Angular Advanced
01 - Introduction

Peter Kassenaar –
info@kassenaar.com

github.com/PeterKassenaar/oce

`…/slides/advanced`

About you…

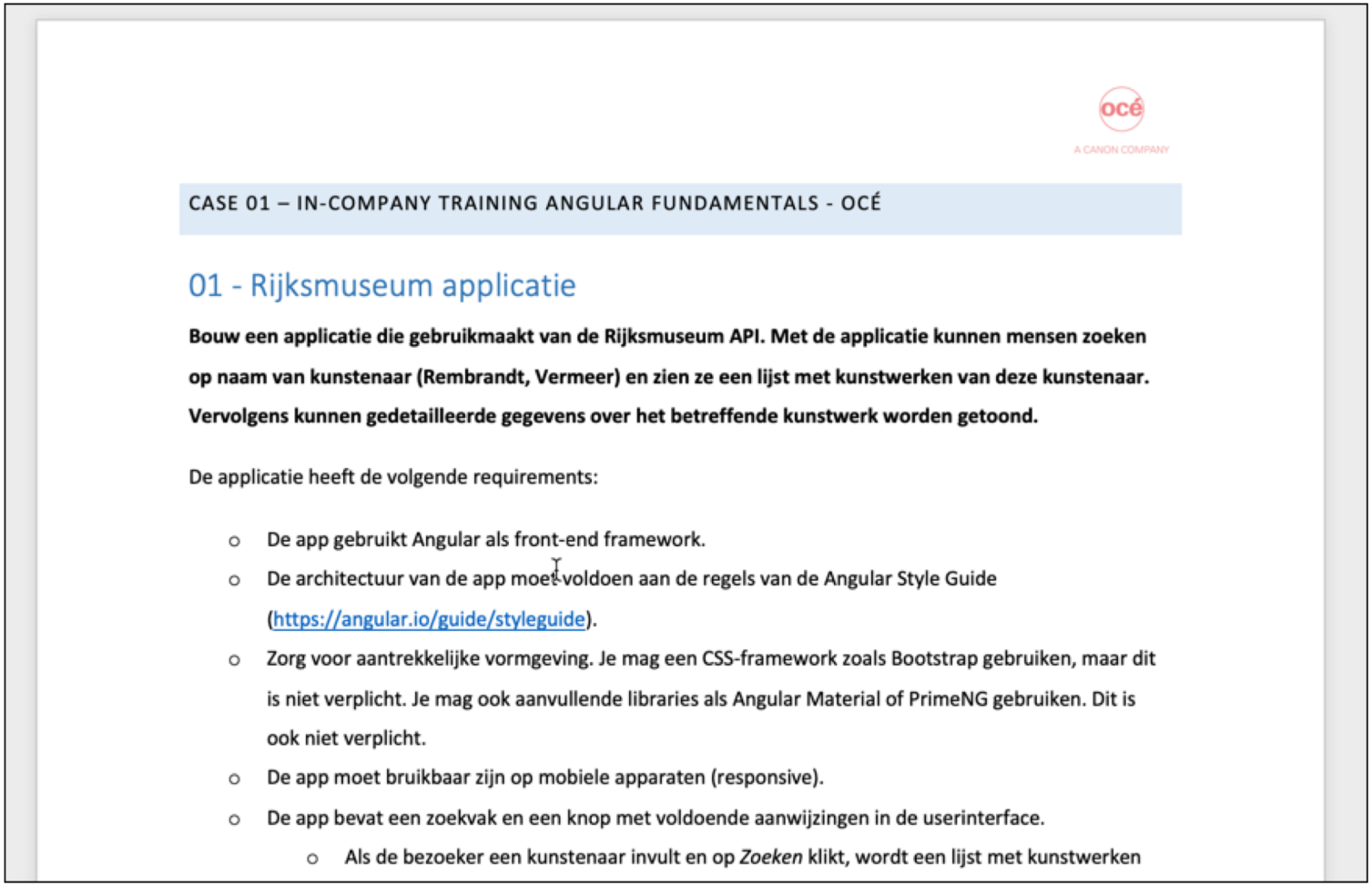
What have you done with Angular yet?

Tell us a little bit about your projects.

What are your expectations of this course?

Case ?

Case



océ
A CANON COMPANY

CASE 01 – IN-COMPANY TRAINING ANGULAR FUNDAMENTALS - OCÉ

01 - Rijksmuseum applicatie

Bouw een applicatie die gebruikmaakt van de Rijksmuseum API. Met de applicatie kunnen mensen zoeken op naam van kunstenaar (Rembrandt, Vermeer) en zien ze een lijst met kunstwerken van deze kunstenaar. Vervolgens kunnen gedetailleerde gegevens over het betreffende kunstwerk worden getoond.

De applicatie heeft de volgende requirements:

- De app gebruikt Angular als front-end framework.
- De architectuur van de app moet voldoen aan de regels van de Angular Style Guide (https://angular.io/guide/styleguide).
- Zorg voor aantrekkelijke vormgeving. Je mag een CSS-framework zoals Bootstrap gebruiken, maar dit is niet verplicht. Je mag ook aanvullende libraries als Angular Material of PrimeNG gebruiken. Dit is ook niet verplicht.
- De app moet bruikbaar zijn op mobiele apparaten (responsive).
- De app bevat een zoekvak en een knop met voldoende aanwijzingen in de userinterface.
    - Als de bezoeker een kunstenaar invult en op *Zoeken* klikt, wordt een lijst met kunstwerken

Volunteers?

2 days ‘Advanced’

What to discuss?

Broadening?

or...

deepening?

Agenda - 2 days

- Various
  - Case
  - Some Angular CLI tips & tricks , Angular Console
- NG applications with multiple modules
  - Routing and Lazy loading modules
  - Loading strategies
- Patterns:
  - Smart components/View components –
  - Content Projection
- Managing state with @ngrx/store (intro)
- E2E-testing
- …

Labs and example code

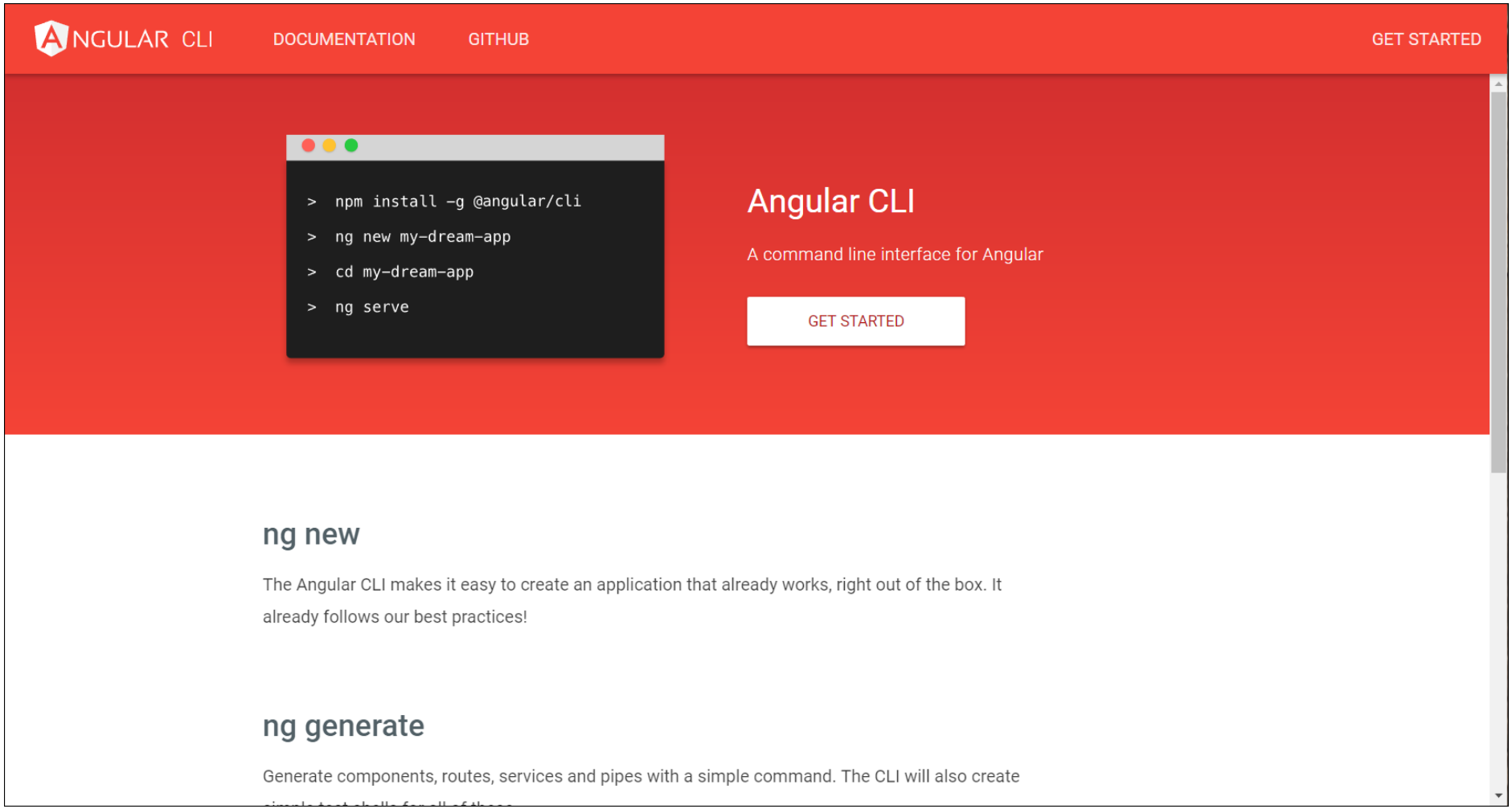
1. Labs/Exercises
    - In the PDF's in the Github-repo. But: feel free to deviate. Adapt to suit your own needs! (hobby, work, current projects)

2. Example code
    - Executions of the exercises, small projects (`npm install`, `npm start`)
    - Work in progress – let me know of additions/errors!
    - github.com/PeterKassenaar/AngularAdvanced

Questions?

Angular CLI

Scaffold new projects, modules, components via command line…


ANGULAR CLI
DOCUMENTATION
GITHUB
GET STARTED
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
Angular CLI
A command line interface for Angular
GET STARTED
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

We'll be using Angular-CLI this course

- It *is* possible to configure your Angular app by hand
- Using the CLI it's much simpler.
- CLI-options:
  - Scaffolding
  - Generating
  - Testing
  - Building
  - AOT-Compiling
  - …

https://cli.angular.io

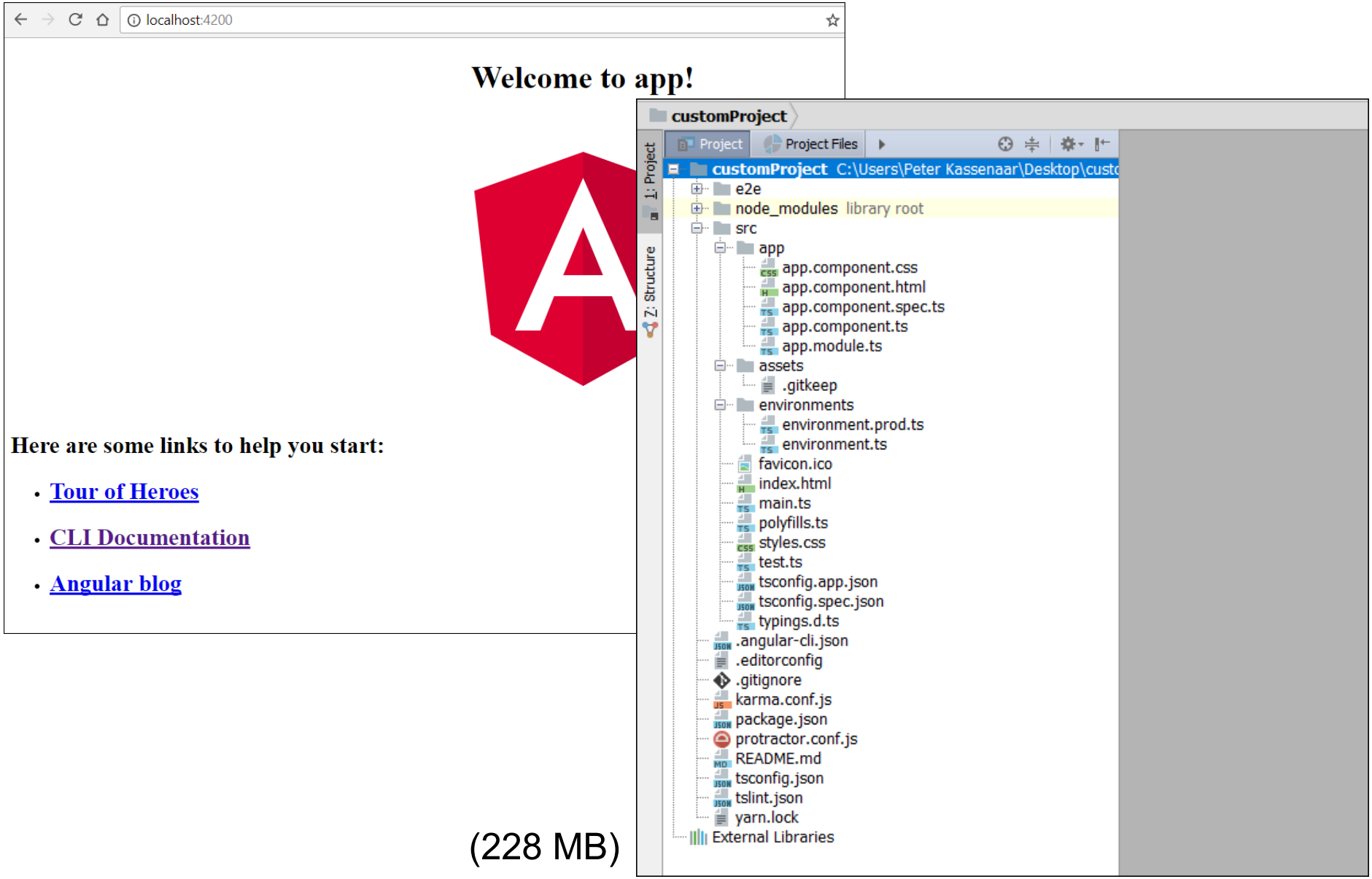
Main commands

ng new – create basic app

```
ng new PROJECT_NAME
cd PROJECT_NAME
ng serve
```

Project is served on `http://localhost:4200`

Default application



(228 MB)

Some CLI tips & tricks

- `ng serve --open` Directly open the compiled project in the browser
- `ng serve --port 4300` Serve project on different port
- `ng serve --ssl` Serve using `https://`
- `ng serve --live-reload false` Do not use live reload
- `ng serve --help` Overview of all other options

More `ng` tooling

- `ng generate <blueprint> --dry-run` Do not write output files
- `ng generate <blueprint> --spec false` Do not write spec file
- `ng generate module <name> --routing` add routing to new module

Lots (!) of options

Adding Features to Your Angular Application

You can use the `ng generate` command to add features to your existing application:

- `ng generate class my-new-class`: add a class to your application
- `ng generate component my-new-component`: add a component to your application
- `ng generate directive my-new-directive`: add a directive to your application
- `ng generate enum my-new-enum`: add an enum to your application
- `ng generate module my-new-module`: add a module to your application
- `ng generate pipe my-new-pipe`: add a pipe to your application
- `ng generate service my-new-service`: add a service to your application

The `generate` command and the different sub-commands also have shortcut notations, so the following commands are similar:

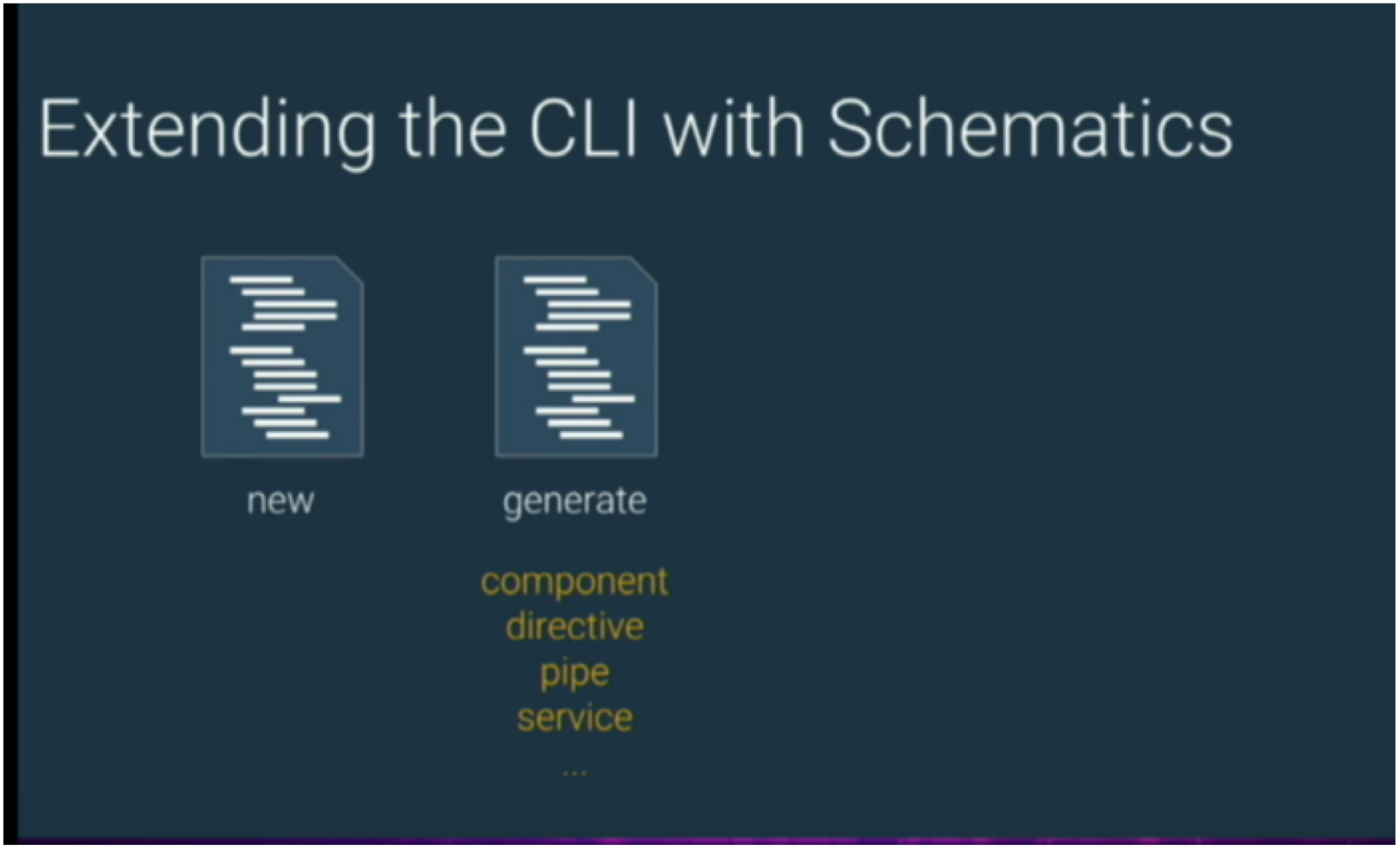
- `ng g cl my-new-class`: add a class to your application
- `ng g c my-new-component`: add a component to your application
- `ng g d my-new-directive`: add a directive to your application
- `ng g e my-new-enum`: add an enum to your application
- `ng g m my-new-module`: add a module to your application
- `ng g p my-new-pipe`: add a pipe to your application
- `ng g s my-new-service`: add a service to your application

Each of the different sub-commands performs a different task and offers different options and parameters.

Let's have a look at each of them.

Adding a new class

New CLI Options



Extending the CLI with Schematics

new

generate

component
directive
pipe
service
...

update

add

Info on the Angular 6.x keynote

https://www.youtube.com/watch?v=dIxknqPOWms

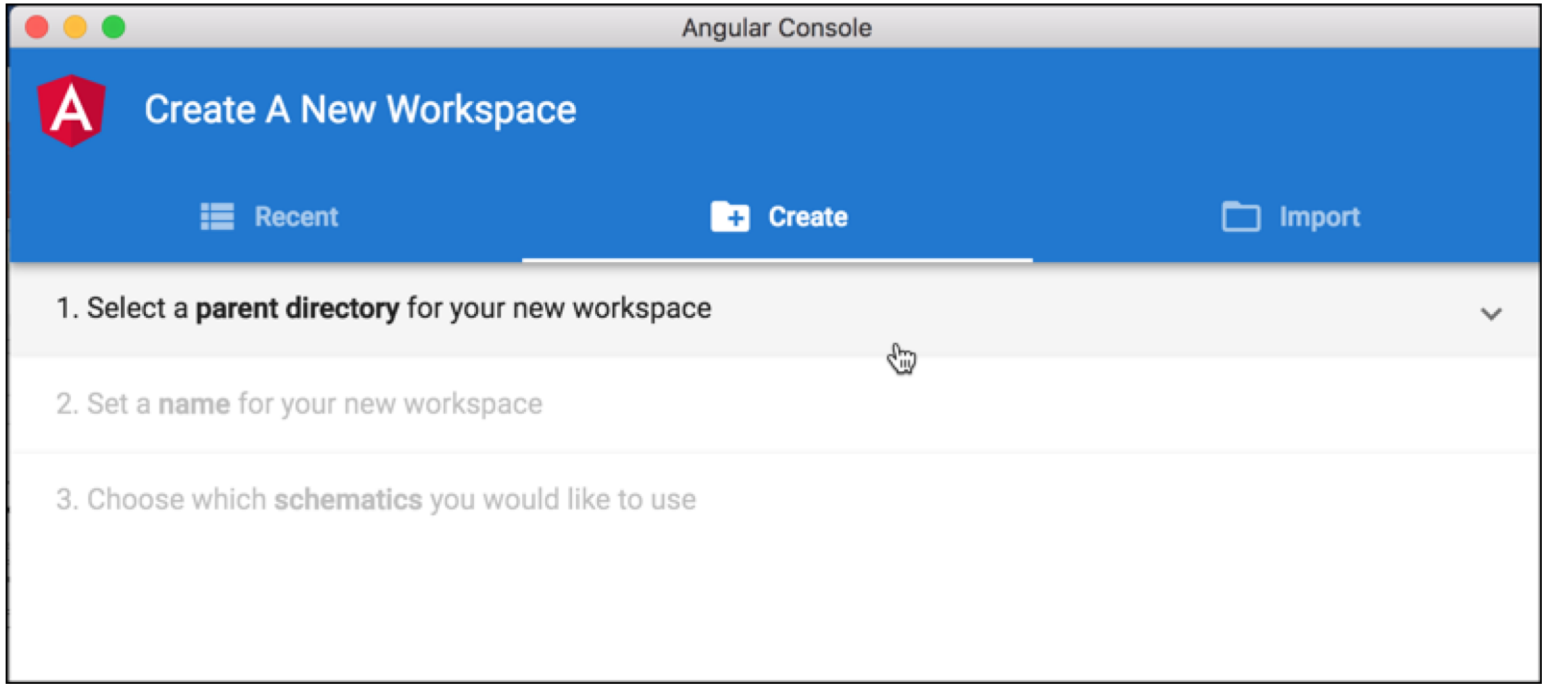
NEW – As of August 2018

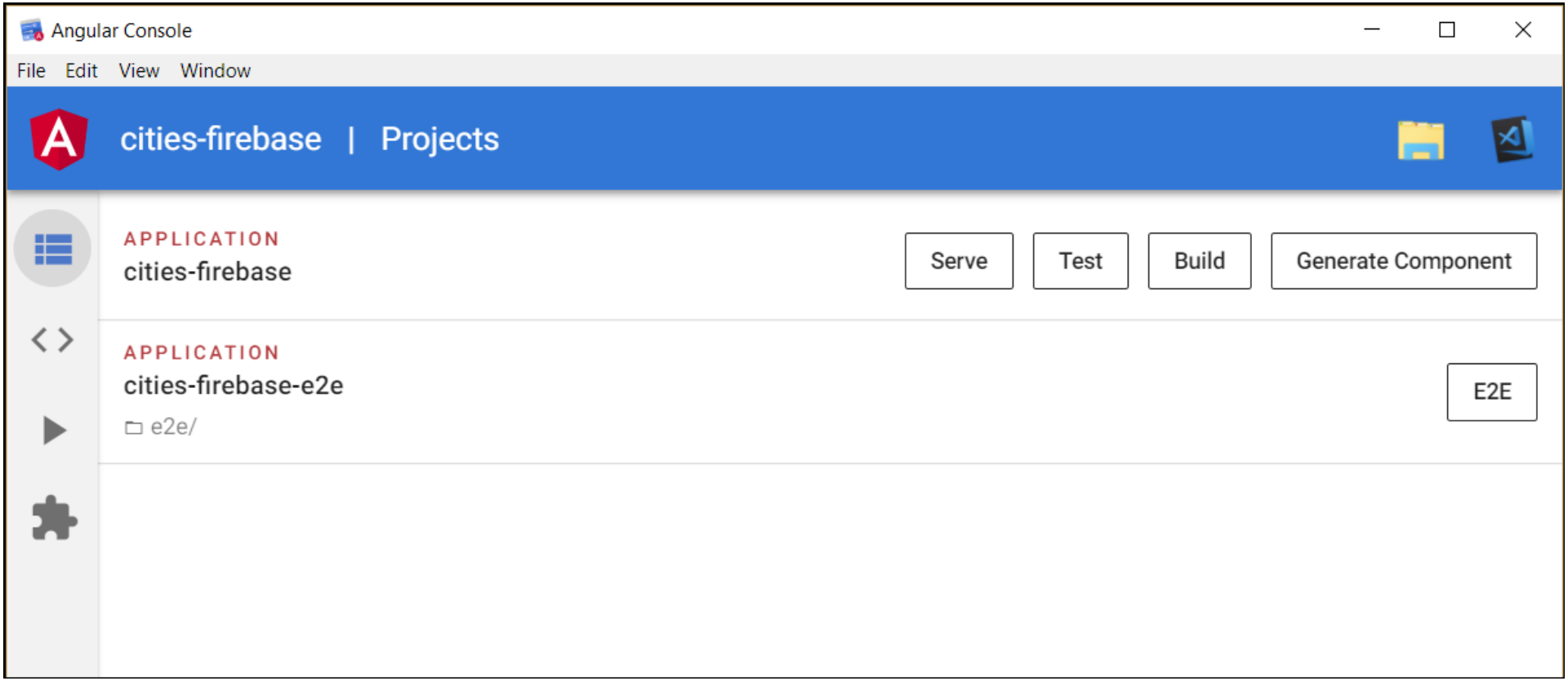
https://angularconsole.com/

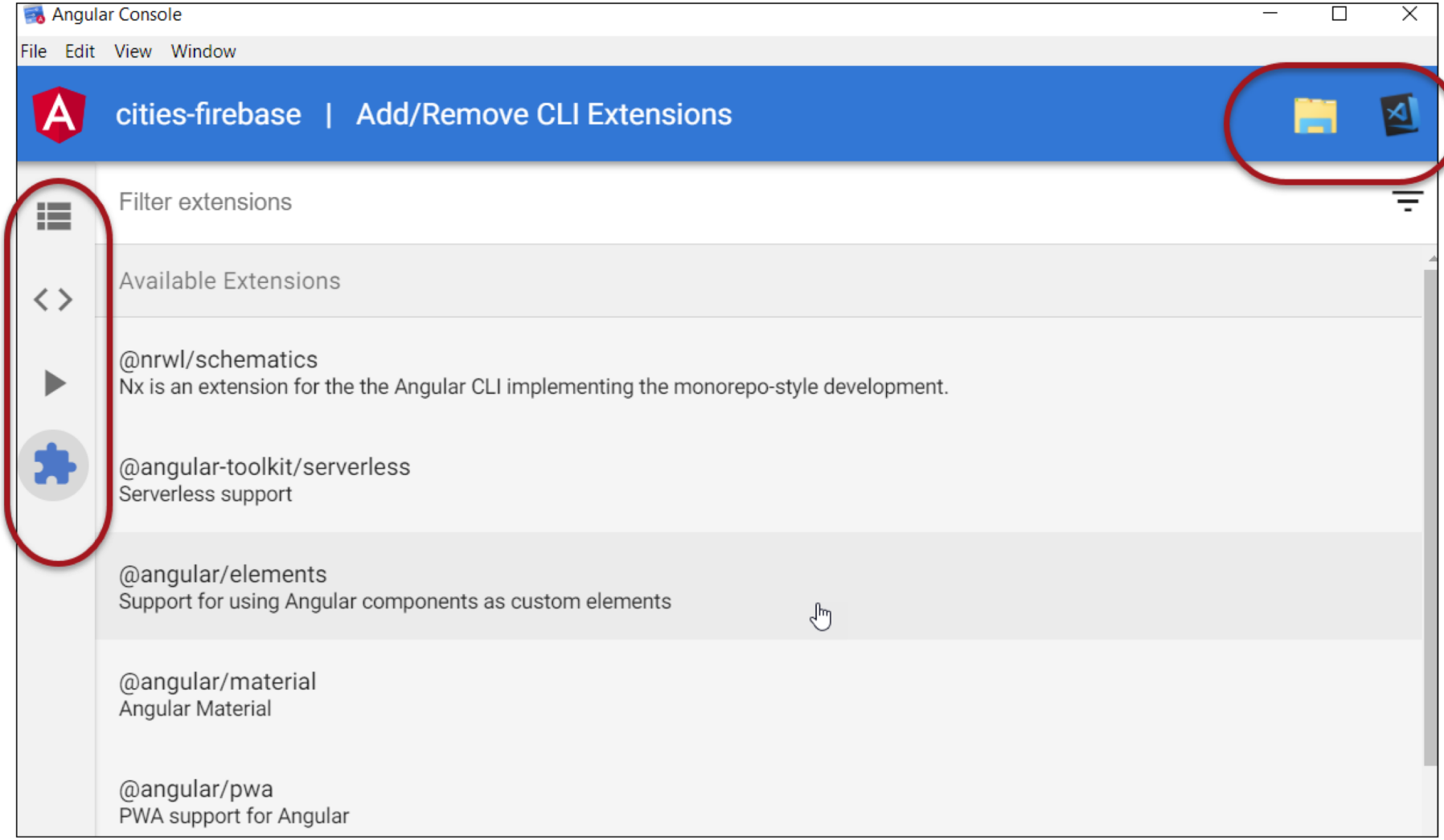
Angular Console

- By nrwl.io, team of Victor Savkin and Jeff Cross
- Visual interface to the CLI
  - Code generation
  - Run Custom NPM scripts
  - Discover and install extensions
  - Build CLI commands visually – no need to remember all shortcuts
  - Integrated terminal output
  - Create new projects, import existing projects

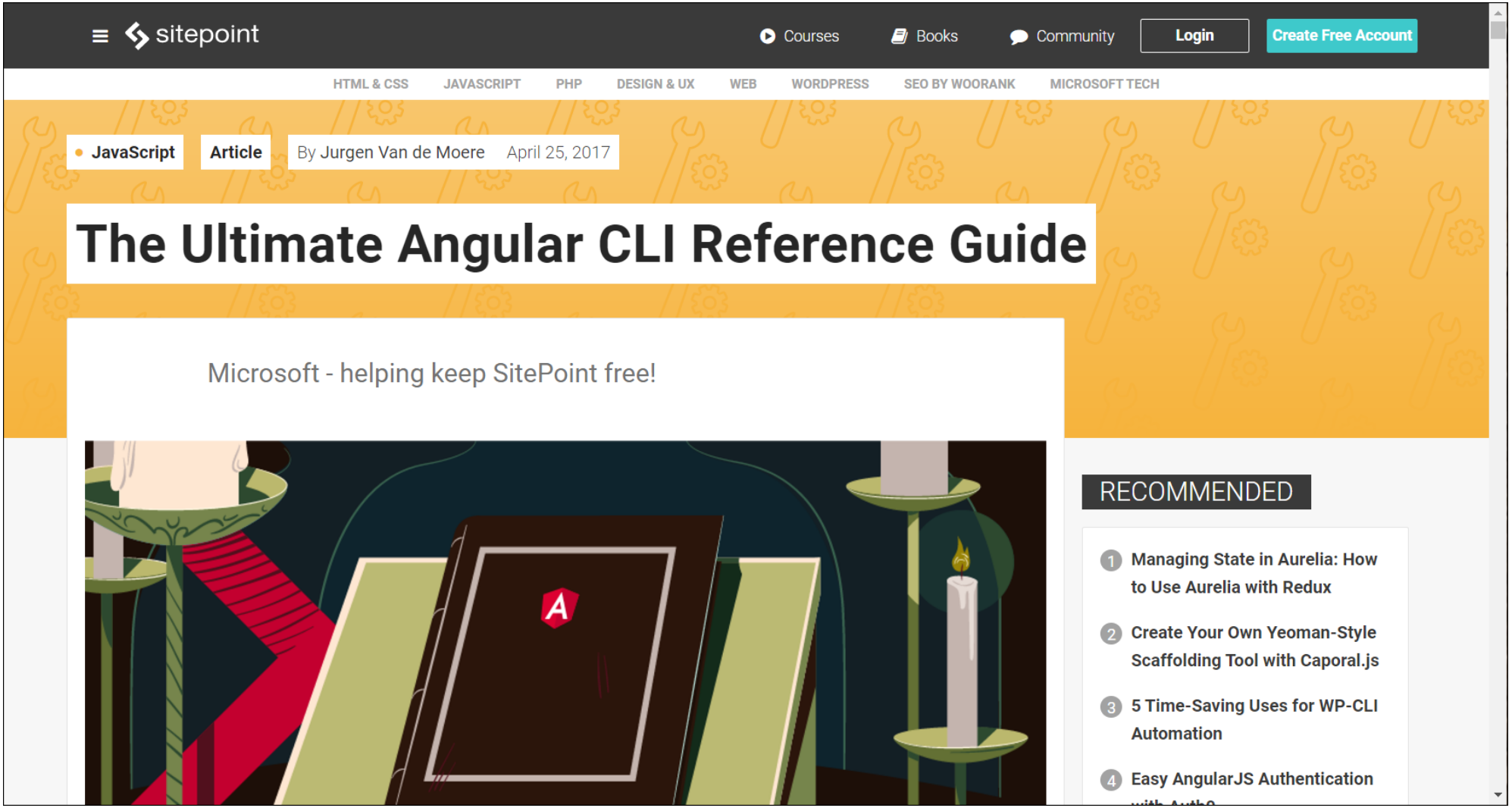
“A Gateway for Beginners. A Powertool for Experts.”




Angular Console
File Edit View Window
cities-firebase | Projects
APPLICATION
cities-firebase
Serve
Test
Build
Generate Component
APPLICATION
cities-firebase-e2e
e2e/
E2E


Angular Console
File Edit View Window
cities-firebase | Add/Remove CLI Extensions
Filter extensions
Available Extensions
@nrwl/schematics
Nx is an extension for the the Angular CLI implementing the monorepo-style development.
@angular-toolkit/serverless
Serverless support
@angular/elements
Support for using Angular components as custom elements
@angular/material
Angular Material
@angular/pwa
PWA support for Angular

More background information

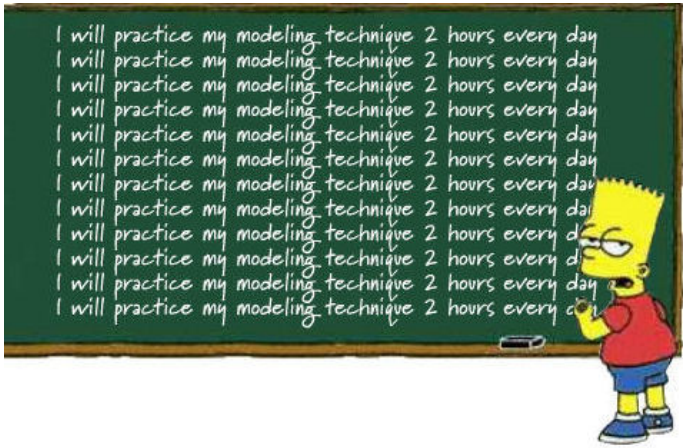


https://www.sitepoint.com/ultimate-angular-cli-reference/

Workshop

- Download and install Angular Console
- Generate a new project with it
- *OR:*
- Import an existing project

- Generate a new component or a new service with it
- Run some scripts from within Angular Console (`start`, `build`, `serve`)
- Add some new CLI extensions, for instance
  - `@angular/material`
  - `@angular/elements`
  - See how/where they are installed

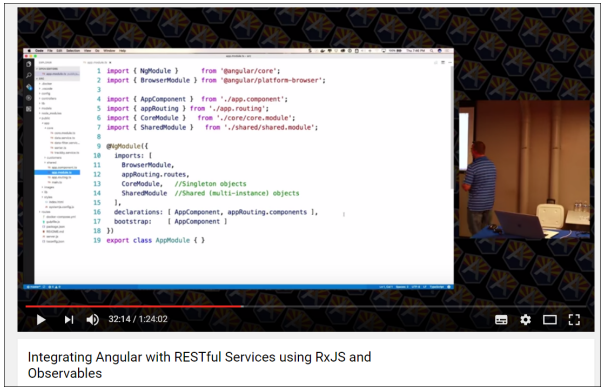


Multiple modules

Splitting your application into separate, reusable modules

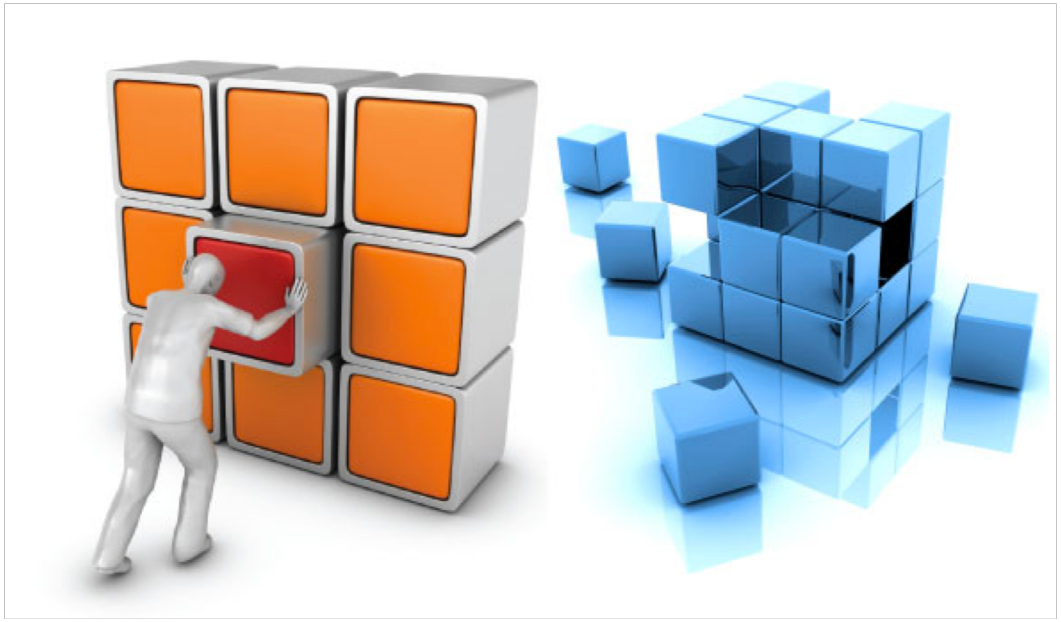
- Divide your app into *logical* and often *reusable* pieces of code
- Keyword : code organization

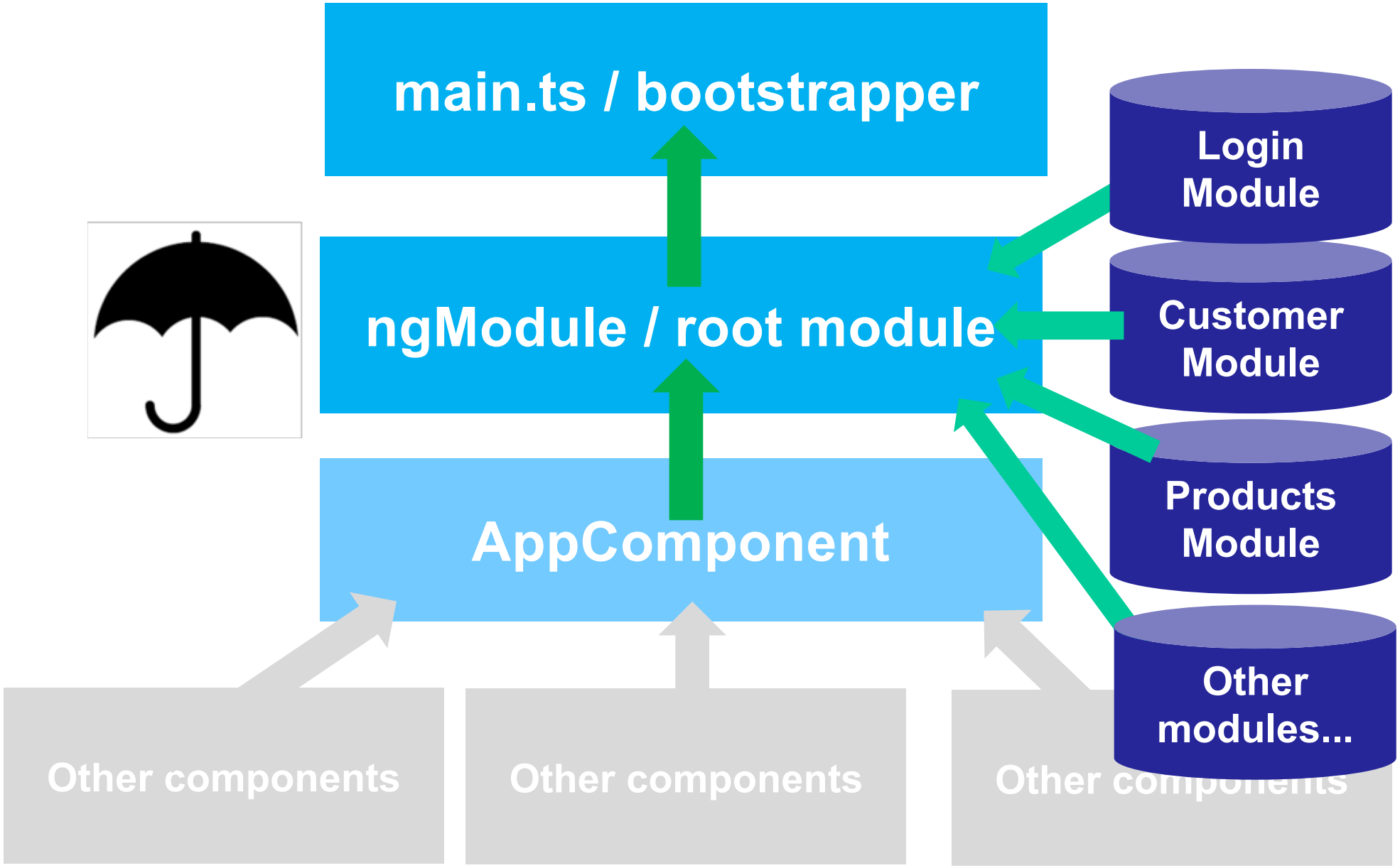
- Use one `AppModule` - the root of your app
- Use one `CoreModule` - containing all *singletons* in your app
- Use one `SharedModule` - containing all shared resources, possible multiple instances
- Use additional modules *per feature*
- https://www.youtube.com/watch?v=YxK4UW4UfCk



Application – multiple Modules

- *Reuse* of Components, Pipes, Routes and Services etc. over different apps
- *Wrap* each set of logical related components, services, etc. in its own module.




main.ts / bootstrapper
ngModule / root module
AppComponent
Login Module
Customer Module
Products Module
Other modules...
Other components
Other components
Other components

Steps

1. Create a new module
   - Optional: test first with `--dry-run`
   - `ng generate module customers --dry-run`
2. Create component(s) inside that module
   - Again: test first with `--dry-run`
   - `ng generate component customers --module customers --dry-run`
3. Apply UI, logic, etc. to your component
4. Export your component inside `customer.module.ts`
   - `exports : [CustomerComponent],`
   - Otherwise it can’t be used in other components!
5. Provide new module to `app.module.ts`
   - `imports: [CustomerModule]`

Optional : SharedModule

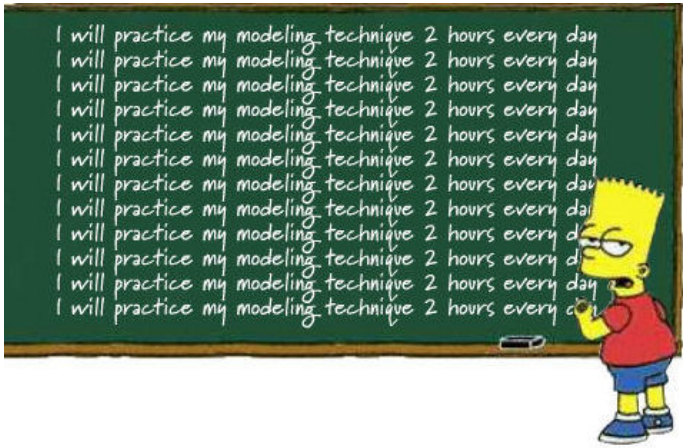
- Reuse components in multiple modules? Use a SharedModule
  - `ng g m shared` – shorthand notation
- Create components inside SharedModule
- Import SharedModule in other modules
- It doesn’t have to be in AppModule if you don’t use it directly!
- It *does* add size to module bundles AFAIK
  - Modules need to be able to run on their own.
  - But: it’s *not* included multiple times in one bundle


Demo

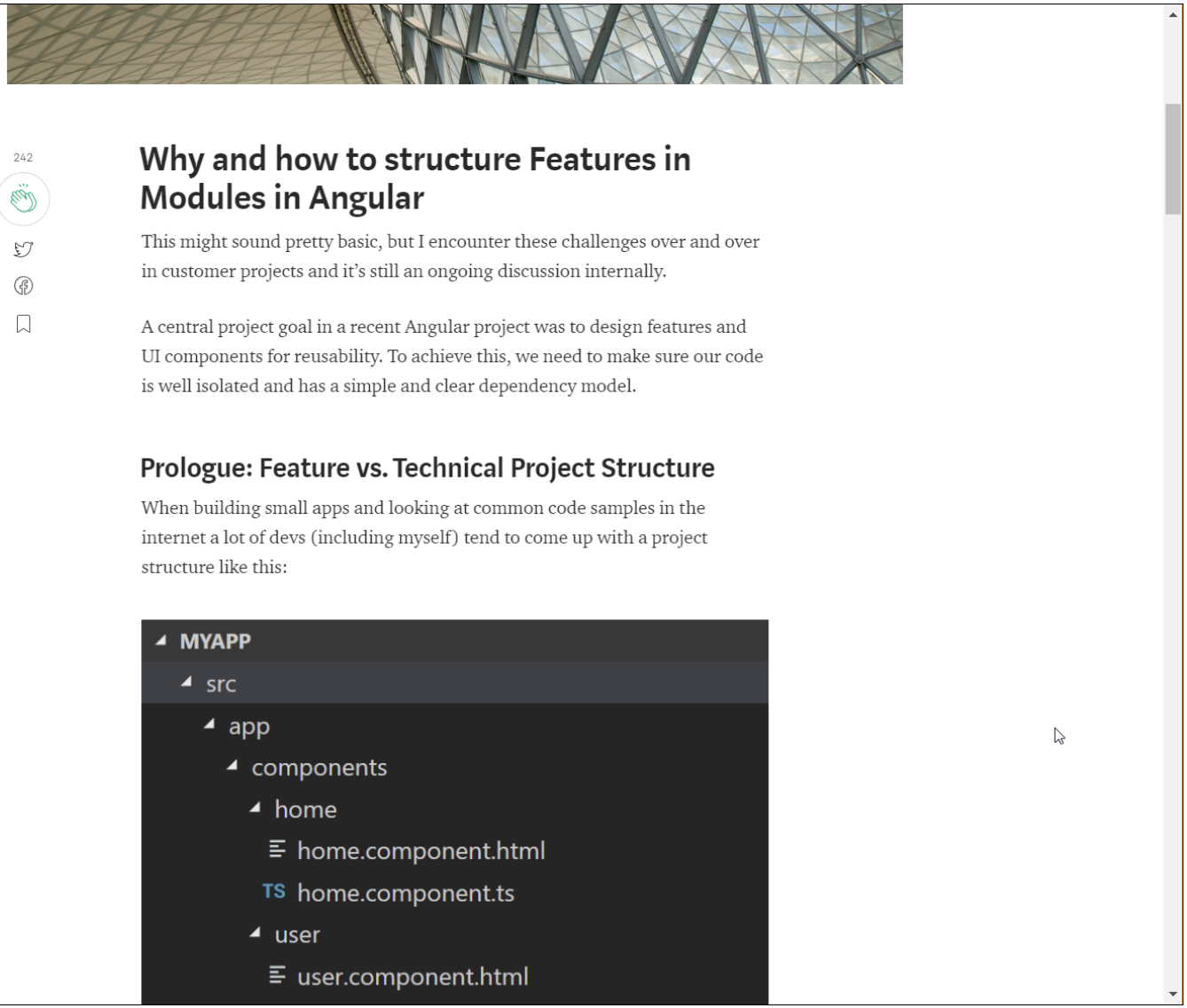
Workshop

- Open `../100-multiple modules`.
- Create a new module
- Create a new component inside this new module and give it some UI.
- Include the module in the Main Module and show it besides other modules
- Include the Search Component in your own module

- *OR:*
- Add Multiple Modules from scratch to your own application, using the steps described in this module.



How to structure feature modules



242

Why and how to structure Features in Modules in Angular

This might sound pretty basic, but I encounter these challenges over and over in customer projects and it's still an ongoing discussion internally.

A central project goal in a recent Angular project was to design features and UI components for reusability. To achieve this, we need to make sure our code is well isolated and has a simple and clear dependency model.

Prologue: Feature vs. Technical Project Structure

When building small apps and looking at common code samples in the internet a lot of devs (including myself) tend to come up with a project structure like this:

https://medium.com/@philippbauknecht/why-and-how-to-structure-features-in-modules-in-angular-d5602c6436be