



# Atos

## Global Knowledge.®

# Angular Advanced Monorepo's - introduction

Peter Kassenaar –  
[info@kassenaar.com](mailto:info@kassenaar.com)

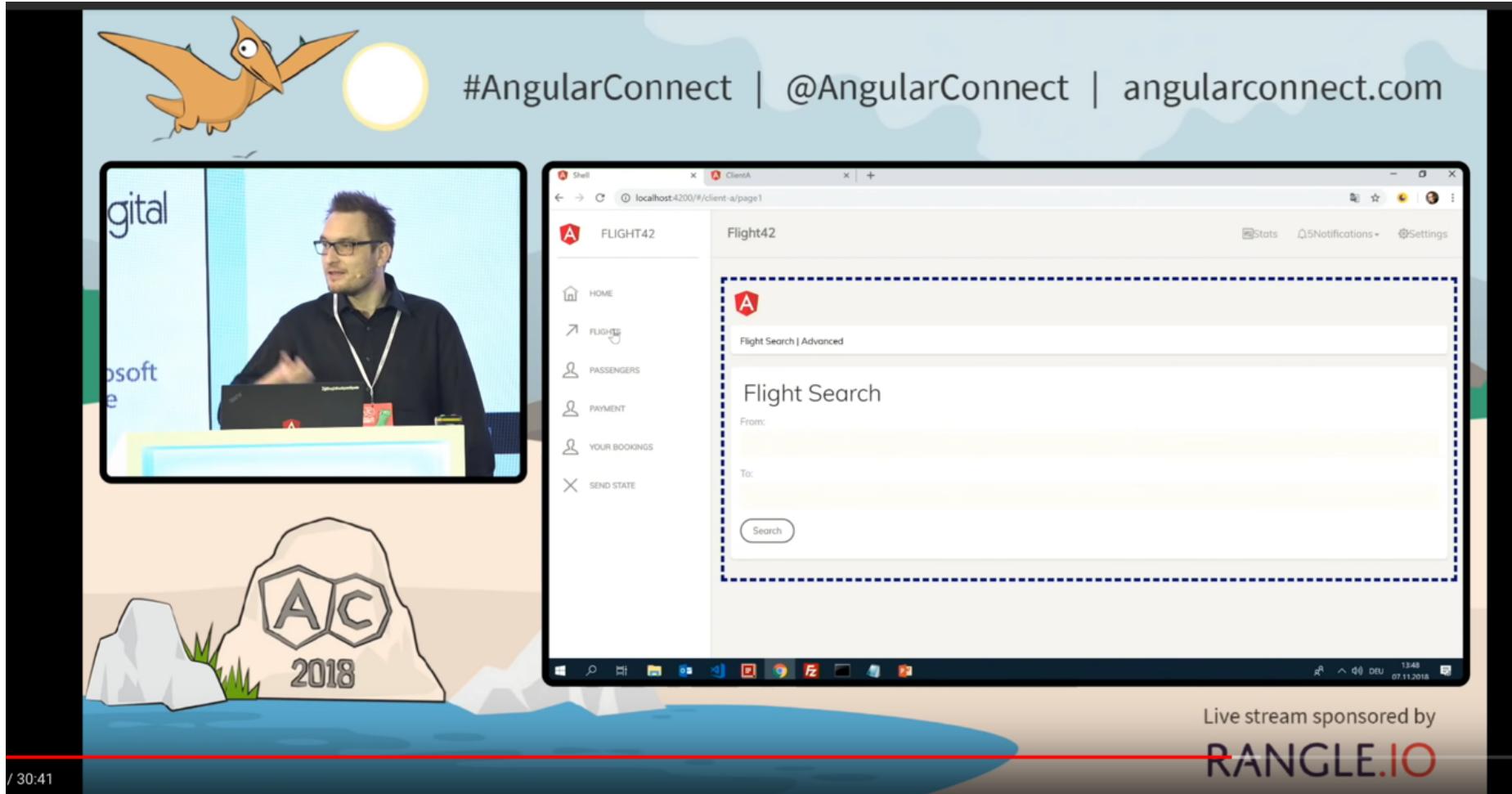
WORLDWIDE LOCATIONS

BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR  
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA

# Angular in the Enterprise

- When? With (really) big(ger) applications
- Multiple solutions, examples:
  - Monorepo approach:  
<https://github.com/PeterKassenaar/ng-monorepo>
  - Micro-app approach:  
<https://github.com/PeterKassenaar/angular-microapp>

# Manfred Steyer – Angular Connect



<https://www.youtube.com/watch?v=YU-fMRs-ZYU>

Code: <https://github.com/PeterKassenaar/angular-microapp>

# What is a monorepo

- Often, bigger applications are not only split up into **modules**, but also in **other applications**
- Applications generally share a lot of code
  - Sharing components
  - Sharing services
  - Sharing pipes, other logic, etc...
- One (opiniated) solution – create a so called *monorepo*
- There are tools that also cover this
  - Nrwl Extensions – free, open source, <https://nrwl.io/nx>
  - Angular CLI as of V6.0.0+, <https://cli.angular.io/>

# Nrwl extensions to create a monorepo



[What is Nx](#)  
[What Nx Provides](#)  
[Why a workspace?](#)  
[Repo on GitHub](#)  
[Free Nx Workspaces Course](#)  
  
[Guides](#)  
  
[Getting Started](#)  
[Nx Workspace](#)  
[Workspace - Specific Schematics](#)  
[Setting Up NgRx](#)  
[Data Persistence](#)  
[AngularJS Upgrade Module](#)  
[AngularJS Downgrade Module](#)



## Nrwl Extensions for Angular

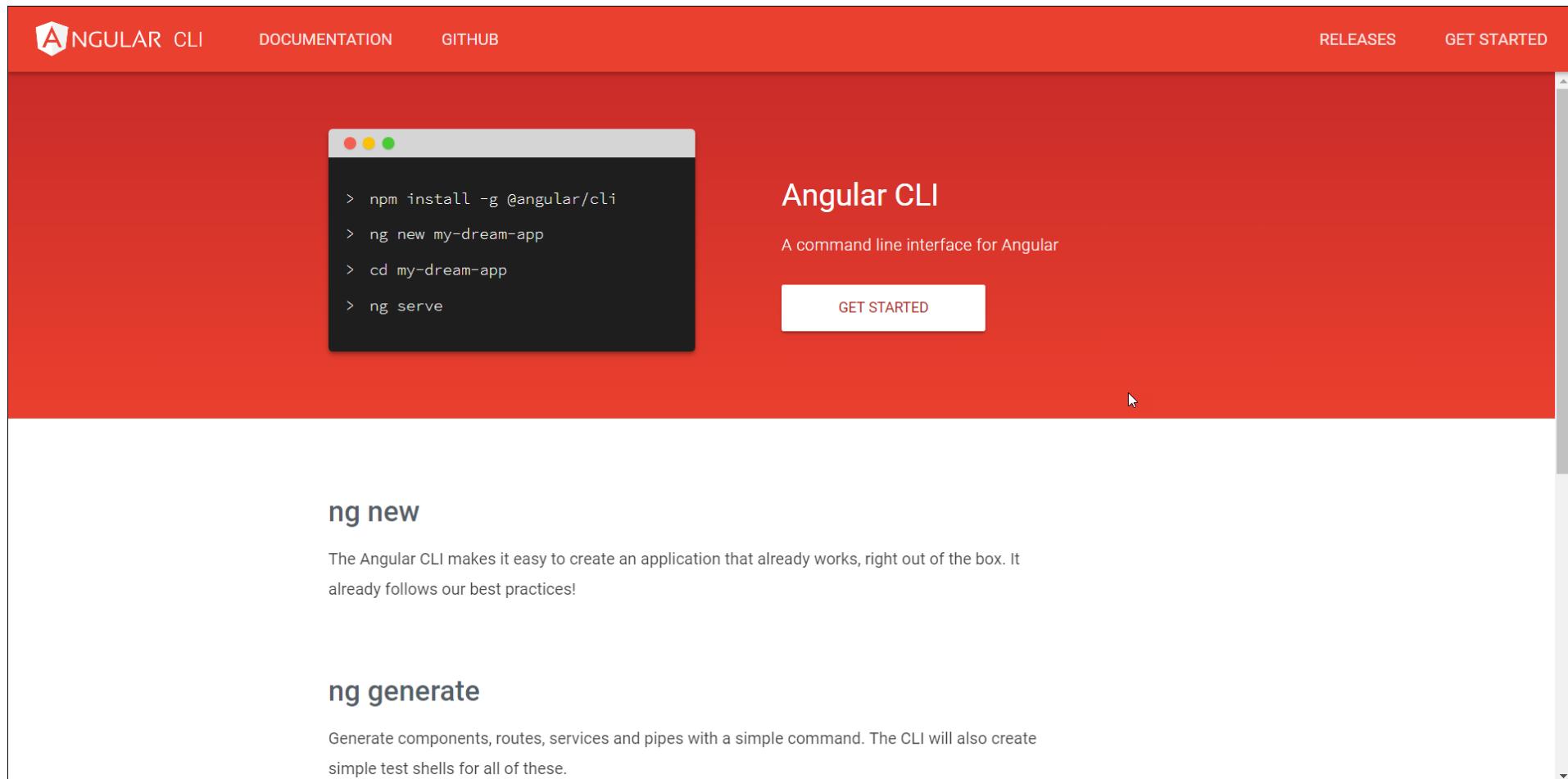
An open source toolkit for enterprise Angular applications.



## Nrwl Extensions for Angular

An open source toolkit for enterprise Angular applications

# Angular CLI – as of V6.0+



The screenshot shows the Angular CLI homepage with a red header. The header includes links for ANGULAR CLI, DOCUMENTATION, GITHUB, RELEASES, and GET STARTED. Below the header, there's a large callout box containing a terminal window with the following commands:

```
> npm install -g @angular/cli  
> ng new my-dream-app  
> cd my-dream-app  
> ng serve
```

Next to the terminal window, the text "Angular CLI" is displayed, followed by "A command line interface for Angular" and a "GET STARTED" button.

**ng new**

The Angular CLI makes it easy to create an application that already works, right out of the box. It already follows our best practices!

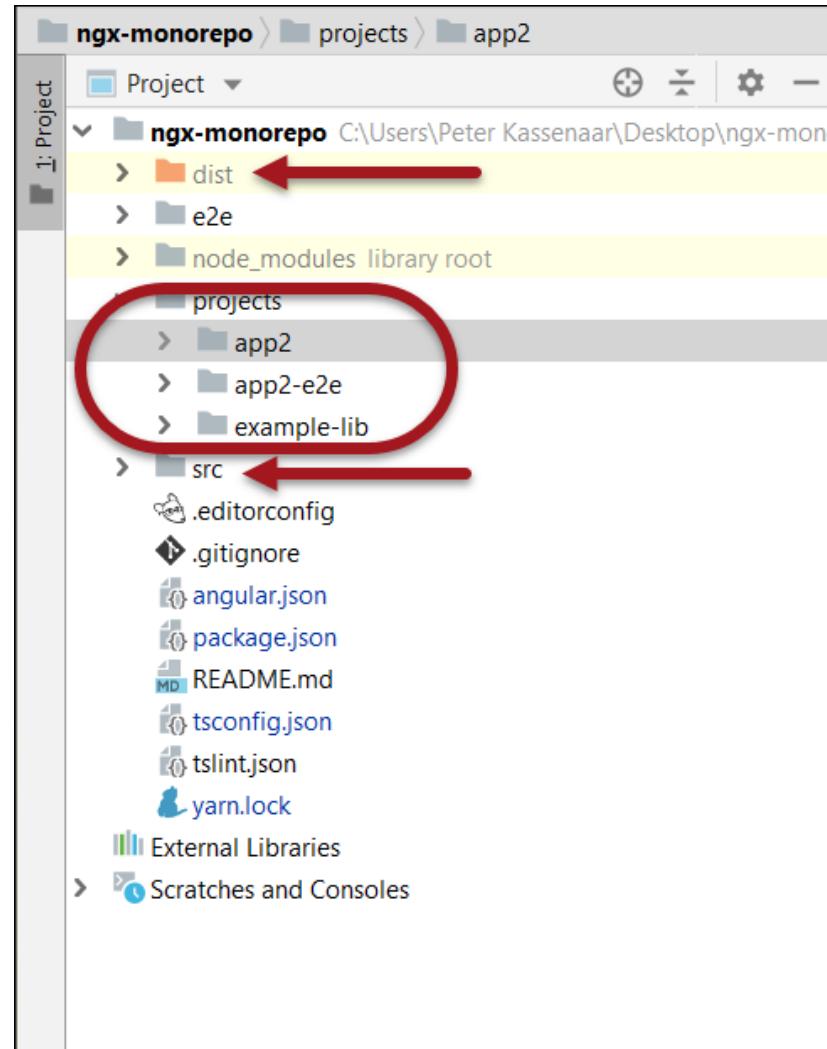
**ng generate**

Generate components, routes, services and pipes with a simple command. The CLI will also create simple test shells for all of these.

# Inside an Angular monorepo

- One `/node_modules`
- One main `package.json`
- `angular.json`, describing all the projects in the workspace
- Optional - One root app, in `/src` folder
- One `/projects` folder, containing:
  - A folder for every project
  - Libraries
  - Applications
  - This folder is created upon the first creation of a library or application
- `/dist` folder, holding the compiled Angular Packages that needs to be shared

# Structure/architecture



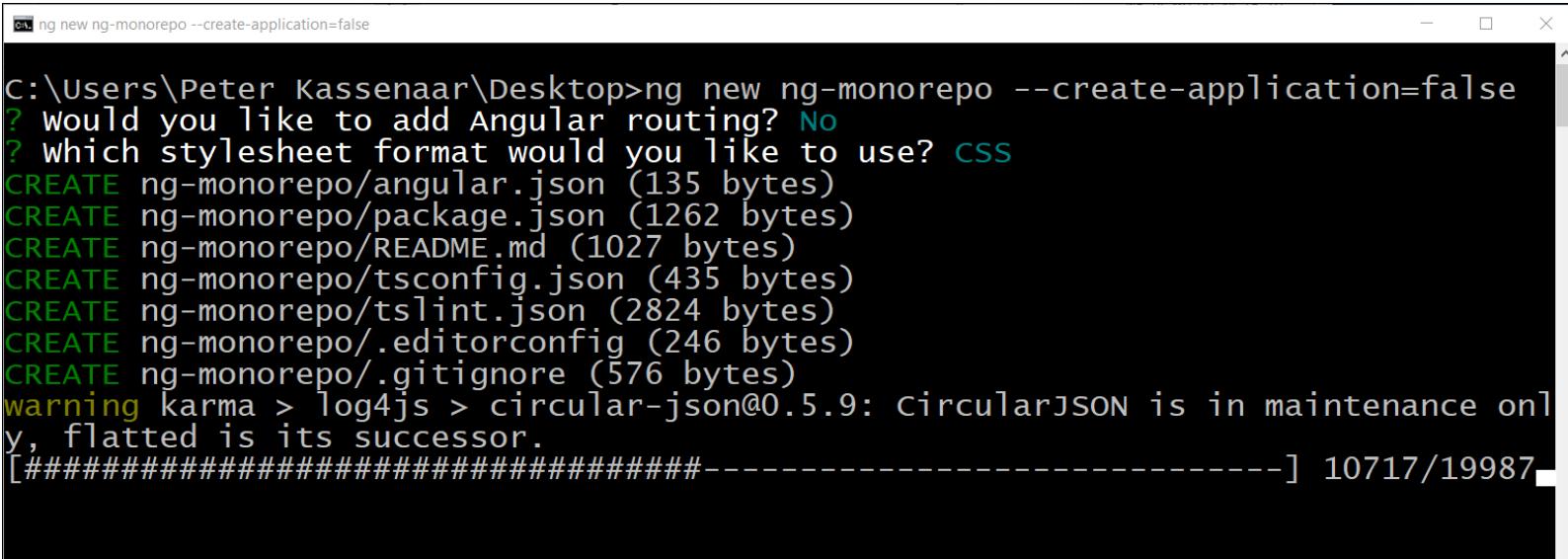
# Credits: Angular in Depth

The screenshot shows a blog post on the Angular In Depth website. The header features a red navigation bar with links for HOME, ANGULAR, RXJS, NGRX, ABOUT, SUPPORT US, and AG-GRID: THE BEST ANGULAR GRID IN THE WORLD. The main title of the post is "The Angular Library Series - Creating a Library with Angular CLI". Below the title is a subtitle: "Angular libraries just got a lot easier to create thanks to the combination of Angular CLI and ng-packagr.". The author's profile picture is a circular image of Todd Palmer, with his name and a "Follow" button next to it. The post was published on May 28, 2018, and has a reading time of 11 minutes.

<https://blog.angularindepth.com/creating-a-library-in-angular-6-87799552e7e5>

# High level overview of the steps

1. Create the library project using `ng new <mono-repo-name>`
  - We call this folder a *workspace*
2. The main app is mostly used for documentation, or example usage of the (shared) libs
  - Don't want the main app? Use `--create-application=false` flag!
  - <https://blog.angularindepth.com/angular-workspace-no-application-for-you-4b451afcc2ba>

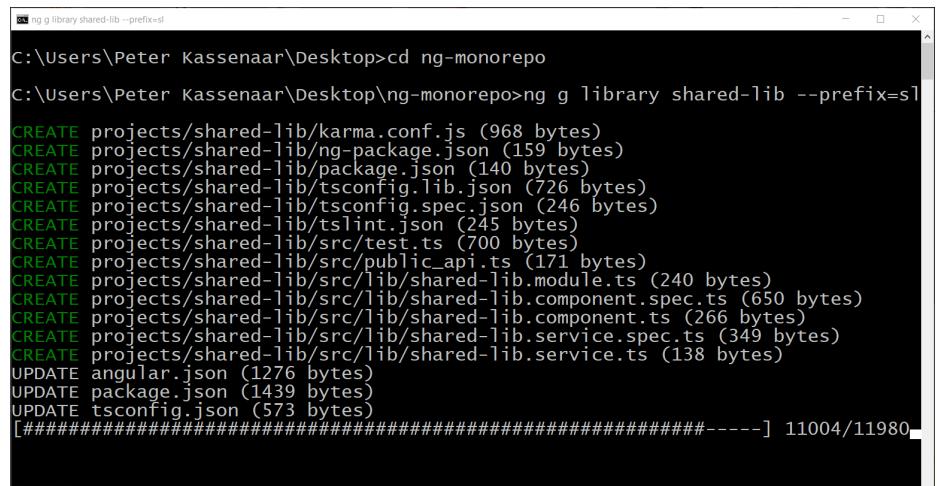


The screenshot shows a terminal window with the following command and its output:

```
PS C:\Users\Peter Kassenaar\Desktop>ng new ng-monorepo --create-application=false
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE ng-monorepo/angular.json (135 bytes)
CREATE ng-monorepo/package.json (1262 bytes)
CREATE ng-monorepo/README.md (1027 bytes)
CREATE ng-monorepo/tsconfig.json (435 bytes)
CREATE ng-monorepo/tslint.json (2824 bytes)
CREATE ng-monorepo/.editorconfig (246 bytes)
CREATE ng-monorepo/.gitignore (576 bytes)
warning karma > log4js > circular-json@0.5.9: circularJSON is in maintenance only, flattened is its successor.
[########################################-----] 10717/19987
```

# Create (shared) lib

- Generate the (shared) library
  - cd ng-monorepo
  - ng generate library shared-lib --prefix=sl (or some other prefix)
  - Angular also updates the global angular.json, package.json and tsconfig.json
- Always use custom prefixes on libraries and projects
  - distinguish components and services!



```
ng g library shared-lib --prefix=sl
C:\Users\Peter Kassenaar\Desktop>cd ng-monorepo
C:\Users\Peter Kassenaar\Desktop\ng-monorepo>ng g library shared-lib --prefix=sl
CREATE projects/shared-lib/karma.conf.js (968 bytes)
CREATE projects/shared-lib/ng-package.json (159 bytes)
CREATE projects/shared-lib/package.json (140 bytes)
CREATE projects/shared-lib/tsconfig.lib.json (726 bytes)
CREATE projects/shared-lib/tsconfig.spec.json (246 bytes)
CREATE projects/shared-lib/tslint.json (245 bytes)
CREATE projects/shared-lib/src/test.ts (700 bytes)
CREATE projects/shared-lib/src/public_api.ts (171 bytes)
CREATE projects/shared-lib/src/lib/shared-lib.module.ts (240 bytes)
CREATE projects/shared-lib/src/lib/shared-lib.component.spec.ts (650 bytes)
CREATE projects/shared-lib/src/lib/shared-lib.component.ts (266 bytes)
CREATE projects/shared-lib/src/lib/shared-lib.service.spec.ts (349 bytes)
CREATE projects/shared-lib/src/lib/shared-lib.service.ts (138 bytes)
UPDATE angular.json (1276 bytes)
UPDATE package.json (1439 bytes)
UPDATE tsconfig.json (573 bytes)
[########################################] 11004/11980
```

```
angular.json
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "shared-lib": {"root": "projects/shared-lib"...}
  },
  "defaultProject": "shared-lib"
}
```

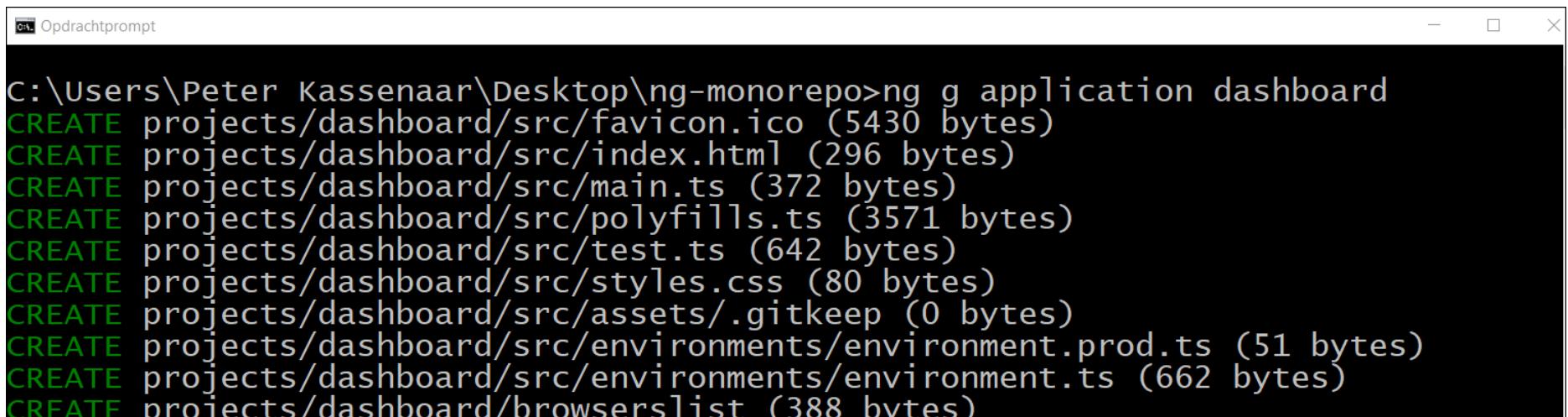
```

"lib": [
  "es2018",
  "dom"
],
"paths": {
  "shared-lib": [
    "dist/shared-lib"
  ],
  "shared-lib/*": [
    "dist/shared-lib/*"
  ]
}
```

tsconfig.json – added paths, so we can import shared stuff easily later on

# Create first app in the monorepo

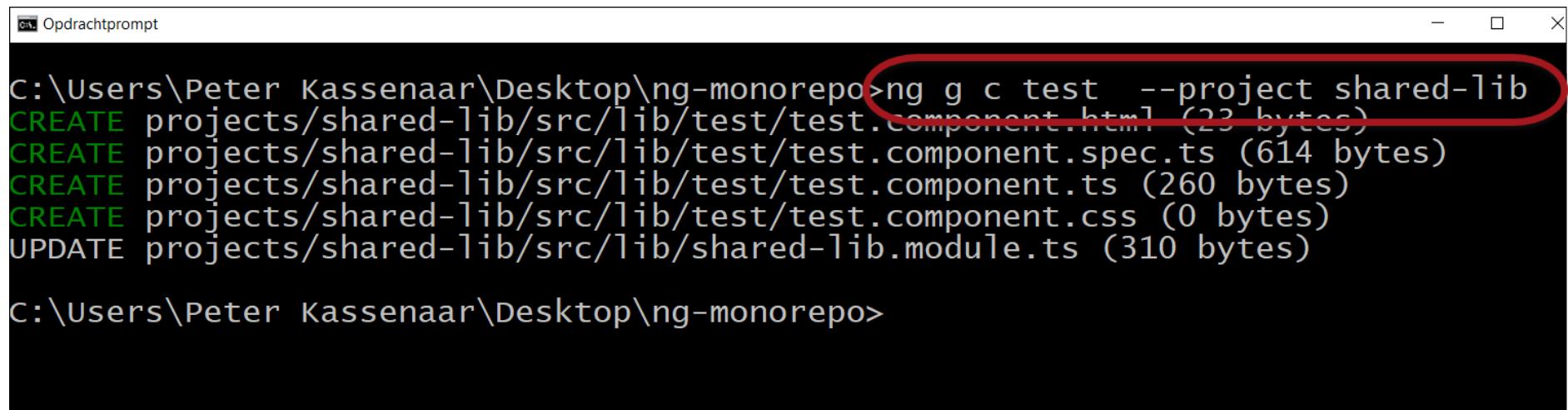
- Generate the (first) application
  - `ng generate application <application-name>`
  - Again, `angular.json` and `package.json` are updated with the new project



```
C:\Users\Peter Kassenaar\Desktop\ng-monorepo>ng g application dashboard
CREATE projects/dashboard/src/favicon.ico (5430 bytes)
CREATE projects/dashboard/src/index.html (296 bytes)
CREATE projects/dashboard/src/main.ts (372 bytes)
CREATE projects/dashboard/src/polyfills.ts (3571 bytes)
CREATE projects/dashboard/src/test.ts (642 bytes)
CREATE projects/dashboard/src/styles.css (80 bytes)
CREATE projects/dashboard/src/assets/.gitkeep (0 bytes)
CREATE projects/dashboard/src/environments/environment.prod.ts (51 bytes)
CREATE projects/dashboard/src/environments/environment.ts (662 bytes)
CREATE projects/dashboard/browserslist (388 bytes)
```

# Create a shared component

- Of course you can create multiple components
  - `ng generate component <component-name> --project shared-lib`
  - Use the `--project` flag to tell the CLI what project you want to add the component to
- Give the component some UI



```
C:\Users\Peter Kassenaar\Desktop\ng-monorepo>ng g c test --project shared-lib
CREATE projects/shared-lib/src/lib/test/test.component.html (23 bytes)
CREATE projects/shared-lib/src/lib/test/test.component.spec.ts (614 bytes)
CREATE projects/shared-lib/src/lib/test/test.component.ts (260 bytes)
CREATE projects/shared-lib/src/lib/test/test.component.css (0 bytes)
UPDATE projects/shared-lib/src/lib/shared-lib.module.ts (310 bytes)

C:\Users\Peter Kassenaar\Desktop\ng-monorepo>
```

# Export the component

- Add it to the exports array of the shared-lib.module.ts

```
exports: [..., TestComponent]
```

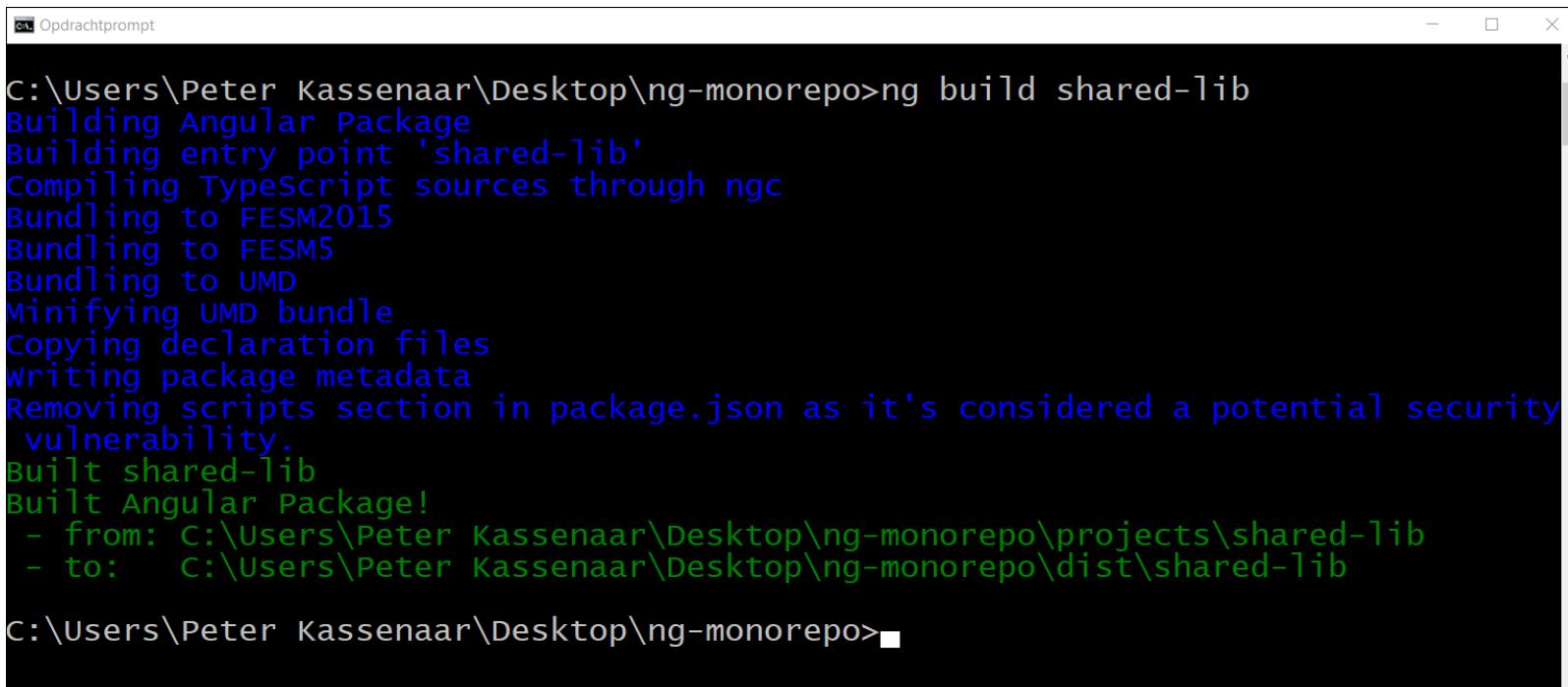
- Update the public\_api.ts file to make the class available
  - Don't forget!

```
/*
 * Public API Surface of shared-lib
 */

export * from './lib/shared-lib.service';
export * from './lib/shared-lib.component';
export * from './lib/shared-lib.module';
export * from './lib/test/test.component';
```

# Build the library

- In order to use the component(s) from a shared library, it must be build
  - `ng build shared-lib`
  - A \dist folder is created
  - This folder is already mentioned in `tsconfig.json`. Verify this!



```
C:\Users\Peter Kassenaar\Desktop\ng-monorepo>ng build shared-lib
Building Angular Package
Building entry point 'shared-lib'
Compiling TypeScript sources through ngc
Bundling to FESM2015
Bundling to FESM5
Bundling to UMD
Minifying UMD bundle
Copying declaration files
Writing package metadata
Removing scripts section in package.json as it's considered a potential security
vulnerability.
Built shared-lib
Built Angular Package!
- from: C:\Users\Peter Kassenaar\Desktop\ng-monorepo\projects\shared-lib
- to:   C:\Users\Peter Kassenaar\Desktop\ng-monorepo\dist\shared-lib

C:\Users\Peter Kassenaar\Desktop\ng-monorepo>
```

# Using the library in the application

- Import the library module in the application `app.module.ts`
  - in our example: `dashboard.module.ts`
  - Remove the path your IDE might add to the AutoImport

The diagram illustrates the correct way to import a library module in an Angular application module. It compares two versions of the `app.module.ts` file.

**Left Snippet (Incorrect):**

```
import { AppComponent } from './app.component';
import {SharedLibModule} from '../../../../../shared-lib/src/lib/shared-lib.module';

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

A red arrow points to the incorrect import path `'../../../../shared-lib/src/lib/shared-lib.module'`, which is highlighted with a red X.

**Right Snippet (Correct):**

```
import { AppComponent } from './app.component';
import {SharedLibModule} from 'shared-lib';

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

A red arrow points to the correct import path `'shared-lib'`, which is highlighted with a green checkmark.

# Using the shared component

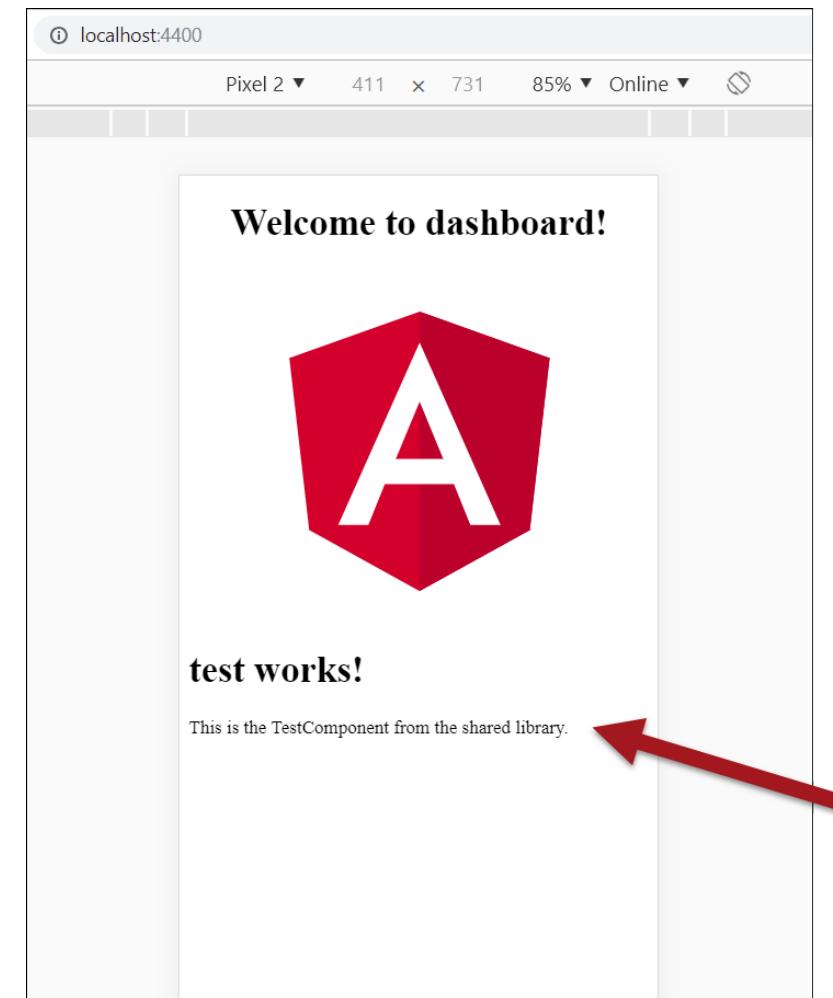
- Use the selector of the component from the shared library as normal

```
<div style="text-align:center">  
  ...  
</div>  
<sl-test></sl-test>
```

# Running the application

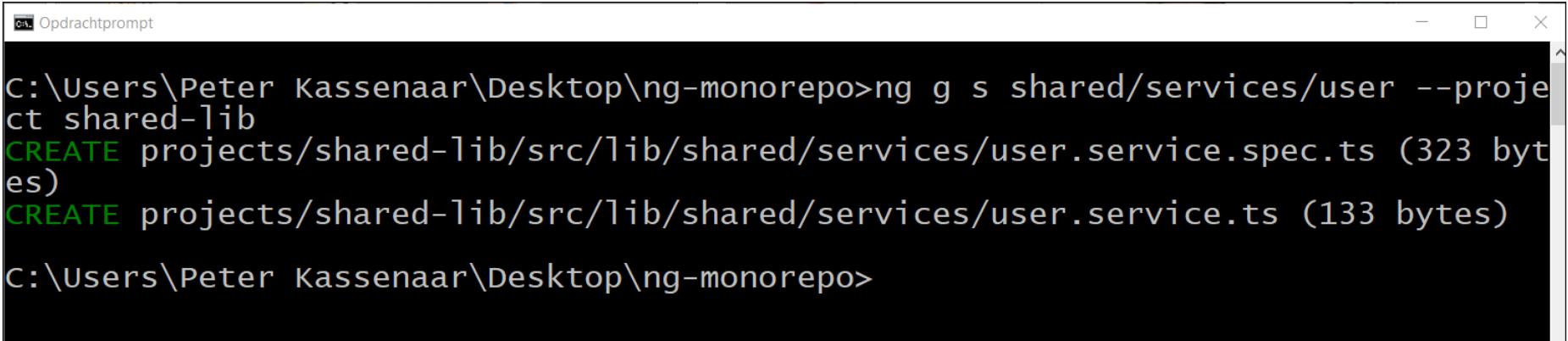
- Use the `ng serve` command
  - Use the `--project=<project-name>` to open the correct project
  - You can use additional flags as needed
- If you update your library contents, you need to rebuild it!
  - Write a script for that. For example

```
"scripts": {  
  ...  
  "build_lib": "ng build shared-lib"  
},
```



# Exporting a service

- Create a service the regular way
  - `ng generate service <service-name> --project=<project-name>`
- You need to make sure to export your service from the module
- But it only needs to be loaded once!
  - Use a `.forRoot()` method on the module



```
c:\Users\Peter Kassenaar\Desktop\ng-monorepo>ng g s shared/services/user --project shared-lib
CREATE projects/shared-lib/src/lib/shared/services/user.service.spec.ts (323 bytes)
CREATE projects/shared-lib/src/lib/shared/services/user.service.ts (133 bytes)
c:\Users\Peter Kassenaar\Desktop\ng-monorepo>
```

# Creating a .forRoot()

```
@NgModule({
  declarations: [SharedLibComponent, TestComponent],
  imports: [
  ],
  exports: [SharedLibComponent, TestComponent]
})
export class SharedLibModule {
  static forRoot(): ModuleWithProviders {
    return {
      ngModule: SharedLibModule,
      providers: [ UserService ]
    };
  }
}
```

- Remember to export the service from `public_api.ts`
- Remember to rebuild the library

# Using the shared service

- In your application, update the module to use  
`SharedLibModule.forRoot()`
- Inject the Service in the component where you want  
to use it
- Use the servicemethods as normal

```
@NgModule({
  ...
  imports: [
    SharedLibModule.forRoot()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# Component

```
import {Component, OnInit} from '@angular/core';
import {User, UserService} from 'shared-lib';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  title = 'dashboard';
  users: User[];

  constructor(private userService: UserService) {}

  ngOnInit(): void {
    this.users = this.userService.getUsers();
  }
}
```



```
<hr>
<h2>Users from our shared service</h2>
{{ users | json }}
```



**test works!**

This is the TestComponent from the shared library.

---

### Users from our shared service

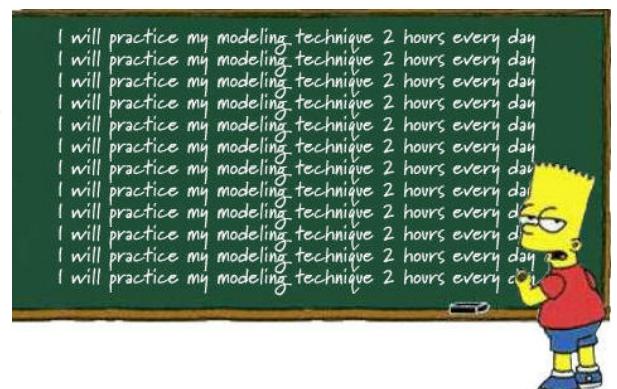
```
[ { "name": "Peter", "email": "test@test.com" }, { "name": "Sandra", "email": "sandra@sandra.com" } ]
```



# Workshop

- Create a new application inside the Monorepo
- Import the shared library
- Use/show the <sl-test> component from the shared lib inside your new project
- Build and run the new project
- Optional: create a new shared component in the lib.
  - Export / use the new component inside your project
  - Use the exported shared service from the library

Example: [github.com/PeterKassenaar/ng-monorepo](https://github.com/PeterKassenaar/ng-monorepo)



# Shorten TypeScript imports

The screenshot shows a web page from [AngularFirebase](#). At the top right, there are buttons for **PRO**, **Search**, and **Lessons**. On the left sidebar, under **Contents**, there are links to **The Problem - Super Long Import Statements**, **The Solution - TypeScript Config**, and **The End**. Under **Recent Posts**, there is a link to [Object Oriented Programming With TypeScript](#). The main content area has a heading **The Problem - Super Long Import Statements** (circled in red). It contains text about deep nesting and code snippets showing long import statements. Below this, there's a section titled **The Solution - TypeScript Config** with code for `tsconfig.json`.

**The Problem - Super Long Import Statements**

Let's arbitrarily create a deeply nested component in Angular and a service to go along with it.

```
ng g component shared/deeply/nested/hello-world  
ng g service core/my
```

When you have a large Angular project with deeply nested files and folders it leads to import statements that look like this...

```
// import from component.ts  
import { MyService } from '../../../../../my.service';  
  
// import from service.ts  
import { HelloWorldComponent } from '../shared/deeply/nested/hello-world/hello-world.component';
```

That's might be fine occasionally, but it becomes very annoying and difficult to maintain for each new component you build. Furthermore, moving a file will require

**The Solution - TypeScript Config**

Fortunately, we can configure TypeScript to make our files behave more like which should be the `src` directory in an Angular CLI project.

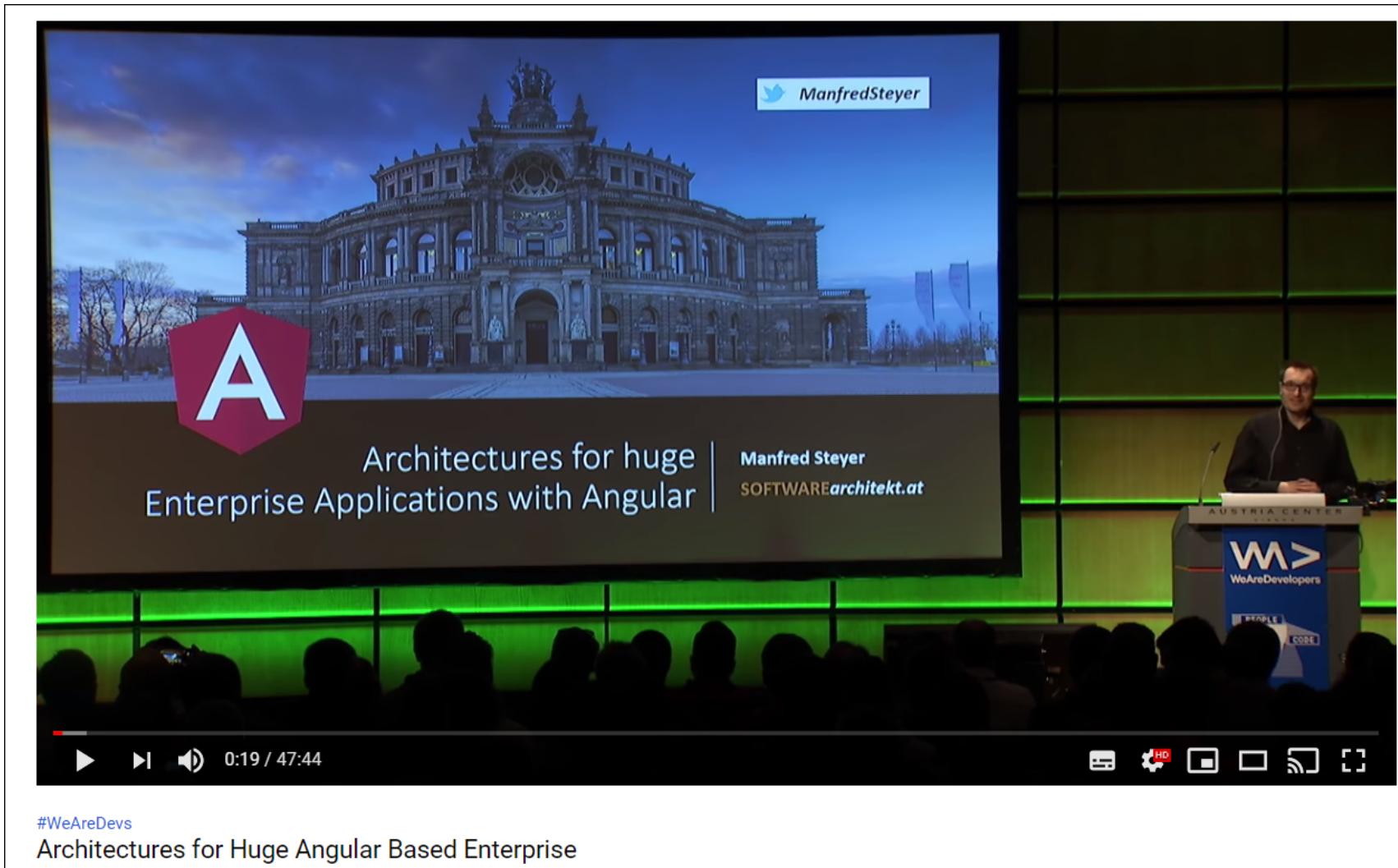
You can then point to any directory in your project and give it a custom name `@angular`, `@ngrx`, etc. The end result looks like...

```
// tsconfig.json in the root dir  
  
{  
  "compileOnSave": false,  
  "compilerOptions": {  
    // omitted...  
  
    "baseUrl": "src",  
    "paths": {  
      "@services/*": ["app/path/to/services/*"],  
      "@components/*": ["app/somewhere/deeply/nested/*"],  
      "@environments/*": ["environments/*"]  
    }  
  }  
}
```

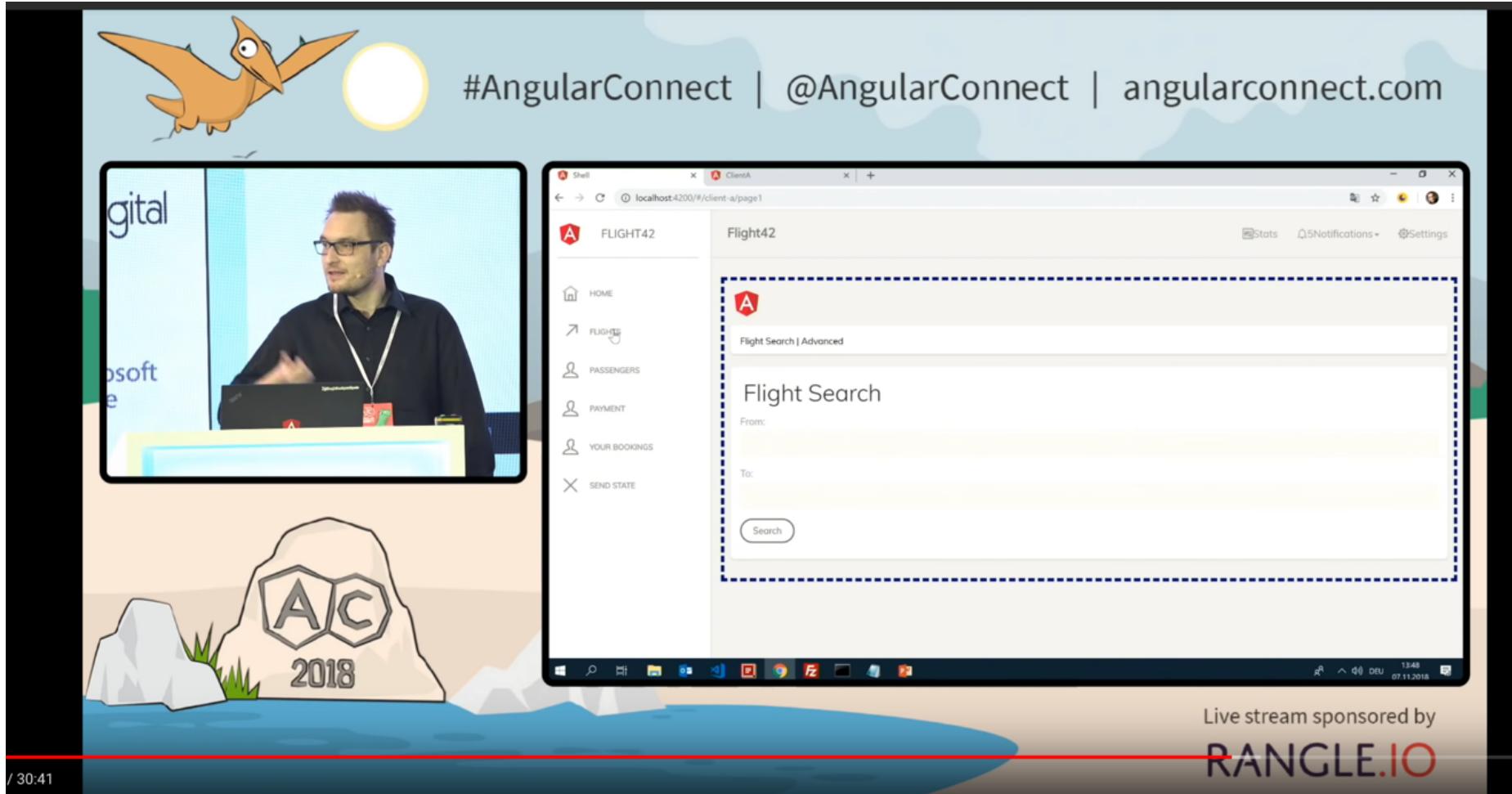
<https://angularfirebase.com/lessons/shorten-typescript-imports-in-an-angular-project/>

# Talks on Angular Monorepo's

- [https://www.youtube.com/watch?v=q4XmAy6\\_ucw](https://www.youtube.com/watch?v=q4XmAy6_ucw)



# Manfred Steyer – Angular Connect



<https://www.youtube.com/watch?v=YU-fMRs-ZYU>

Code: <https://github.com/PeterKassenaar/angular-microapp>

# Victor Savkin – creator of Nx

- <https://www.youtube.com/watch?v=piQ0EZhtus0>

The image shows a YouTube video player interface. At the top left is the Angular logo (a red hexagon with a white 'A'). Below it is the text 'Before' followed by two bullet points: '• Googler on Angular team' and '• Blogged at [vsavkin.com](http://vsavkin.com)'. To the right is the Narwhal Technologies Inc. logo (a blue narwhal head next to the text 'Nrwli') and the text 'Now' followed by two bullet points: '• Co-Founder at [nrwl.io](http://nrwl.io)' and '• Blog at [blog.nrwl.io](http://blog.nrwl.io)'. On the right side of the video player is a vertical thumbnail for a video titled 'NG CONF' featuring a man speaking at a podium. The main video player area displays the title 'ANGULAR AT LARGE ORGANIZATIONS' and the speaker's name 'Victor Savkin'. The bottom of the video player shows standard YouTube controls and a timestamp of '0:14 / 25:28'.

Angular at Large Organizations - Victor Savkin

# Publishing your library to npm

The screenshot shows a blog post on the Angular In Depth website. The header features a red bar with the site's logo and navigation links for 'ABOUT', 'SUPPORT US', and 'AG-GRID: THE BEST ANGULAR GRID IN THE WORLD'. The main title is 'The Angular Library Series — Publishing', with a subtitle 'Publishing your Angular Library to npm'. Below the author information (Todd Palmer, Aug 29, 2018) is a large, ornate photograph of a grand library interior with multiple levels of bookshelves and detailed architectural columns.

<https://blog.angularindepth.com/the-angular-library-series-publishing-ce24bb673275>

## More info

- <https://blog.angularindepth.com/creating-a-library-in-angular-6-87799552e7e5>
- <https://blog.angularindepth.com/creating-a-library-in-angular-6-part-2-6e2bc1e14121>
- <https://blog.angularindepth.com/angular-workspace-no-application-for-you-4b451afcc2ba>