



Angular Advanced

01 - Introduction



A CANON COMPANY



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 ?

Broadening?



or...

deepening?



Agenda - 2 days

- Various
 - 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...

```
> 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

localhost:4200

Welcome to app!



Here are some links to help you start:

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)

customProject

Project Project Files

customProject C:\Users\Peter Kassenaar\Desktop\custo

e2e

node_modules library root

src

app

app.component.css

app.component.html

app.component.spec.ts

app.component.ts

app.module.ts

assets

.gitkeep

environments

environment.prod.ts

environment.ts

favicon.ico

index.html

main.ts

polyfills.ts

styles.css

test.ts

tsconfig.app.json

tsconfig.spec.json

typings.d.ts

.angular-cli.json

.editorconfig

.gitignore

karma.conf.js

package.json

protractor.conf.js

README.md

tsconfig.json

tslint.json

yarn.lock

External Libraries

(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

...

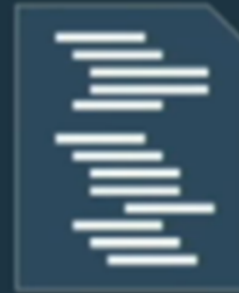
Extending the CLI with Schematics



new



generate



update



add

component
directive
pipe
service

...

Info on the Angular 6.x keynote



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

NEW – As of August 2018

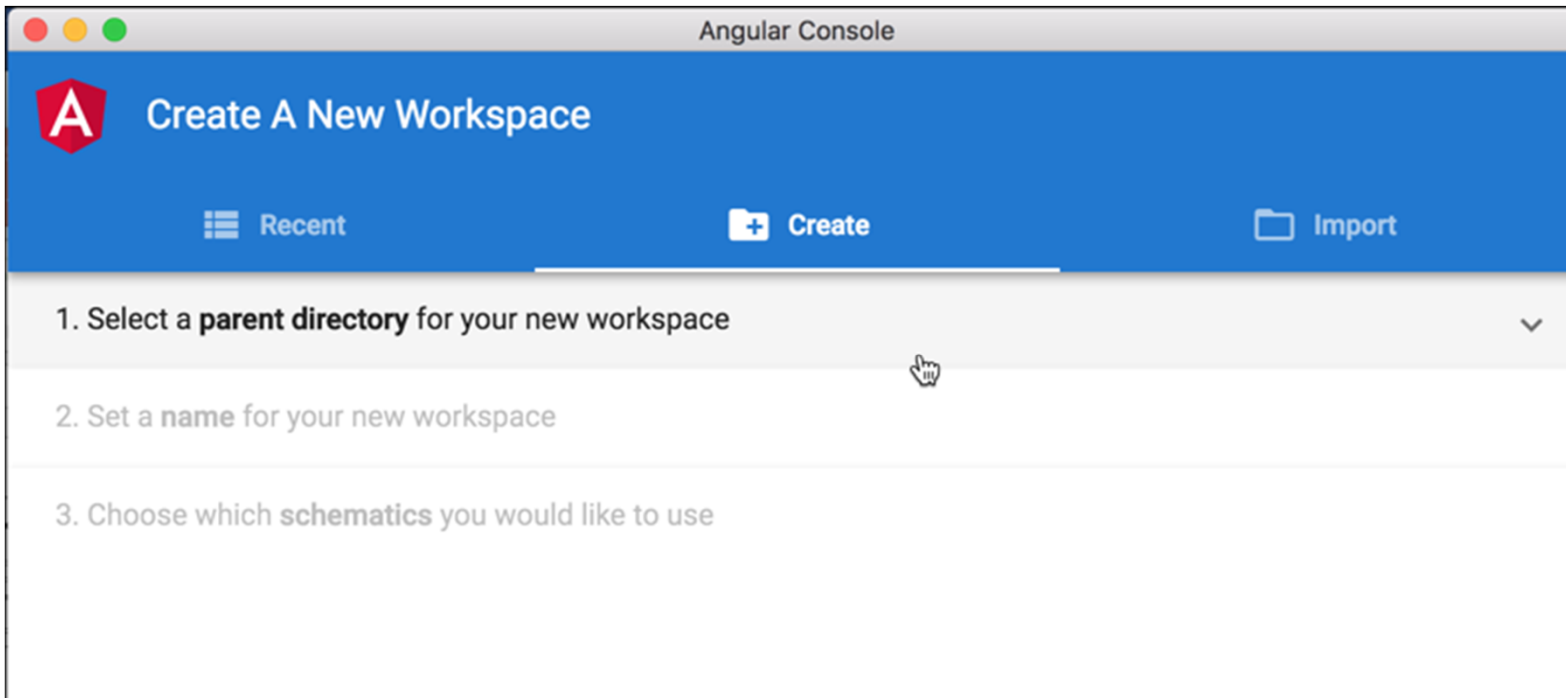


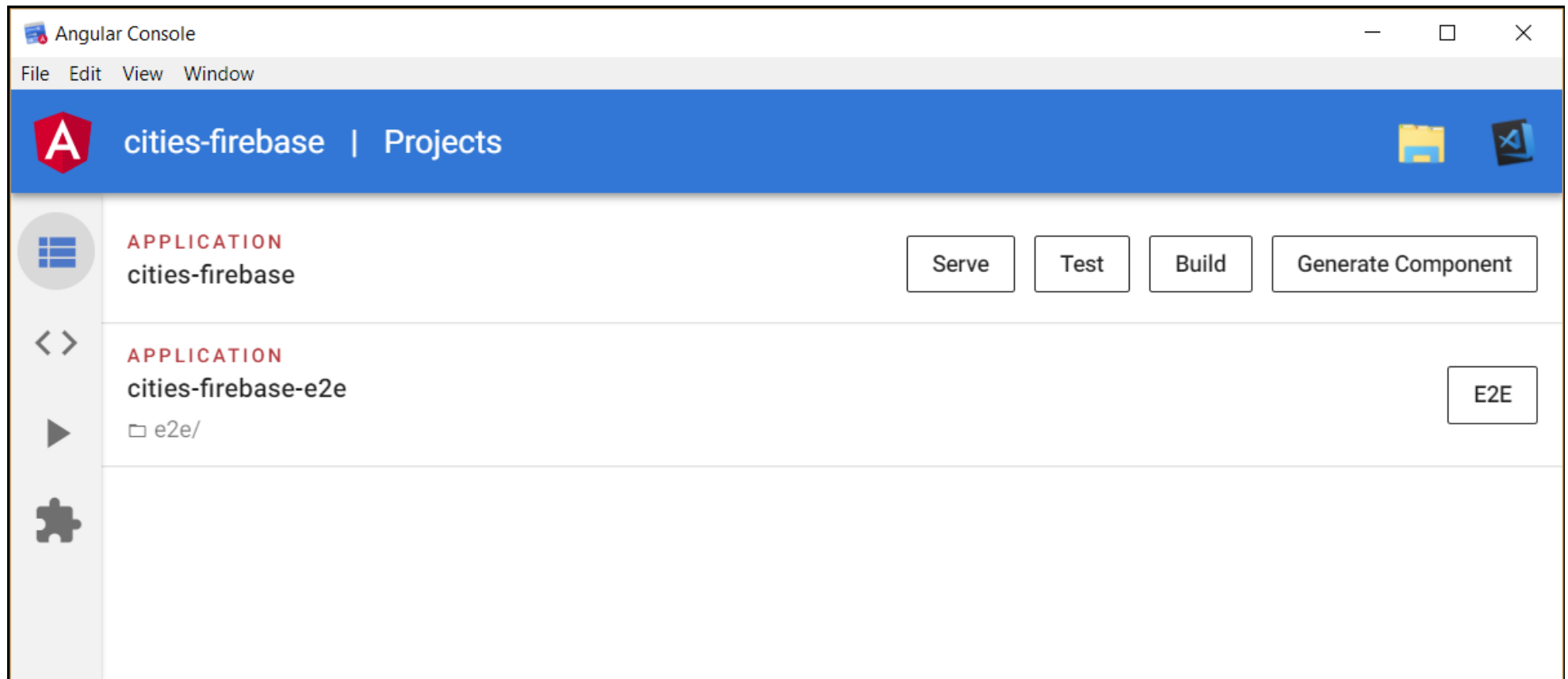
<https://angularconsole.com/>

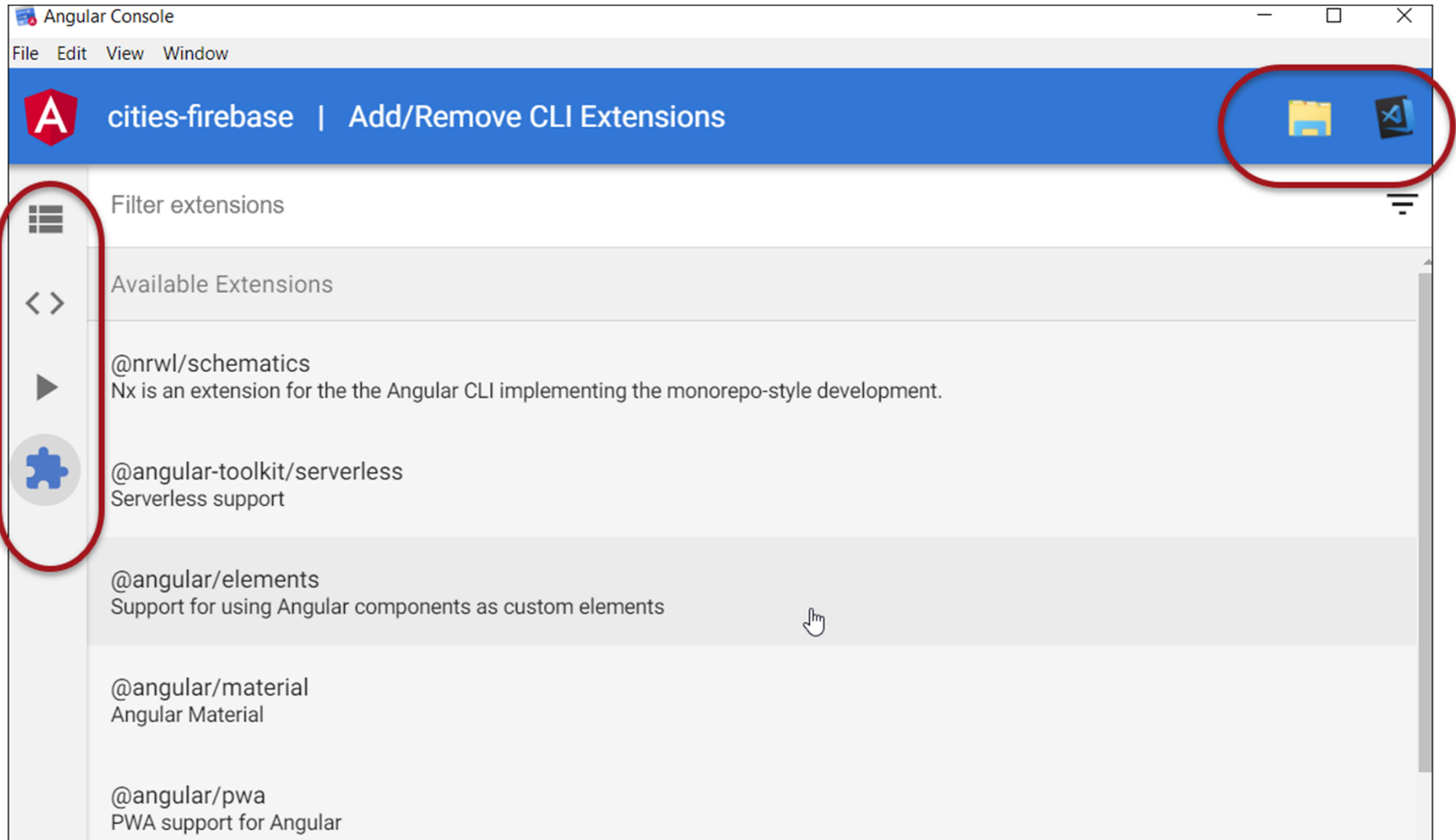
Angular Console

- 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.”*







More background information

The screenshot shows the SitePoint website interface. At the top, there is a dark navigation bar with the SitePoint logo, a menu icon, and links for Courses, Books, Community, Login, and Create Free Account. Below this is a white category bar with links for HTML & CSS, JAVASCRIPT, PHP, DESIGN & UX, WEB, WORDPRESS, SEO BY WOORANK, and MICROSOFT TECH. The main content area has an orange background with a pattern of tools. A white box at the top left of the article section contains the tags 'JavaScript' and 'Article', the author 'By Jurgen Van de Moere', and the date 'April 25, 2017'. The article title 'The Ultimate Angular CLI Reference Guide' is displayed in large, bold, black text. Below the title, a white box contains the text 'Microsoft - helping keep SitePoint free!'. The main image of the article is a stylized illustration of a desk with a laptop displaying the Angular logo, a candle, and a book. To the right of the main image, there is a 'RECOMMENDED' section with a list of four articles.

sitepoint

Courses Books Community Login Create Free Account

HTML & CSS JAVASCRIPT PHP DESIGN & UX WEB WORDPRESS SEO BY WOORANK MICROSOFT TECH

JavaScript Article By Jurgen Van de Moere April 25, 2017

The Ultimate Angular CLI Reference Guide

Microsoft - helping keep SitePoint free!

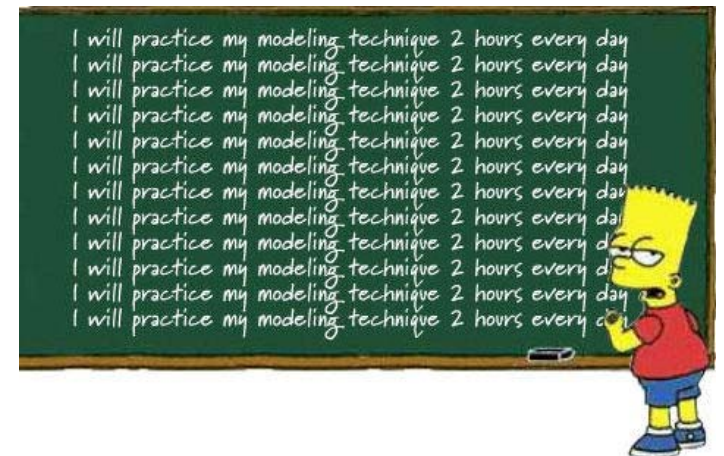
RECOMMENDED

- 1 Managing State in Aurelia: How to Use Aurelia with Redux
- 2 Create Your Own Yeoman-Style Scaffolding Tool with Caporal.js
- 3 5 Time-Saving Uses for WP-CLI Automation
- 4 Easy AngularJS Authentication with Auth0

<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

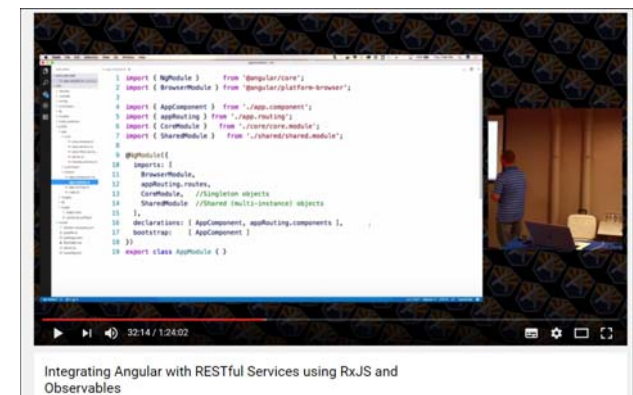




Multiple modules

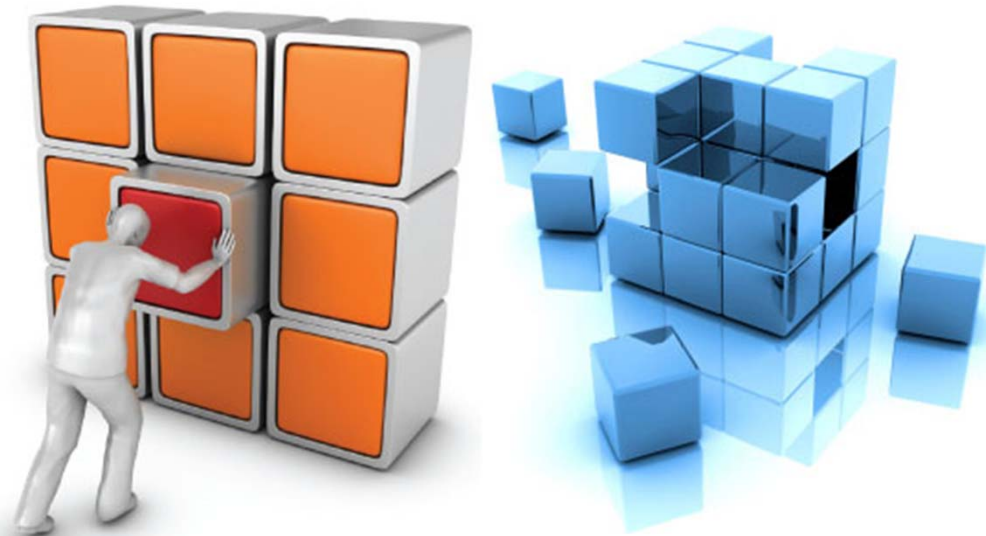
Splitting your application into separate, reusable modules

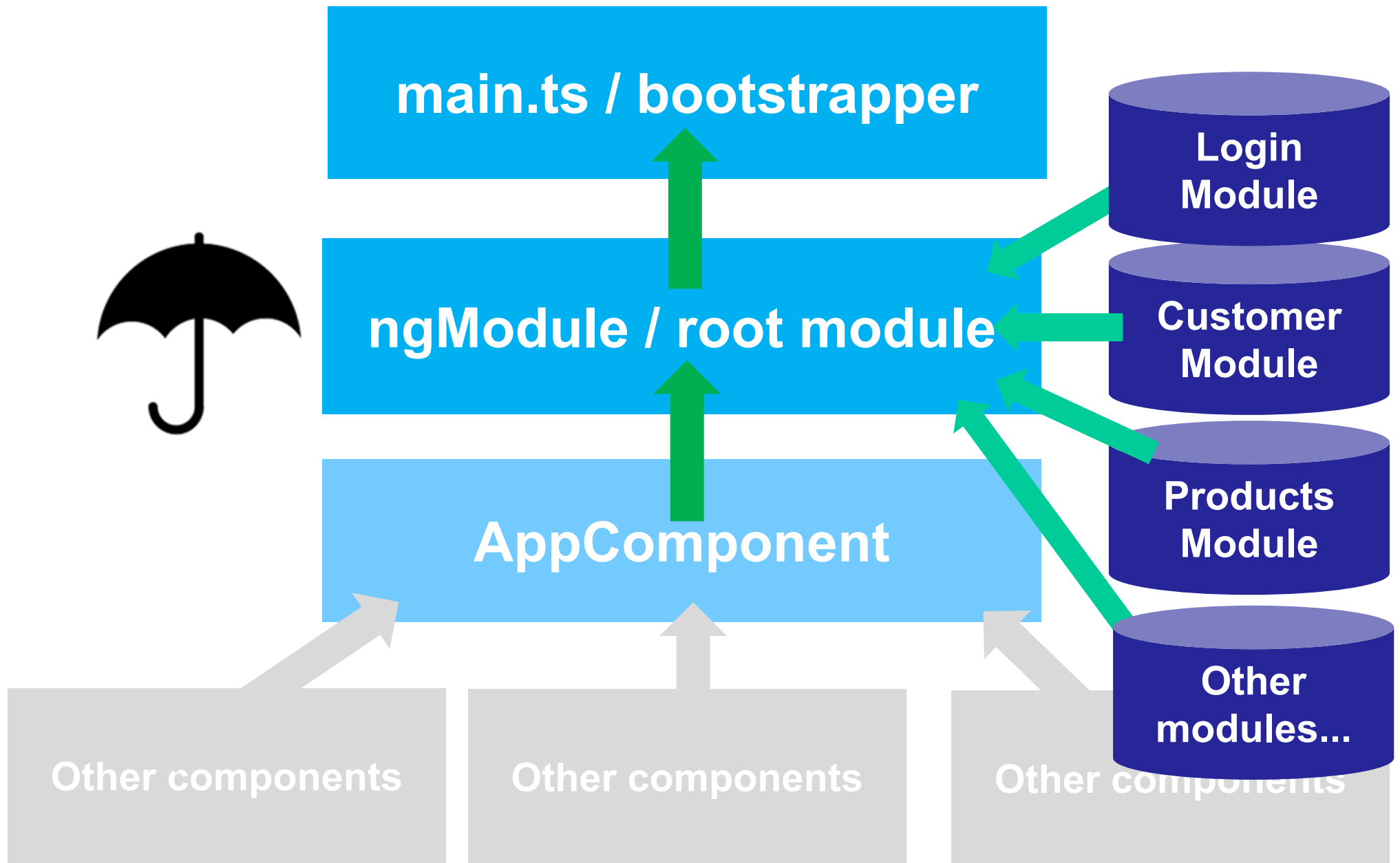
- Divide your app into *logical* and often *reusable* pieces of code
- Keyword : **code organization**
- Use one AppModule - the root of your app
- Use one CoreModule - containing all *singletons* in your app
- Use one SharedModule - containing all shared resources, possible multiple instances
- Use 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.





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]`

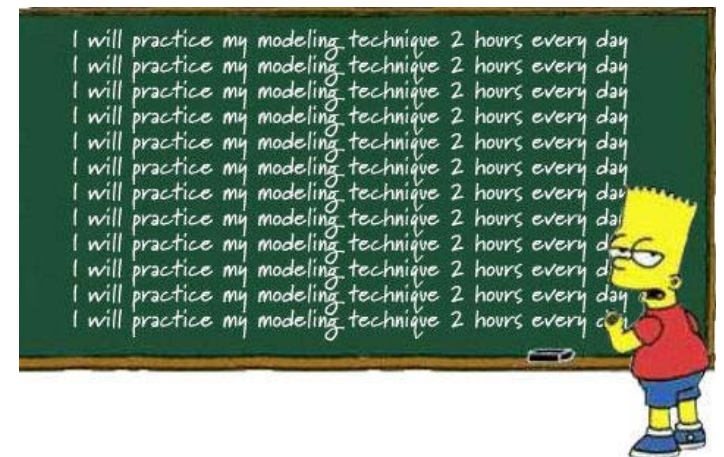
Optional : SharedModule

- Reuse components in multiple modules? Use a SharedModule
 - `NgModule.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



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:

```
MYAPP
├── src
│   ├── app
│   │   ├── components
│   │   │   ├── home
│   │   │   │   ├── home.component.html
│   │   │   │   ├── home.component.ts
│   │   │   └── user
│   │   │       ├── user.component.html
```

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