



Angular Advanced Module - Monorepo's



Peter Kassenaar
info@kassenaar.com

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/ng-microfrontends>

Manfred Steyer – Angular Connect

The screenshot shows a live stream presentation. On the left, a video inset shows a man (Manfred Steyer) speaking at a podium. The main background features a light blue sky with a yellow sun, a cartoon orange pterosaur, and the text "#AngularConnect | @AngularConnect | angularconnect.com". Below this, a large browser window displays a web application titled "Flight42". The application has a sidebar menu with options: HOME, FLIGHTS, PASSENGERS, PAYMENT, YOUR BOOKINGS, and SEND STATE. The main content area is titled "Flight Search | Advanced" and "Flight Search", with input fields for "From:" and "To:" and a "Search" button. A dashed blue box highlights the search area. At the bottom right, it says "Live stream sponsored by RANGLE.IO". In the bottom left corner, there is a timestamp "/ 30:41".

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

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

Enterprise applications – multiple options

- There are always *multiple solutions*
- There is **NOT** one solution that is 'the best'
- Options:
 1. **NPM packages**

Publish your own packages/libraries to `npm`, so others can `npm install` them
 2. **Monorepo**

Multiple projects in one code base, optionally sharing code
 3. **Micro-apps / micro-frontends**

Multiple applications, not sharing code, optionally different techniques/frameworks

So basically...

**NPM
packages**

Monorepo

Micro apps



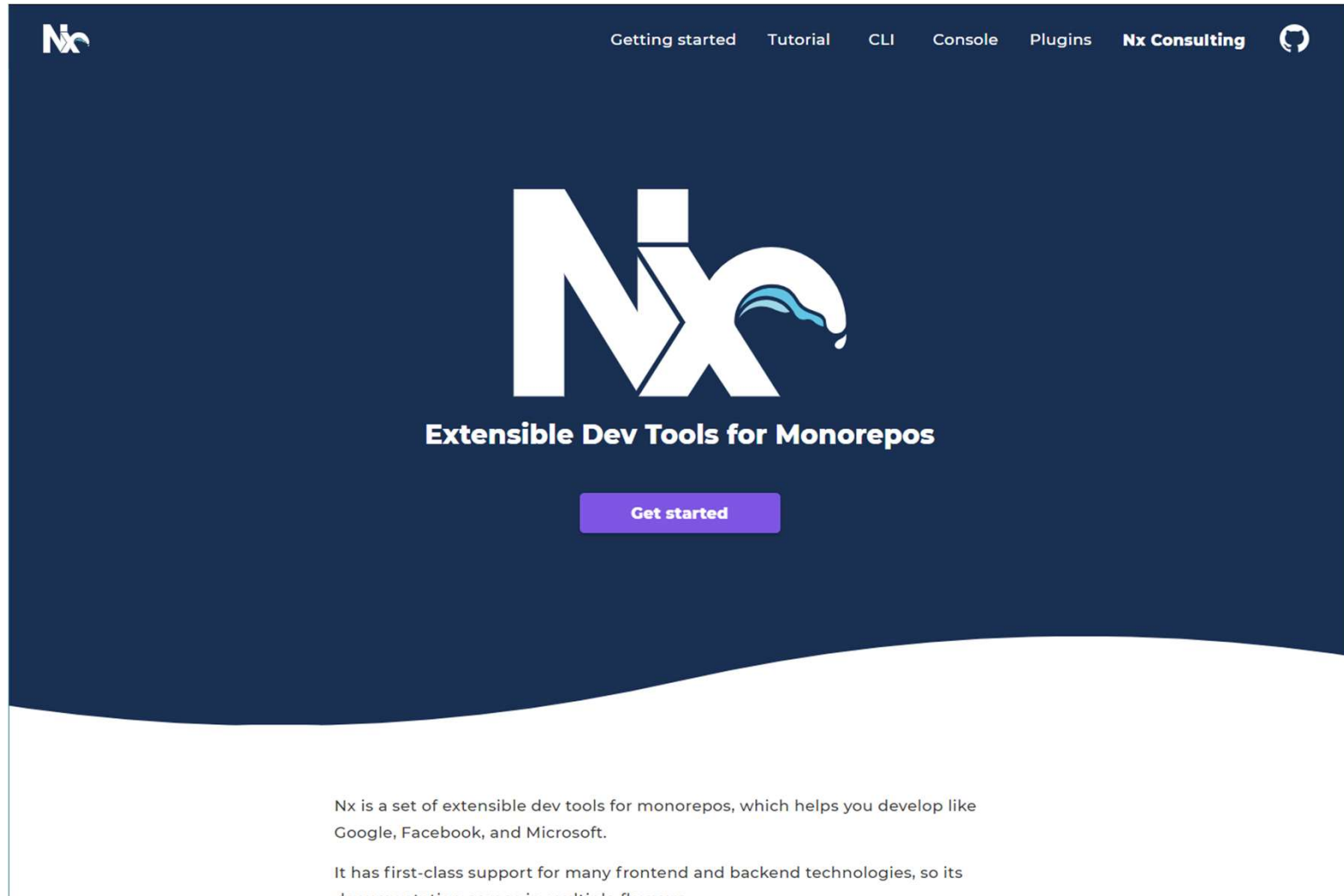
Monorepos

Multiple projects in one solution, optionally sharing code

What is a monorepo

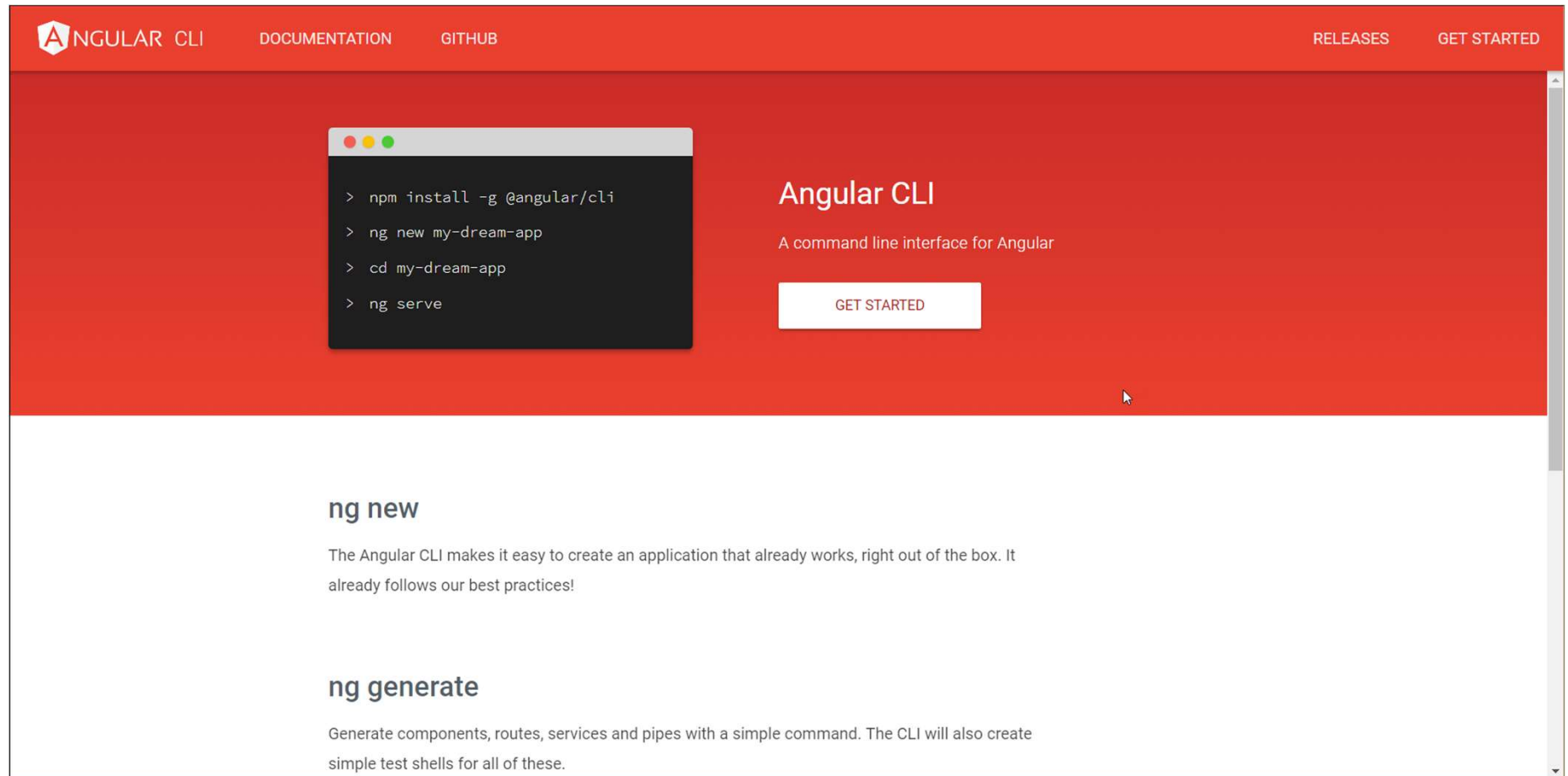
- Often, bigger applications are not only split up into **modules**, but also in **other applications**
- Applications generally share a lot of code
 - Sharing components
 - Sharing services
 - Sharing pipes, other logic, etc...
- One (opiniated) solution – create a so called *monorepo*
- There are tools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

Nrwl extensions to create a monorepo



<https://nx.dev/>

Angular CLI – as of V6.0+



Which one to use?

- This is highly *opiniated*
- The work (roughly) the same
 - Creating a **workspace**
 - Creating **applications** and **libraries** inside that workspace
- Nx is also available for React
- Angular CLI is limited to (duh...) Angular
 - But...Nx is a wrapper around Angular CLI

mostly - Personal Preference!

We'll be using Angular CLI in this demo

The screenshot shows the Angular CLI documentation page for the `ng generate` command. The page is titled "CLI" and "ng generate". It describes the command as "Generates and/or modifies files based on a schematic." and provides the command syntax: `ng generate <schematic> [options]` and `ng g <schematic> [options]`.

The "Arguments" section contains a table with the following columns: ARGUMENT, DESCRIPTION, and VALUE TYPE.

ARGUMENT	DESCRIPTION	VALUE TYPE
<code><schematic></code>	The schematic or collection:schematic to generate. This option can take one of the following sub-commands:	string

- `app-shell`
- `application`
- `class`
- `component`
- `directive`

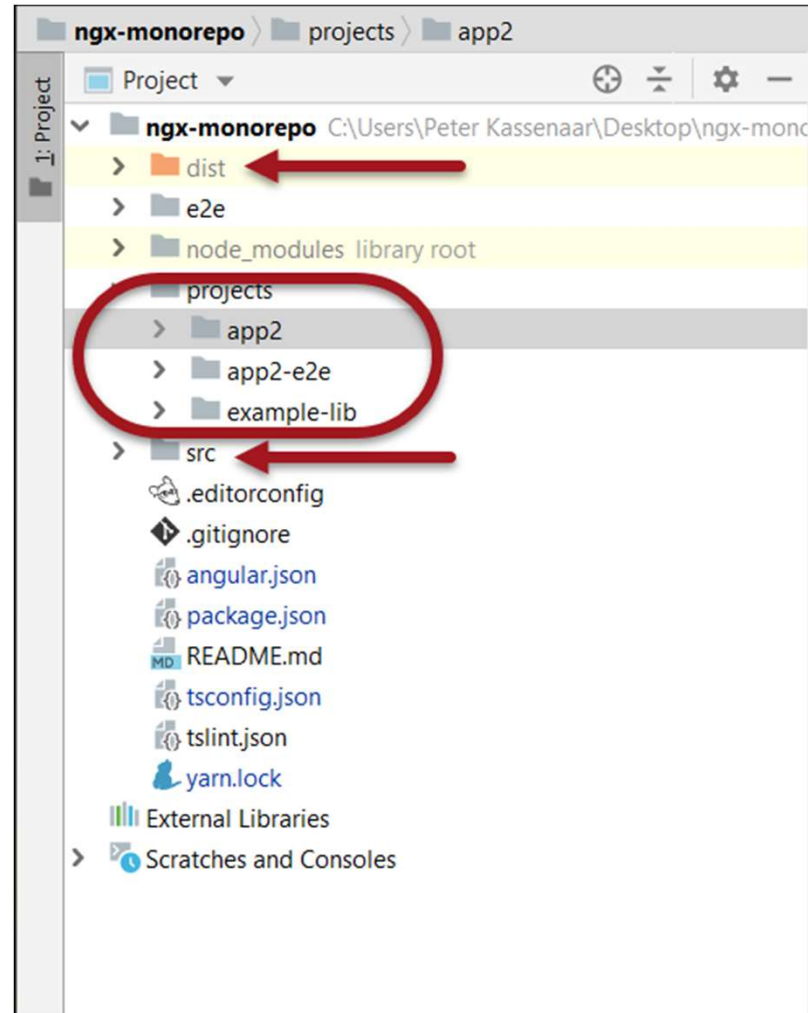
A red arrow points to the `application` sub-command in the list.

The right sidebar shows a list of sub-commands: `ng generate`, `Arguments`, `Options`, `Schematic commands`, `app-shell`, `application`, `class`, `component`, `directive`, `enum`, `guard`, `interceptor`, `interface`, `library`, `module`, `pipe`, `resolver`, `service`, and `service-worker`.

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



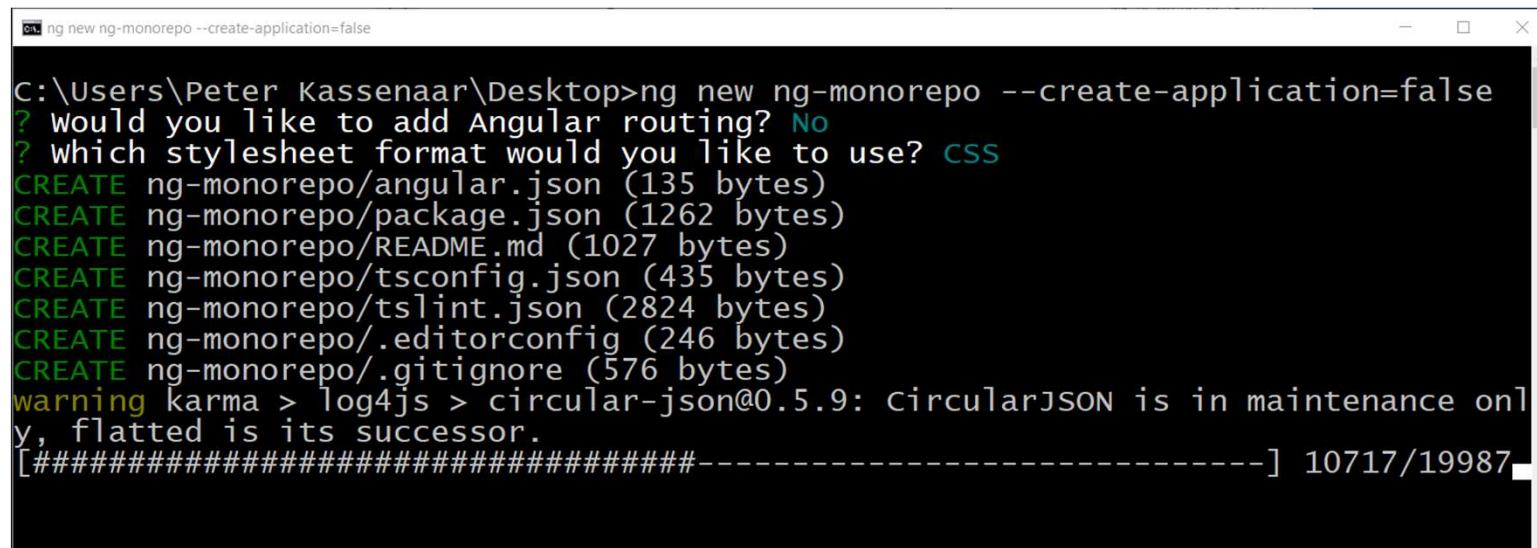
Credits: Angular in Depth



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

High level overview of the steps

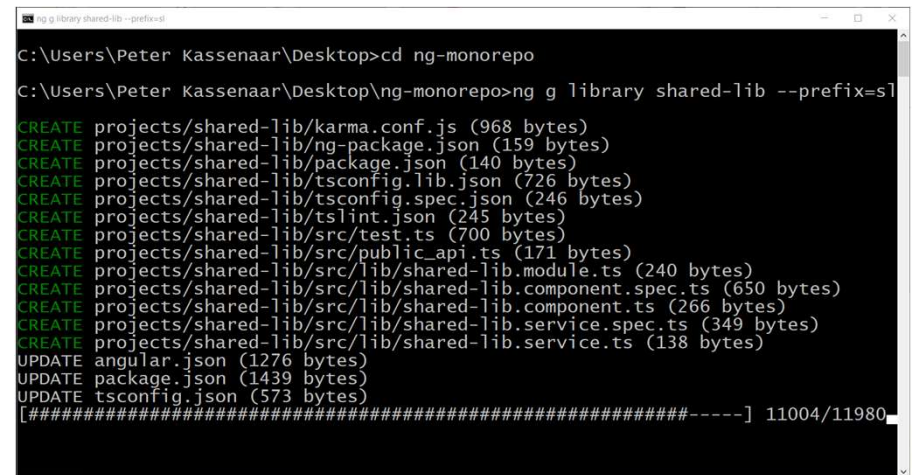
1. Create the container monorepo using `ng new <mono-repo-name>`
 - We call this folder a *workspace*
2. Create *without* the main app: `ng new <name> --create-application=false!`
 - <https://blog.angularindepth.com/angular-workspace-no-application-for-you-4b451afcc2ba>



```
ng new ng-monorepo --create-application=false
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!



```
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
```



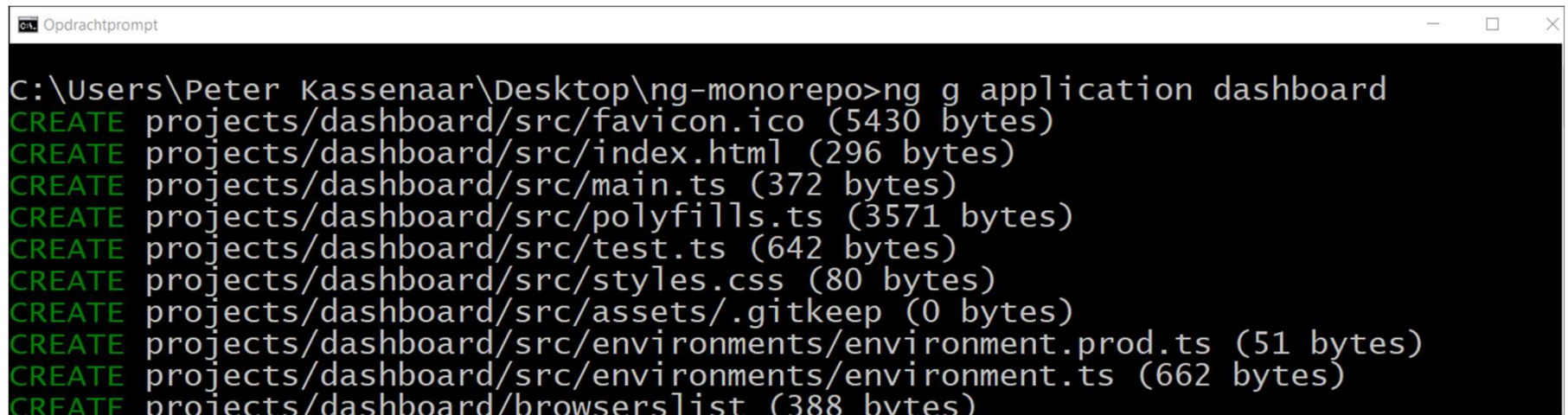
```
angular.json x
1 {
2   "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
3   "version": 1,
4   "newProjectRoot": "projects",
5   "projects": {
6     "shared-lib": {"root": "projects/shared-lib"...}
41  },
42   "defaultProject": "shared-lib"
43 }
```

```
17 "lib": [
18   "es2018",
19   "dom"
20 ],
21 "paths": {
22   "shared-lib": [
23     "dist/shared-lib"
24   ],
25   "shared-lib/*": [
26     "dist/shared-lib/*"
27   ]
28 }
29 }
30 }
```

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

Create first app in the monorepo

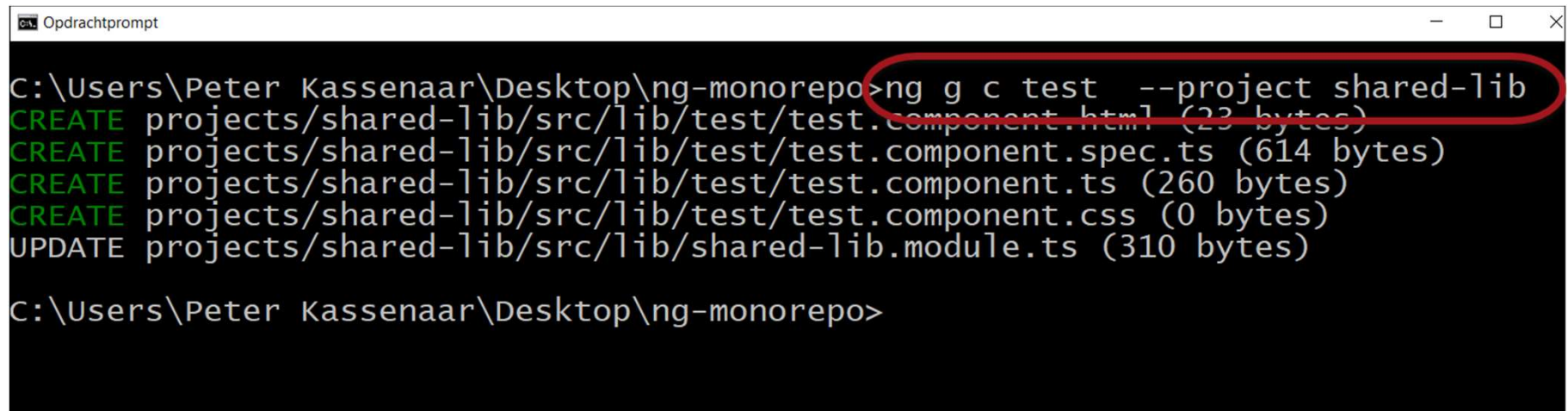
- Generate the (first) application
 - `ng generate application <application-name>`
 - Again, `angular.json` and `package.json` are updated with the new project



```
Opdrachtprompt
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



```
Opdrachtprompt
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`

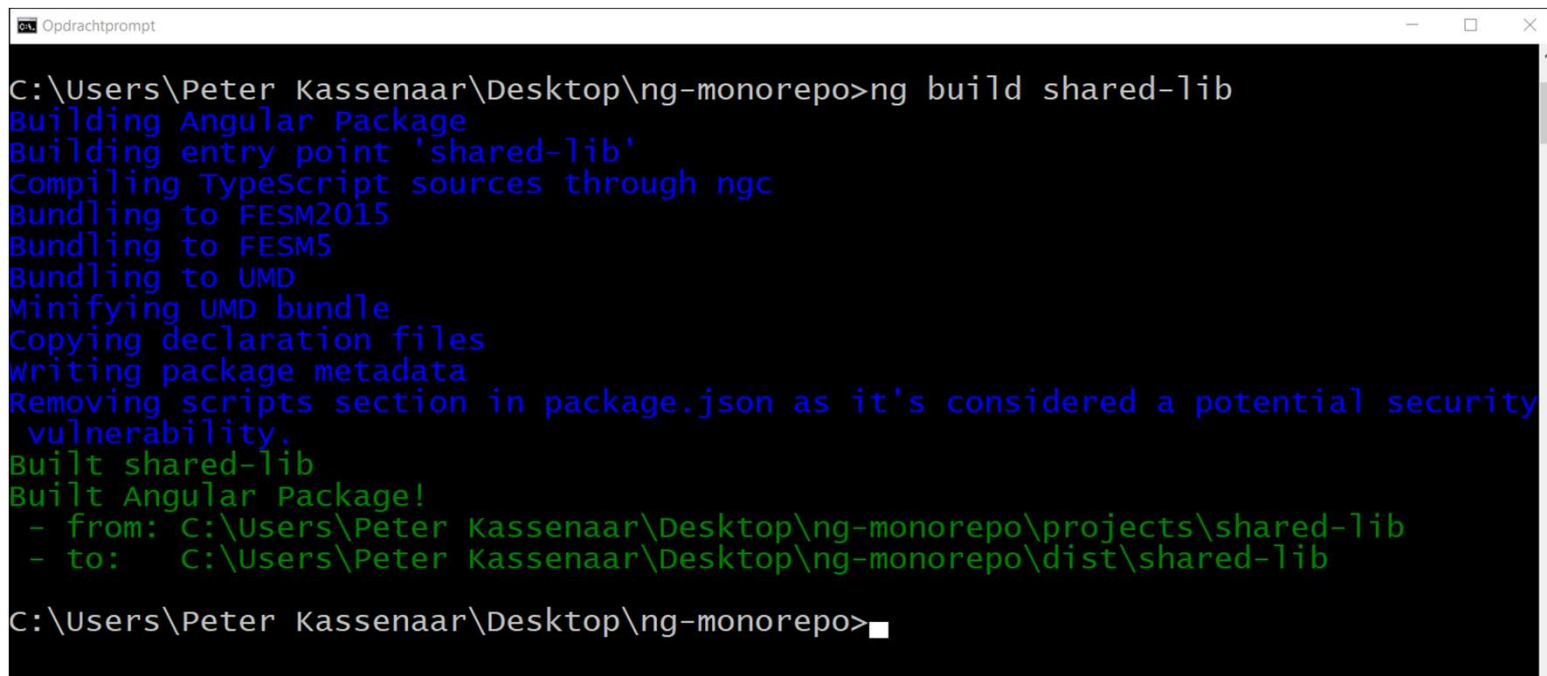
```
exports: [..., TestComponent]
```

- 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!



```
Opdrachtprompt
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

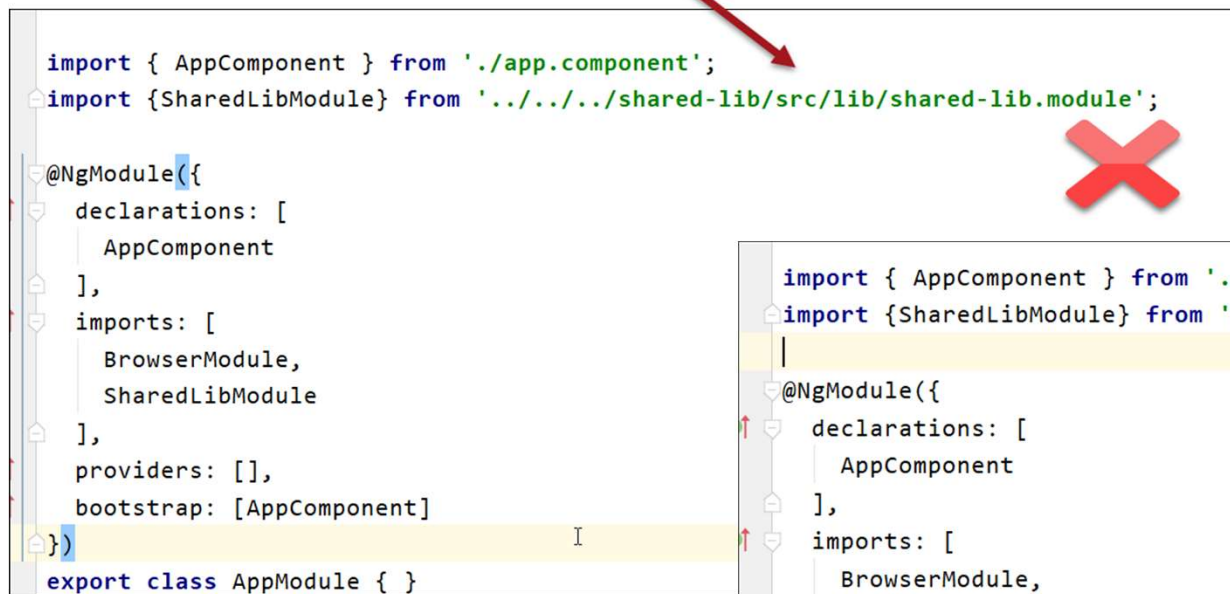


Diagram illustrating the incorrect import path for the library module. A red arrow points to the import statement, and a large red 'X' indicates it is wrong.

```
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 { }
```

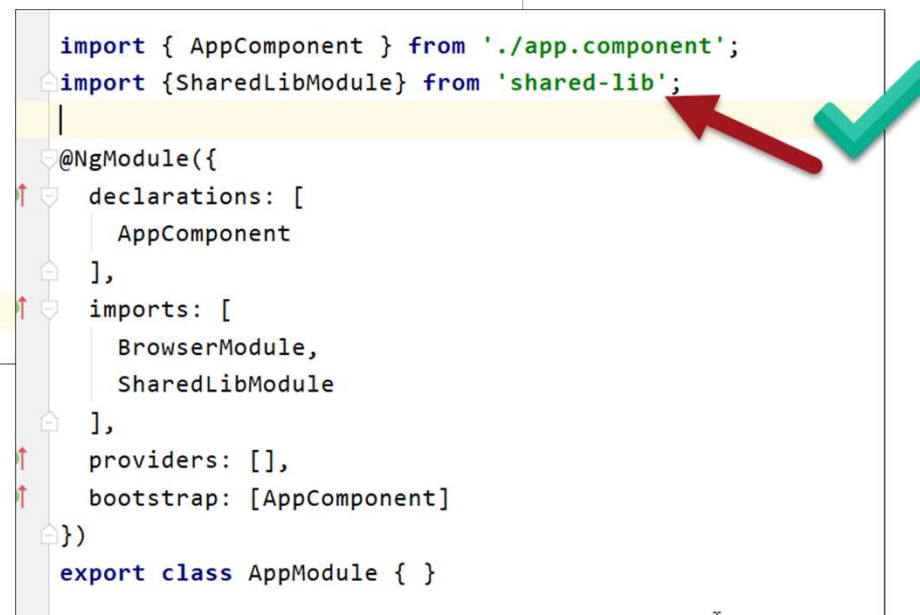


Diagram illustrating the correct import path for the library module. A red arrow points to the import statement, and a large green checkmark indicates it is correct.

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

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

Using the shared component

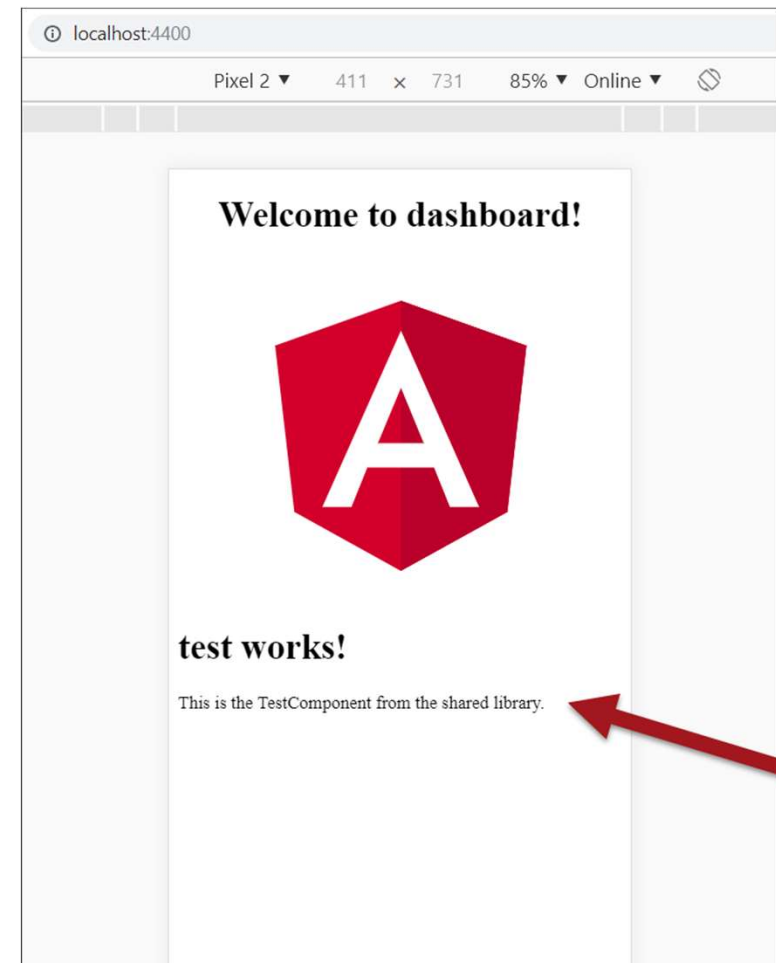
- Use the selector of the component from the shared library as normal

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

Running the application

- Use the `ng serve` command
 - Use the `--project=<application-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



```
Opdrachtprompt
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';

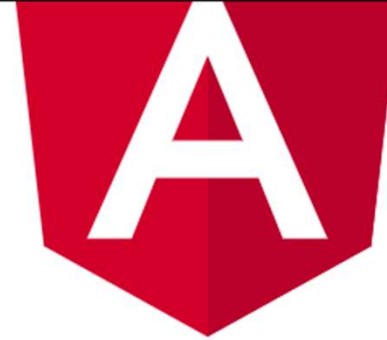
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  title = 'dashboard';
  users: User[];

  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

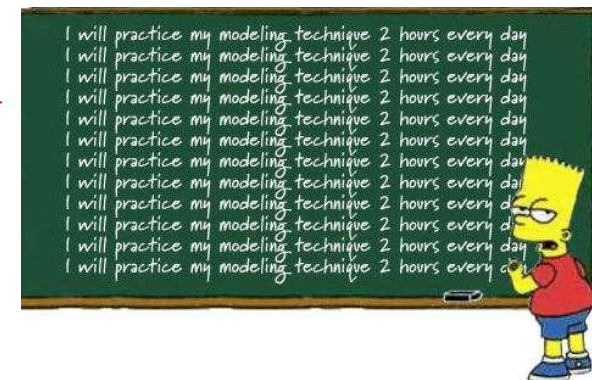
[{ "name": "Peter", "email": "test@test.com" }, { "name":
"Sandra", "email": "sandra@sandra.com" }]



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



Verdict on monorepo

- PRO:
 - Same version of Angular
 - Clear responsibilities per project/application
 - Share code and share services
- CON
 - Harder to use shared state / store
 - All the pro's, are actually also cons! (depending on your project)
 - Checkout: you always get the complete monorepo- unless working with *Github subtrees*



More info

Info on npm packages, monorepo's and micro apps


Info on...

**NPM
packages**

Monorepo

Micro apps

Talks on Angular Monorepo's



The screenshot shows a YouTube video player. The main content is a presentation slide with a background image of the Vienna State Opera. On the left of the slide is the Angular logo (a red shield with a white 'A'). The text on the slide reads: "Architectures for huge Enterprise Applications with Angular". To the right of this text, it says "Manfred Steyer" and "SOFTWAREarchitekt.at". In the top right corner of the slide, there is a Twitter icon and the name "ManfredSteyer". To the right of the slide, a man (Manfred Steyer) is standing at a podium, speaking into a microphone. The podium has a logo that says "W> WeAreDevelopers". The video player interface at the bottom shows a progress bar at 0:19 / 47:44, a volume icon, and several control icons. Below the video player, there is a hashtag "#WeAreDevs" and the text "Architectures for Huge Angular Based Enterprise".

#WeAreDevs
Architectures for Huge Angular Based Enterprise

- https://www.youtube.com/watch?v=q4XmAy6_ucw

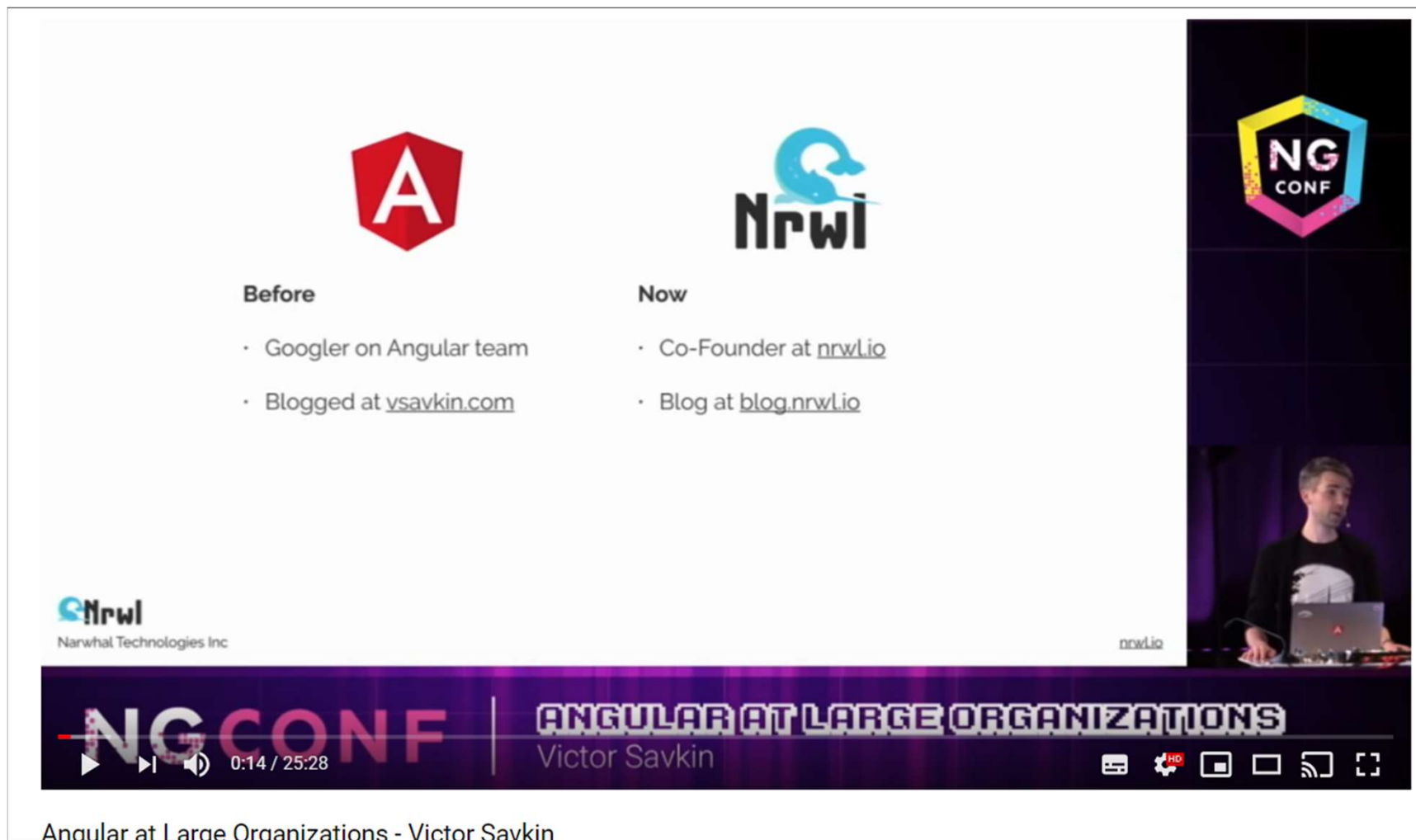
Manfred Steyer – Angular Connect

The screenshot shows a live stream presentation. On the left, a video inset shows a man with glasses and a dark shirt speaking at a podium. The background of the stream features a cartoon pterosaur, a sun, and the text "#AngularConnect | @AngularConnect | angularconnect.com". Below the video is a rock with the "AC 2018" logo. On the right, a browser window displays a web application titled "Flight42". The application has a sidebar menu with options: HOME, FLIGHTS, PASSENGERS, PAYMENT, YOUR BOOKINGS, and SEND STATE. The main content area is titled "Flight Search | Advanced" and contains a "Flight Search" form with "From:" and "To:" input fields and a "Search" button. The browser's address bar shows "localhost:4200/#/client-a/page1". At the bottom right, text indicates "Live stream sponsored by RANGLE.IO". A timestamp "30:41" is visible in the bottom left corner.

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

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

Victor Savkin – creator of Nx



The screenshot shows a video player interface. The main content is a presentation slide with two columns: 'Before' and 'Now'. The 'Before' column features the Angular logo and lists 'Googler on Angular team' and 'Blogged at vsavkin.com'. The 'Now' column features the Nx logo and lists 'Co-Founder at nrwl.io' and 'Blog at blog.nrwl.io'. The slide also includes the Nx logo and 'Narwhal Technologies Inc.' at the bottom left. The video player has a purple and black theme with 'NG CONF' and 'ANGULAR AT LARGE ORGANIZATIONS' in the background. The title 'ANGULAR AT LARGE ORGANIZATIONS' and the speaker's name 'Victor Savkin' are displayed at the bottom. The video progress bar shows 0:14 / 25:28. The video player controls include play, pause, volume, and full screen buttons.

Before

- Googler on Angular team
- Blogged at vsavkin.com

Now

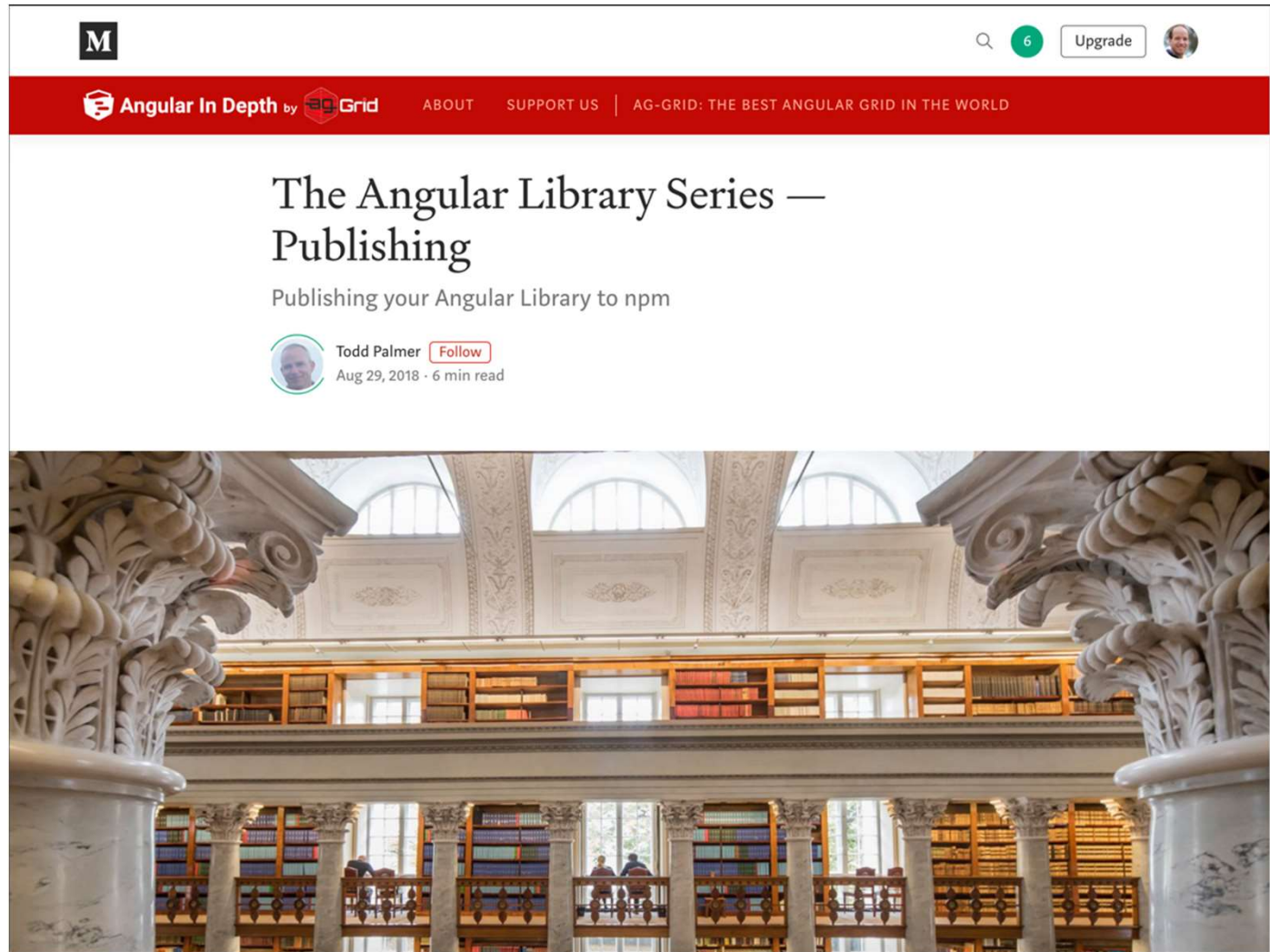
- Co-Founder at nrwl.io
- Blog at blog.nrwl.io

NG CONF | **ANGULAR AT LARGE ORGANIZATIONS**
Victor Savkin

Angular at Large Organizations - Victor Savkin

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

Publishing your library to npm



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

Implementing micro apps in Angular



<https://medium.com/bb-tutorials-and-thoughts/how-to-implement-micro-frontend-architecture-with-angular-e6828a0a049c>