OverviewComponent
  }
];

@NgModule({
  declarations: [OverviewComponent, OverviewItemComponent],
  imports: [CommonModule, RouterModule.forChild(routes)],
})
export class OverviewModule {}
```

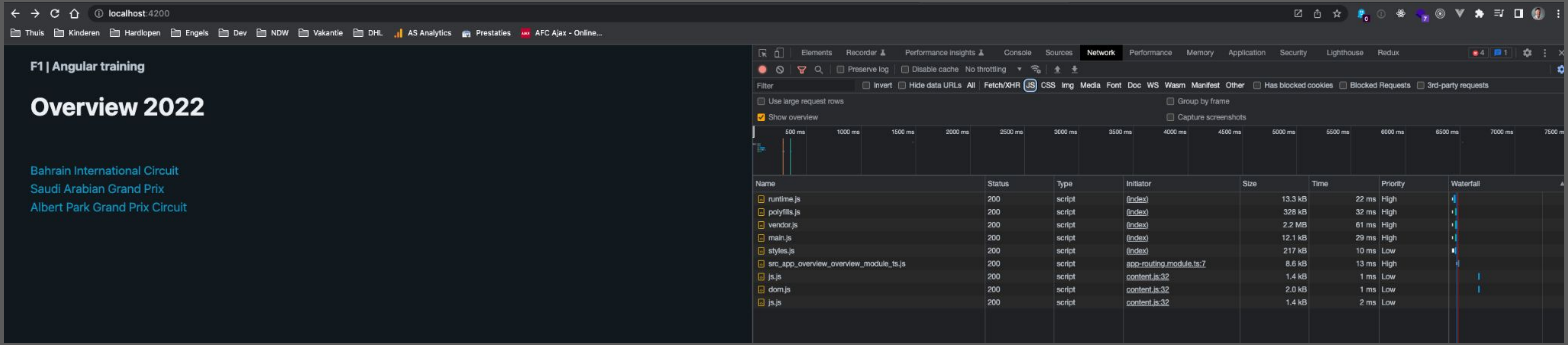
#4 Add routes in detail

1. Add routes in detail.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { DetailComponent } from './detail.component';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: ':id',
    component: DetailComponent
  }
];

@NgModule({
 imports: [CommonModule, RouterModule.forChild(routes)],
 declarations: [DetailComponent]
})
export class DetailModule {}
```

CINQ.

#4 Lazy loaded



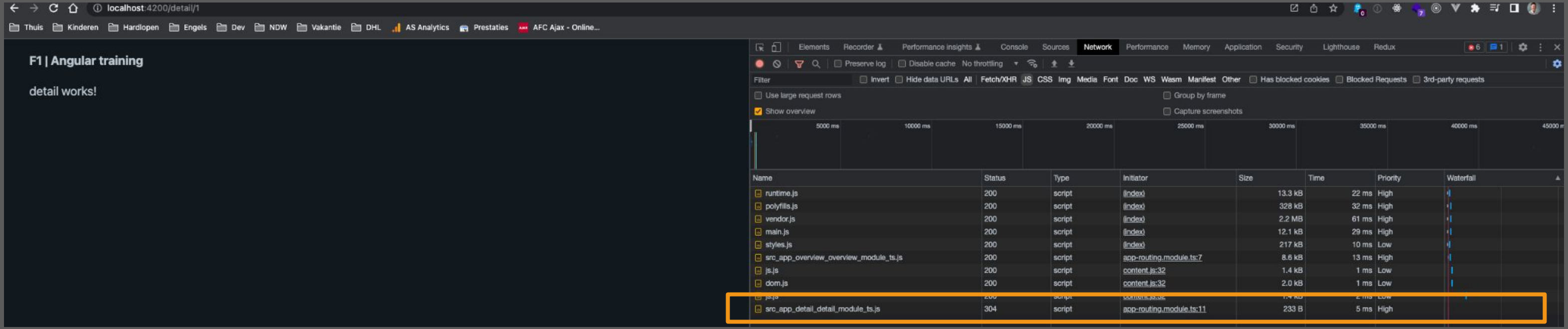
localhost:4200

F1 | Angular training

Overview 2022

Bahrain International Circuit
Saudi Arabian Grand Prix
Albert Park Grand Prix Circuit

Name	Status	Type	Initiator	Size	Time	Priority
runtime.js	200	script	(index)	13.3 kB	22 ms	High
polyfills.js	200	script	(index)	328 kB	32 ms	High
vendor.js	200	script	(index)	2.2 MB	61 ms	High
main.js	200	script	(index)	12.1 kB	29 ms	High
styles.js	200	script	(index)	217 kB	10 ms	Low
src_app_overview_overview_module_ts.js	200	script	app-routing.module.ts:7	8.6 kB	13 ms	High
js.js	200	script	content.js:32	1.4 kB	1 ms	Low
dom.js	200	script	content.js:32	2.0 kB	1 ms	Low
js.js	200	script	content.js:32	1.4 kB	2 ms	Low



localhost:4200/detail/1

F1 | Angular training

detail works!

Name	Status	Type	Initiator	Size	Time	Priority
runtime.js	200	script	(index)	13.3 kB	22 ms	High
polyfills.js	200	script	(index)	328 kB	32 ms	High
vendor.js	200	script	(index)	2.2 MB	61 ms	High
main.js	200	script	(index)	12.1 kB	29 ms	High
styles.js	200	script	(index)	217 kB	10 ms	Low
src_app_overview_overview_module_ts.js	200	script	app-routing.module.ts:7	8.6 kB	13 ms	High
js.js	200	script	content.js:32	1.4 kB	1 ms	Low
dom.js	200	script	content.js:32	2.0 kB	1 ms	Low
js.js	200	script	content.js:32	1.4 kB	2 ms	Low
src_app_detail_detail_module_ts.js	304	script	app-routing.module.ts:11	233 B	5 ms	High

CINQ.

#5 Writing services and fetching data

#5 Dependency injection

- Dependency Injection (DI) is a design pattern and mechanism
- Create and deliver some features to your application
- Most of the times services are a dependency, but you can also use them for values, config, functions, etc
- A provider is an instruction to the Dependency Injection system
- @Inject() and @Injectable()
- Register your injectable functionality in your @NgModule, @Component, or @Directive, this is what we call a provider

#5 Inject a service

- Specify the type of your provider in the constructor

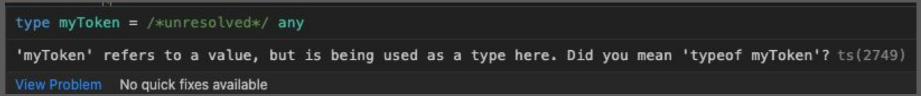
```
import { Component, OnInit } from '@angular/core';
import { Race } from '../shared/races/races.model';
import { RacesService } from '../shared/races/races.service';

@Component({
 selector: 'app-overview',
 templateUrl: './overview.component.html',
 styleUrls: ['./overview.component.scss']
})
export class OverviewComponent implements OnInit {
 …

 constructor(private readonly racesService: RacesService)  {}

 public ngOnInit(): void {
```

- The type definition here is racesServices (note the capital R)

#5 When you use @Inject()?

- @Inject() should be use when you can not inject as type

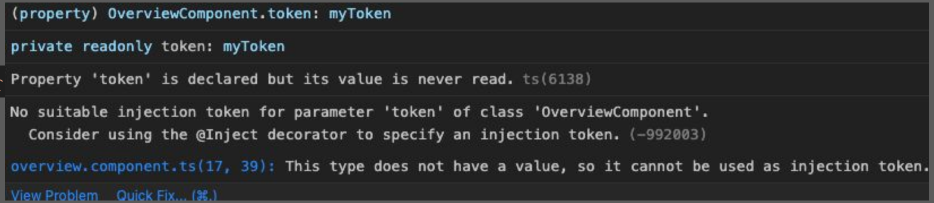
```
const myToken = 'myTokenExample';
export class OverviewComponent implements OnInit {
 constructor(private readonly token: myToken) {}
```



```
type myToken = /*unresolved*/ any
'myToken' refers to a value, but is being used as a type here. Did you mean 'typeof myToken'? ts(2749)
View Problem   No quick fixes available
```

- Use @Inject() to describe the type

```
import { Component, Inject, OnInit } from '@angular
const myToken = 'myTokenExample';

@Component({
 …
 providers: [{ provide: myToken, useValue: myToken }]
})
export class OverviewComponent implements OnInit {

 constructor(@Inject(myToken) private readonly token: string) {
   // this.token outputs "myTokenExample"
 }
```



```
(property) OverviewComponent.token: myToken
private readonly token: myToken
Property 'token' is declared but its value is never read. ts(6138)
No suitable injection token for parameter 'token' of class 'OverviewComponent'.
  Consider using the @Inject decorator to specify an injection token. (-992003)
overview.component.ts(17, 39): This type does not have a value, so it cannot be used as injection token.
View Problem   Quick Fix... (⌘.)
```

#5 When you use @Injectable()?

- @Injectable describe how we our services should use

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

@Injectable({
 providedIn: 'root'
})
export class RacesService {
 // logic
}
```

- `providedIn: 'root'` should be provided in the root
- `providedIn: 'any'` should be provided for lazy loading
- `providedIn: Module` should be provided in a particular @NgModule

#5 When you not use @Injectable()?

- Then you should provide them into your @NgModule, @Component, or @Directive
- Don’t forget to use an empty @Injectable() decorator above your class
- Provider examples for component and module:

```
@Component({
 selector: 'app-overview',
 templateUrl: './overview.component.html',
 styleUrls: ['./overview.component.scss'],
 providers: [RacesService]
})

@NgModule({
 declarations: [OverviewComponent, OverviewItemComponent],
 imports: [CommonModule, RouterModule.forChild(routes)],
 providers: [RacesService]
})
export class OverviewModule {}
```

#5 Multi providers

- You might one day come across a case requiring you to use multiple values for one token, but such cases are not extremely common. This option is usually used with Angular built-in tokens.

- This multi: true option is pretty rare and can therefore be confusing when it appears in some Angular documentation. But it does what the name says it does: it lets you define multiple values for one given token. Use cases are not obvious, and you will rarely have to use it with one of your own tokens. It is generally used with Angular built-in ones.

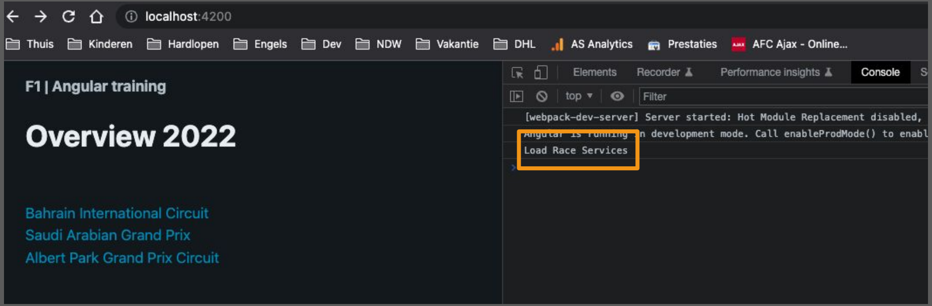
  https://levelup.gitconnected.com/angular-dependency-injection-multi-providers-87c55acc4857

```
providers: [
   { provide: myToken, useValue: 'myTokenExample1', multi: true },
   { provide: myToken, useValue: 'myTokenExample2', multi: true }
]
```

#5 Create a service

1. Create a races services in src/shared/races:
   ng g s shared/races/races or ng generate service shared/races/races
   https://angular.io/cli/generate#service

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

@Injectable({
 providedIn: 'root'
})
export class RacesService {
 constructor() {
   console.log('Load Race Services');
 }
}
```



localhost:4200

F1 | Angular training

Overview 2022

Bahrain International Circuit
Saudi Arabian Grand Prix
Albert Park Grand Prix Circuit

```
[webpack-dev-server] Server started: Hot Module Replacement disabled,
Angular is running in development mode. Call enableProdMode() to enabl
Load Race Services
```

2. Inject the service in overview.component.ts

```
 constructor(private readonly racesService: RacesService) {}
```

#5 Setup for backend communication

Communicating with backend services using HTTP

1. Import (add) HttpClientModule in your app.module.ts

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

@NgModule({
 declarations: [AppComponent],
 imports: [BrowserModule, AppRoutingModule, CoreModule, HttpClientModule],
```

2. Then inject the HttpClient service as a dependency in races.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable({
 providedIn: 'root'
})
export class RacesService {
 constructor(private readonly http: HttpClient) {
   console.log('Load Race Services');
 }
}
```

CINQ.

#5 Requesting data from a server

1. Create a getRaces method in races.services.ts

```
public getRaces() {}
```

2. Get data from the API with the http injector (build-in from Angular)

```
public getRaces() {
   return this.http.get('https://ergast.com/api/f1/2022.json');
}
```

3. Refactor the race interface (should be the same as the json response)

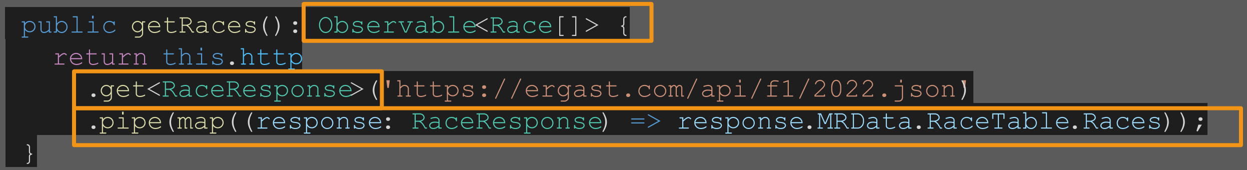
4. Add type definitions to the method getRaces()

```
public getRaces(): Observable<Race[]> {
   return this.http
     .get<RaceResponse>('https://ergast.com/api/f1/2022.json')
     .pipe(map((response: RaceResponse) => response.MRData.RaceTable.Races));
}
```

#5 Observables / RxJS

- It’s not a build-in functionality from Angular, but from RxJS
- It’s a recommended way to use in combination with Angular
- With Observables can you develop async
- Observables open up a continuous channel of communication in which multiple values of data can be emitted over time
- It is a pattern of dealing with data by using array-like operations to parse, modify and maintain data
- Read the following documentation about RxJS
  https://rxjs.dev

#5 What happened in this case?



```
public getRaces(): Observable<Race[]> {
   return this.http
     .get<RaceResponse>('https://ergast.com/api/f1/2022.json)
     .pipe(map((response: RaceResponse) => response.MRData.RaceTable.Races));
 }
```

#5 Binding data from API response

1. Update the races property in overview.component.ts

```
public races: Observable<Race[]> = of([]);
```

2. Update this.races in ngOnInit

```
this.races = this.racesService.getRaces();
```

3. Update way of data binding in overview.component.html

```
<app-overview-item *ngFor="let race of races | async" [race]="race"></app-overview-item>
```

#5 Change template

1. Update date in overview-item.component.html

```
<article>
 <header>#{{ race?.round }} {{ race?.Circuit?.circuitName }} ({{
race?.Circuit?.Location?.country}})</header>
 <div>
   <div><strong>First:</strong> {{ race?.FirstPractice?.date }} {{ race?.FirstPractice?.time }}</div>
   <div><strong>Second:</strong> {{ race?.SecondPractice?.date }} {{ race?.SecondPractice?.time }}</div>
   <div><strong>Third:</strong> {{ race?.ThirdPractice?.date }} {{ race?.ThirdPractice?.time }}</div>
   <div><strong>Qualifying:</strong> {{ race?.Qualifying?.date }} {{ race?.Qualifying?.time }}</div>
   <div><strong>Sprint:</strong> {{ race?.Sprint?.date }} {{ race?.Sprint?.time }}</div>
   <div><strong>Race:</strong> {{ race?.date }} {{ race?.time }}</div>
 </div>
 <footer>
   <a [routerLink]="['detail', race?.round]" role="button">More</a>
 </footer>
</article>
```

#6 Transform data with pipes

CINQ.

#6 What are pipes?

- A pipe takes data as input and transforms it into output
- There are a few built-in pipes such as DatePipe, UpperCasePipe, LowerCasePipe
  https://angular.io/guide/pipes
- You can also create custom pipes when needed

#6 Using a built-in pipe

- Add a DatePipe in overview-item.component.ts for the race date time
```
<div><strong>Race:</strong> {{ race?.date + ' ' + race?.time | date: 'MM/dd/YYYY | HH:mm' }}</div>
```
- https://angular.io/api/common/DatePipe
- Add a DatePipe for the other date / time value (first, second, third, sprint and qualifying)

#6 Add *ngIf to solve errors

- Add *ngIf for third and sprint values, because sometimes these are null values

```
   <div *ngIf="race?.ThirdPractice">
     <strong>Third:</strong> {{ race?.ThirdPractice?.date + ' ' + race?.ThirdPractice?.time | date:
'MM/dd/YYYY | HH:mm' }}
   </div>
   <div *ngIf="race?.Sprint">
     <strong>Sprint:</strong> {{ race?.Sprint?.date + ' ' + race?.Sprint?.time | date: 'MM/dd/YYYY |
HH:mm' }}
   </div>
```

#6 Create a custom pipe

1. Create a zeroPad module in src/shared
   ng g m shared/zero-pad

2. Create a zeroPad pipe in src/shared/zero-pad
   ng g p shared/zero-pad/zero-pad
   https://angular.io/cli/generate#pipe

3. Export zeroPadPipe in the zeroPadModule
   ```
    exports: [ZeroPadPipe]
   ```

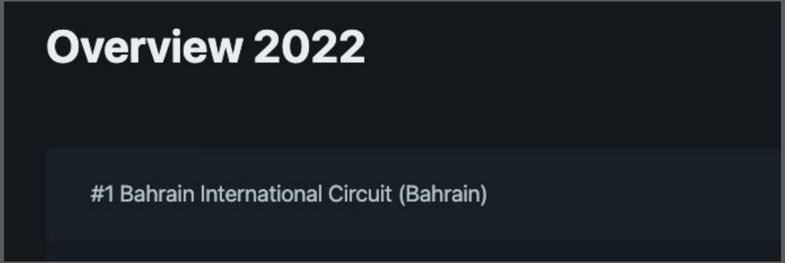
4. Import (add) zeroPadModule in overview.module.ts
   ```
    imports: [CommonModule, RouterModule.forChild(routes), ZeroPadModule]
   ```

5. Add zeroPad pipe for the {{ race?.round }} in overview-item.component.html
   ```
   <header>#{{ race?.round | zeroPad }}</header>
   ```

#6 Customize zeroPad pipe

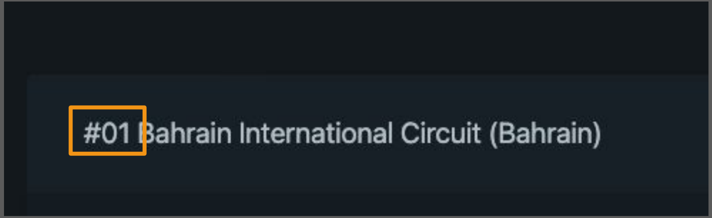
1. Update zeroPad in zero-pad.pipe.ts

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

@Pipe({
 name: 'zeroPad'
})
export class ZeroPadPipe implements PipeTransform {
 public transform(num: string | undefined, places: number): string | null {
   if (!num) {
     return null;
   }
   const zero = places - num.length + 1;
   return Array(+(zero > 0 && zero)).join('0') + num;
 }
}
```



Overview 2022

#1 Bahrain International Circuit (Bahrain)



#01 Bahrain International Circuit (Bahrain)

2. Add param for places argument where we are using the zeroPad pipe

```
<header>#{{ race?.round | | zeroPad: 2 }}</header>
```

CINQ.

TIMETABLE DAY 2

{

09:30 | Welcome
#0 | About yesterday
#7 | Build a form
#8 | Directives
16:00 | Recap and closing

}

#7 Build a form

#7 Get query param from route

1. Declare a raceId property to detail.component.ts

```
private raceId: number | undefined;
```

2. Inject ActivatedRoute in the constructor

```
constructor(private readonly route: ActivatedRoute)  {}
```

3. Save param in raceId property

```
this.raceId = this.route.snapshot.params['id'];
```

#7 Fetch data for detail

1. Inject RacesService in the constructor

```
constructor(private readonly route: ActivatedRoute, private readonly racesService: RacesService) {}
```

2. Add race property

```
public race: Race | undefined;
```

- Add getRace in RacesService

```
 public getRace(id: number): Observable<Race> {
   return this.http
     .get<RaceResponse>(`https://ergast.com/api/f1/2022${id}.json`)
     .pipe(map((response: RaceResponse) => response.MRData.RaceTable.Races[0]));
 }
```

- Add race data from backend in ngOnInit from detail.component.ts and bind data in template

```
   if (this.raceId) {
     this.racesService.getRace(this.raceId).subscribe((race: Race) => (this.race = race));
   }
```

CINQ.

#7 Create subroutes for detail

1. Create a detail-qualifying, detail-sprint and detail-result component

2. Update the routes in detail.module.ts

```
const routes: Routes = [
  {
     path: ':id',
     component: DetailComponent,
     children: [
        {
          path: 'qualifying',
          component: DetailQualifyingComponent
        },
        {
          path: 'sprint',
          component: DetailSprintComponent
        },
        {
          path: 'race',
          component: DetailResultsComponent
        }
     ]
  }
];
```

#7 Add router outlet and links

1. Add a <router-outlet> in your detail.component.ts

2. Setup a list of menu items

```
<ul>
        <li><a [routerLink]="['./', 'qualifying']" routerLinkActive="active">Qualifying</a></li>
        <li><a [routerLink]="['./', 'sprint']" routerLinkActive="active">Sprint</a></li>
        <li><a [routerLink]="['./', 'race']" routerLinkActive="active">Race</a></li>
        <li><a [routerLink]="['/']">Back</a></li>
</ul>\\
```

#7 Fetch data for qualifying, sprint and race

1. Add for each component a way of fetching data (based on param and path in url), see:

   https://github.com/cinqict/angular-training/blob/feature/007-build-a-form/src/app/detail/detail-results/detail-results.component.ts

   https://github.com/cinqict/angular-training/blob/feature/007-build-a-form/src/app/shared/races/races.service.ts

   Endpoints:
   https://ergast.com/api/f1/2022/4/results.json
   https://ergast.com/api/f1/2022/4/sprint.json
   https://ergast.com/api/f1/2022/4/results.json

#7 What are forms?

Angular knows Reactive forms and template-driven forms process and manage form data differently. Each approach offers different advantages.

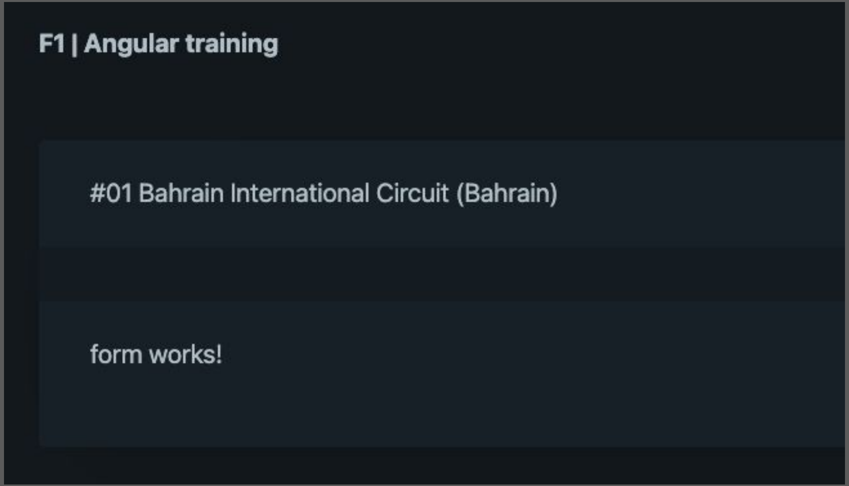
Reactive forms
Provide direct, explicit access to the underlying form's object model. Compared to template-driven forms, they are more robust: they're more scalable, reusable, and testable. If forms are a key part of your application, or you're already using reactive patterns for building your application, use reactive forms.

Template-driven forms
Rely on directives in the template to create and manipulate the underlying object model. They are useful for adding a simple form to an app, such as an email list signup form. They're straightforward to add to an app, but they don't scale as well as reactive forms. If you have very basic form requirements and logic that can be managed solely in the template, template-driven forms could be a good fit.

https://angular.io/guide/forms-overview

#7 We want to comment

1. Add a comments module in shared/comments
2. Add a comments form in shared/comments
3. Export the form in the comment module
4. Import CommentModule in DetailModule
5. Use selector of form form component in detail template



F1 | Angular training

#01 Bahrain International Circuit (Bahrain)

form works!

#7 Setup simple form

1. Import (add) ReactiveFormsModule in CommentsModule

2. Declare a form property in form component

```
public form: FormGroup = new FormGroup({});
```

3. Define the form property in the ngOnInit

```
   this.form = new FormGroup({
     firstname: new FormControl(null),
     lastname: new FormControl(null),
     age: new FormControl(null),
     email: new FormControl(null),
     comment: new FormControl(null)
   });
```

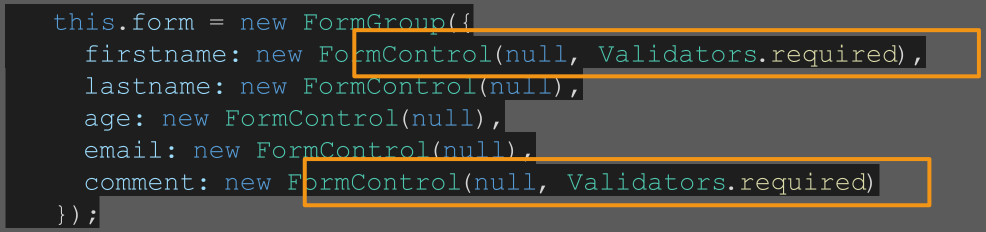
You can also build a form with form builder
https://angular.io/start/start-forms

CINQ.

#7 Data binding form

```
<form [formGroup]="form">
 <div class="grid">
   <label for="firstname">
     First name
     <input type="text" id="firstname" name="firstname" placeholder="First name" formControlName="firstname" />
   </label>

   <label for="lastname">
     Last name
     <input type="text" id="lastname" name="lastname" placeholder="Last name" formControlName="lastname" />
   </label>
 </div>

… -> TODO: placeholder for age and email address

 <label for="comment">Comment</label>
 <textarea id="comment" name="comment" placeholder="Comment" formControlName="comment"></textarea>

 <button type="submit">Submit</button>
</form>
```

#7 Add built-in validators



```
this.form = new FormGroup({
  firstname: new FormControl(null, Validators.required),
  lastname: new FormControl(null),
  age: new FormControl(null),
  email: new FormControl(null),
  comment: new FormControl(null, Validators.required)
});
```

#7 Show errors

```
 <label for="comment">Comment</label>
 <textarea
   id="comment"
   name="comment"
   placeholder="Comment"
   formControlName="comment"
   [attr.aria-invalid]="form.get('comment')?.invalid && (form.get('comment')?.dirty ||
form.get('comment')?.touched) ? true : null"
 ></textarea>
 <p *ngIf="form.get('comment')?.invalid && (form.get('comment')?.dirty || form.get('comment')?.touched)">Comment
is required.</p>

 <button type="submit" [disabled]="form.invalid">Submit</button>
```

#7 Add a custom validation

1. Add an empty form-forbidden.validator.ts in your form folder

2. Create a custom validator

```
import { AbstractControl, ValidationErrors, ValidatorFn } from '@angular/forms';
export const forbiddenValidator =
 (word: string): ValidatorFn =>
 (control: AbstractControl): ValidationErrors | null => {
   let isForbidden = false;
   try {
     isForbidden = control.value.includes(word);
   } catch (error) {
     isForbidden = false;
   }
   return isForbidden ? { forbidden: { value: true } } : null;
 };
```

3. Add forbiddenValidator for comment control

```
     comment: new FormControl(null, [Validators.required, forbiddenValidator('soccer')])
```

https://netbasal.com/how-to-trim-the-value-of-angulars-form-control-87660941e6cb

CINQ.

#7 Show your forbidden error

```
<div *ngIf="form.get('comment')?.invalid && (form.get('comment')?.dirty || form.get('comment')?.touched)">
  <p *ngIf="form.get('comment')?.errors?.['required']">Comment required.</p>
  <p *ngIf="form.get('comment')?.errors?.['forbidden']">Remove forbidden word.</p>
</div>
```

#7 Submit a form

1. Add a (ngSubmit) binding for form element

```
<form [formGroup]="form" (ngSubmit)="onSubmit()">
```

2. Create a onSubmit method in the component

```
 public onSubmit(): void {}
```

3. Create interface for comment (in separate file)

```
export interface Comment {
 firstname: string;
 lastname: string | string;
 age: number | null;
 email: string | null;
 comment: string;
}
```

4. Read values from from (in onSubmit method)

```
const values: Comment = this.form.getRawValue();
```

#7 If we want save in a comment service

1. Add a raceId property to the Comment interface

2. Add a type to the Comment interface

```
export type CommentType = 'all' | 'qualify' | 'sprint' | 'race';

export interface Comment {
…other code
 raceId?: number;
 type: CommentType;
}
```

3. Add a raceId @Input() and type @Input() to form element

```
@Input() public raceId: number | undefined;
@Input() public type: CommentType = 'all';
```

4. Add properties to <app-form> in detail template

```
<app-form [raceId]="raceId" type="all"></app-form>
```

#7 Saving in local property

1. Combine form values and other properties as comment in onSubmit()
   ```
   const values: Comment = { ...this.form.getRawValue(), raceId: this.raceId, type: this.type };
   ```
   https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax

2. Add a comment service

3. Add a comment property subject with BehaviorSubject
   ```
   private readonly commentSubject: BehaviorSubject<Comment[]> = new BehaviorSubject<Comment[]>([]);
   ```

4. Add a saveComments method
   ```
    public saveComments(comment: Comment): void {
      const prevComments = this.commentSubject.getValue();
      const nextComments = [...prevComments, comment];
      this.commentSubject.next(nextComments);
    }
   ```

CINQ.

#7 Call SaveComments

1. Inject CommentService in form component

2. Call saveComments in onSubmit with values
```
this.commentsService.saveComments(values);
```

3. Reset form after submitting
```
this.form.reset();
```

#7 Display comments in new component

1. Create a getComments() method in the CommentsService

```
public getComments(raceId: number, type: CommentType): Observable<Comment[]> {
  return this.commentSubject.asObservable().pipe(
    map((allComments: Comment[]) =>
      allComments.filter((comment: Comment) => {
        const isCommentToRaceAndType = comment.raceId === raceId && comment.type === type;
        return isCommentToRaceAndType;
      })
    )
  );
}
```

2. Create a comment list component
3. Use them in detail component
4. Subscribe to getComments
5. Data binding of comments

#7 Make type property interactive for comments

1. Add a commentType property in your detail.component.ts (don't forget your type!)

2. Add a router injectable from the Angular in the constructor

3. Add a defineType method in our component

```
private definePathType(): void {
    this.commentType = 'all';
}
```

4. Call this.defineType() in the ngOnInit lifecycle

#7 Listen to router event subscription

1. Create a subscribe of events from the router events

```
this.router.events.pipe(filter((event: Event) => event instanceof NavigationEnd)).subscribe(() => {
     this.definePathType();
   });
```

2. Change the definePathType with new logic

```
 private definePathType(): void {
   if (this.route.snapshot.children.length) {
     this.commentType = this.route.snapshot.children[0].url[0].path as CommentType;
   } else {
     this.commentType = 'all';
   }
 }
```

3. Add commentType as property to our list and form component

#8 Directives

#8 What are directives?

Directives are classes that add additional behavior to elements in your Angular applications. Use Angular's built-in directives to manage forms, lists, styles, and what users see.

Components
Used with a template. This type of directive is the most common directive type.

Attribute directives
Change the appearance or behavior of an element, component, or another directive.

Structural directives
Change the DOM layout by adding and removing DOM elements.

https://angular.io/guide/built-in-directives

#8 Built-in attribute

Attribute directives listen to and modify the behavior of other HTML elements, attributes, properties, and components.

NgClass
Adds and removes a set of CSS classes.
```
<div [ngClass]="isSpecial ? 'special' : ''">This div is special</div>
```

NgStyle
Adds and removes a set of HTML styles.
```
<div [ngStyle]="{{ ‘backgroundColor’: ‘#000’ }}">
```

NgModel
Adds two-way data binding to an HTML form element.

#8 Built-in structural directives

Structural directives are responsible for HTML layout. They shape or reshape the DOM's structure, typically by adding, removing, and manipulating the host elements to which they are attached.

***ngIf**
Conditionally creates or disposes of subviews from the template.

***ngFor**
Repeat a node for each item in a list.

***ngSwitch**
A set of directives that switch among alternative views.

#8 Setup an attribute directive

1. Add a module and directive:
   ng g m shared/highlight
   ng g d shared/highlight/highlight

2. Add (inject) the elementRef from Angular in the constructor of the highlight directive

```
constructor(private readonly elementRef: ElementRef) {}
```

3. Use the directive in the component

```
<div appHighlight>{{ result.status }}</div>
```

4. Add a green text after loading

```
public ngAfterViewInit(): void {
   if (this.elementRef.nativeElement.innerText === 'Finished') {
     this.elementRef.nativeElement.style.color = 'green';
   }
}
```

#8 Use HostListener events

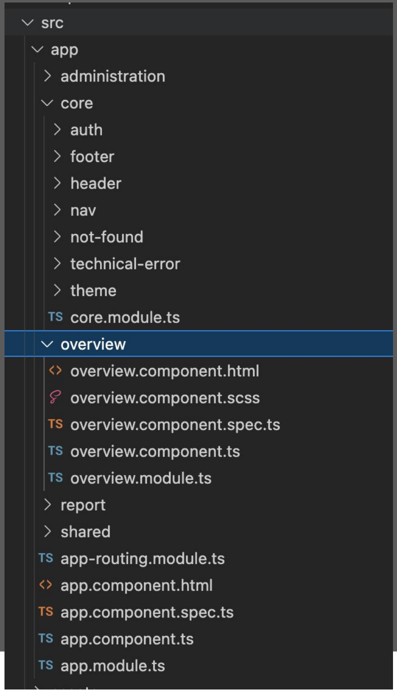
```
@HostListener('mouseenter') public onMouseEnter(): void {
  if (this.isFinished) {
    this.elementRef.nativeElement.style.backgroundColor = 'green';
    this.elementRef.nativeElement.style.color = 'white';
  }
}

@HostListener('mouseleave') public onMouseLeave(): void {
  if (this.isFinished) {
    this.elementRef.nativeElement.style.color = 'green';
    this.elementRef.nativeElement.style.backgroundColor = 'transparent';
  }
}
```

#9 Testing and publish your app (in practice)

CINQ.

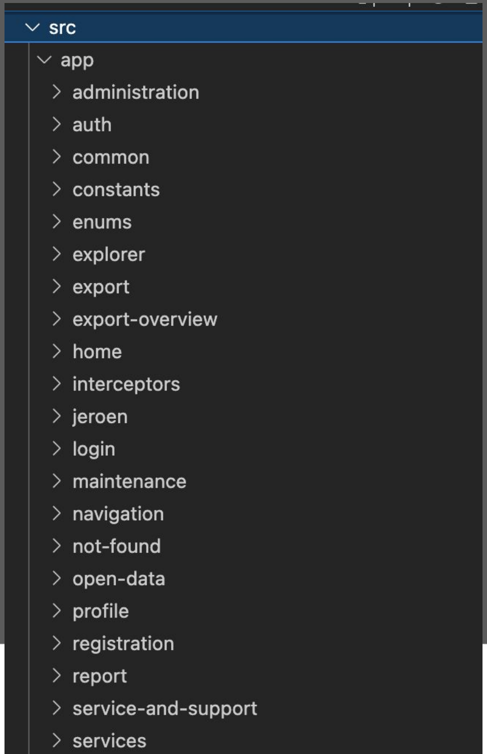
#9 File structure

Best practice



```
src
  app
    administration
    core
      auth
      footer
      header
      nav
      not-found
      technical-error
      theme
      core.module.ts
    overview
      overview.component.html
      overview.component.scss
      overview.component.spec.ts
      overview.component.ts
      overview.module.ts
    report
    shared
    app-routing.module.ts
    app.component.html
    app.component.spec.ts
    app.component.ts
    app.module.ts
```

Bad practice



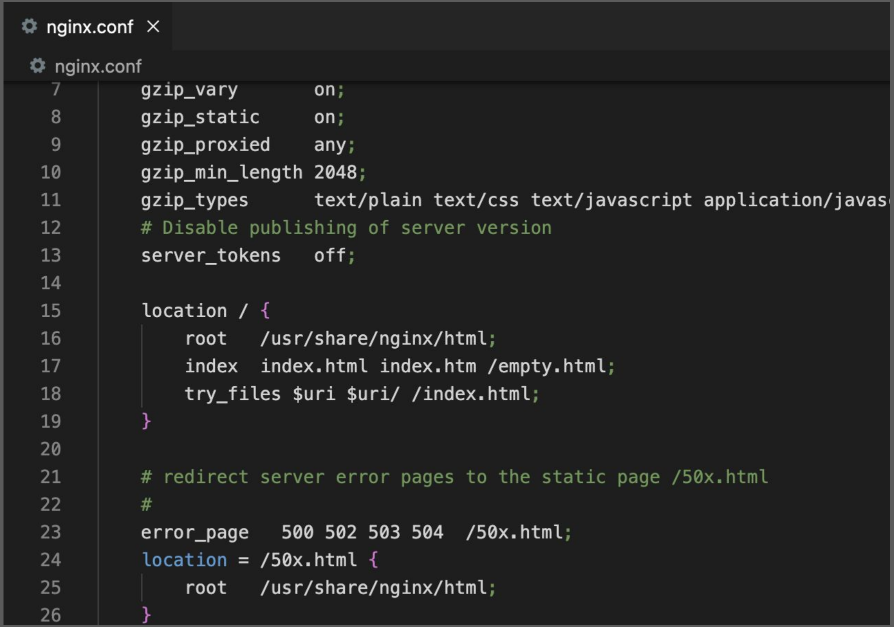
```
src
  app
    administration
    auth
    common
    constants
    enums
    explorer
    export
    export-overview
    home
    interceptors
    jeroen
    login
    maintenance
    navigation
    not-found
    open-data
    profile
    registration
    report
    service-and-support
    services
```

#9 Unit test

- Angular use Karma as testing framework
- Use npm run test or ng test to start unit testing
- Isolated test (no dependencies on other components, services, etc)
- Testing structure
  - Describe
    - beforeEach() -> setup your test suite
    - it(‘test something 1’) -> test specific logic
    - it(‘test something 2’) -> test specific logic
    - afterEach() -> if need
- In practice unit test

#9 Build

- npm run build or ng build
  -> generate a production build of our Angular application
- After running you find a dist directory in your codebase
- This can be used to publish your code with nginx or other possible ways (like static hosting as Vercel or cloudflare)


nginx.conf
nginx.conf

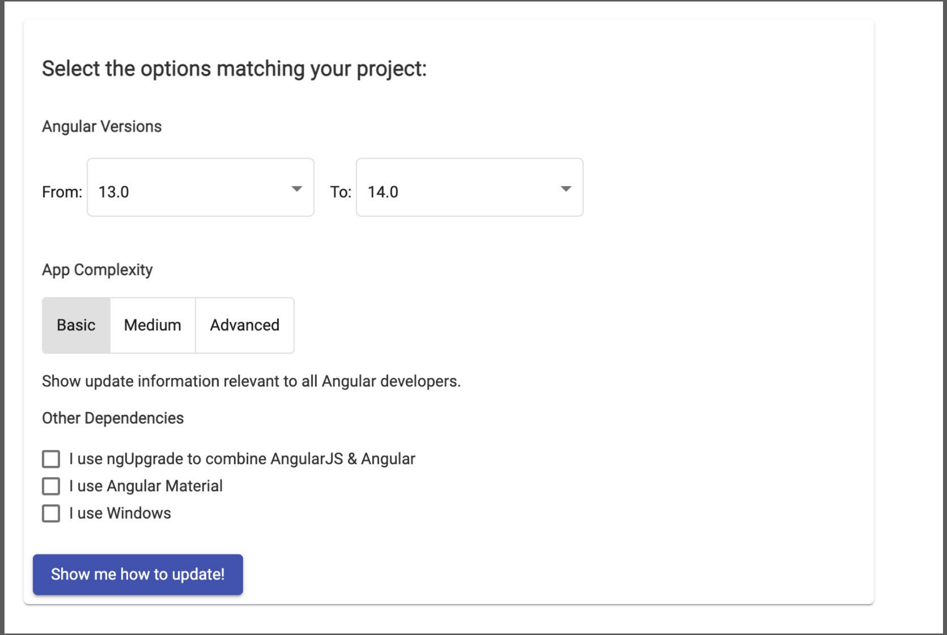
```
    gzip_vary       on;
    gzip_static     on;
    gzip_proxied    any;
    gzip_min_length 2048;
    gzip_types      text/plain text/css text/javascript application/javas
    # Disable publishing of server version
    server_tokens   off;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm /empty.html;
        try_files $uri $uri/ /index.html;
    }

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
```

CINQ.

#9 Code quality

- Prettier
    - You press save and code is formatted
    - No need to discuss style in code review
    - Saves you time and energy
- Eslint
    - Find and fix problems in your JavaScript code
- Renovate
    - Save time and reduce risk by automating dependency updates in software projects.
- Husky (commit hooks)
    - Husky improves your commits and more 🐶 woof!

#9 Angular Update Guide



Select the options matching your project:

Angular Versions

From: 13.0 To: 14.0

App Complexity

[✓ Basic] Medium Advanced

Show update information relevant to all Angular developers.

Other Dependencies

- [ ] I use ngUpgrade to combine AngularJS & Angular
- [ ] I use Angular Material
- [ ] I use Windows

Show me how to update!

https://update.angular.io

#10 Build your own todo app