ilionx

Global Knowledge®

# Angular
# Module – Advanced Introduction

Peter Kassenaar –
info@kassenaar.com

# Peter Kassenaar

- Trainer, author, developer – sinds 1996
- Specialty: *“Everything JavaScript”*
- JavaScript, ES6, Angular, NodeJS, TypeScript, jQuery, PhoneGap, Ionic

www.kassenaar.com/blog

info@kassenaar.com

Twitter: @PeterKassenaar

ING OHRA euricom A DIMENSION DATA COMPANY sanoma delta lloyd zenito BETERE ZEKERHEID VOOR ONDERNEMERS Atos woonbron OBERON INTERACTIVE ROC West-Brabant the eforum FACTORY


www.angulartraining.nl

# About you…

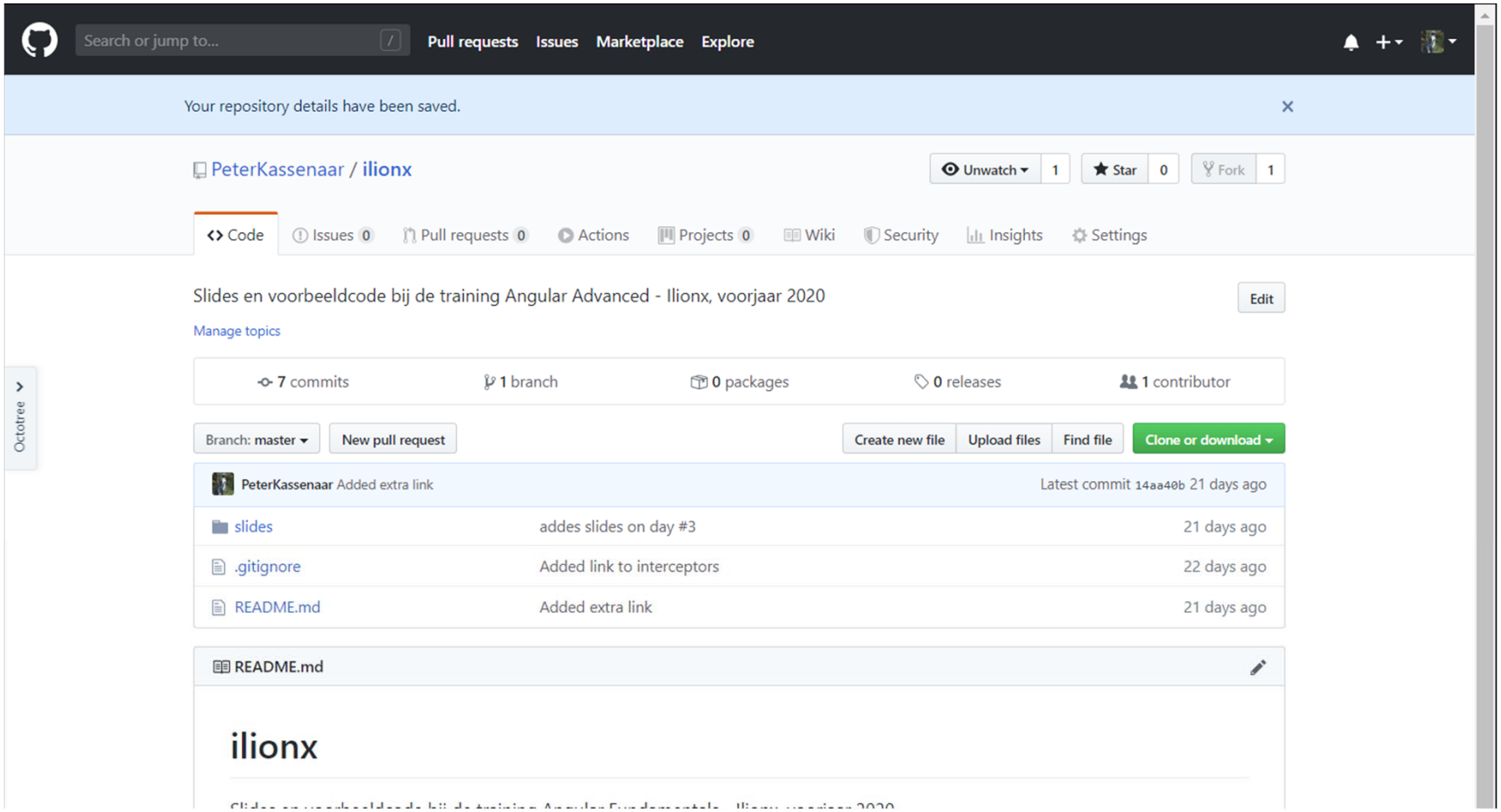
Knowledge of Angular, (mobile/web-) apps?

How long have you worked with Angular yet?

Tell us a little bit about your projects.

What are your expectations of this course?

# github.com/PeterKassenaar/ilionx



# Agenda

1-2 April 2020 – wo + do

20-21 April 2020 – ma + di – *voorlopig*

~18:00 start

~ 19:30 Break

~21:00 einde

# Agenda - 4 nights - Thematic

- Day 1: Architectuur
  - Composing Applications with multiple modules
  - Routing and lazy loading modules
  - Loading Strategies
  - Advanced components
- Day 2: Observables
  - Observables from scratch, more operators
  - Examples (typeahead, fetching data from multiple sources)

# Agenda - 4 nights - Thematic

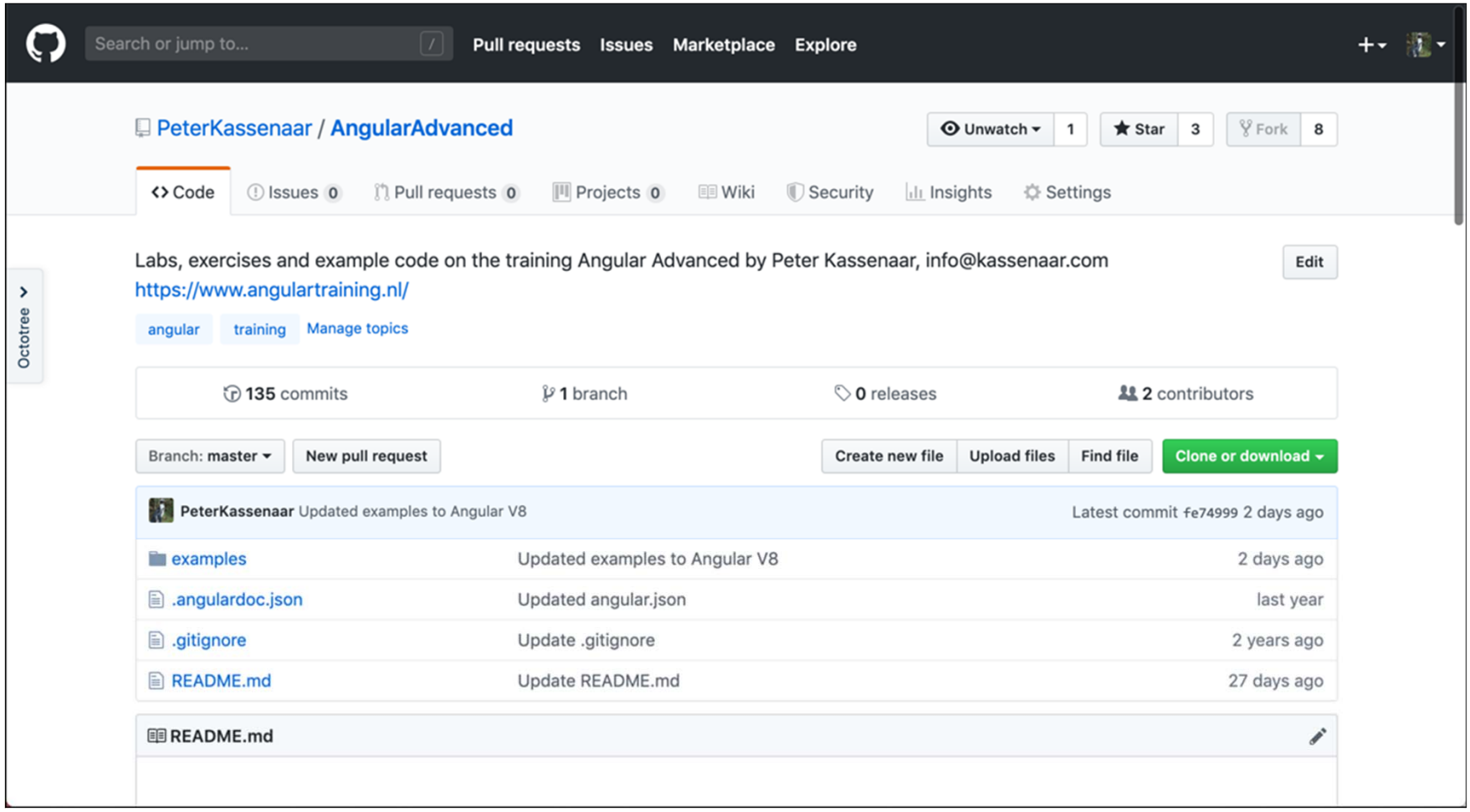
- Day 3: State Management
  - State management with @ngrx/store
  - Concepts – actions, dispatchers, reducers, store
  - Payload, complex types
- Day 4: Enterprise applications
  - Working with monorepo's, Angular CLI
  - More on Angular Schematics

# 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

# Generic ‘Advanced’ Github repo

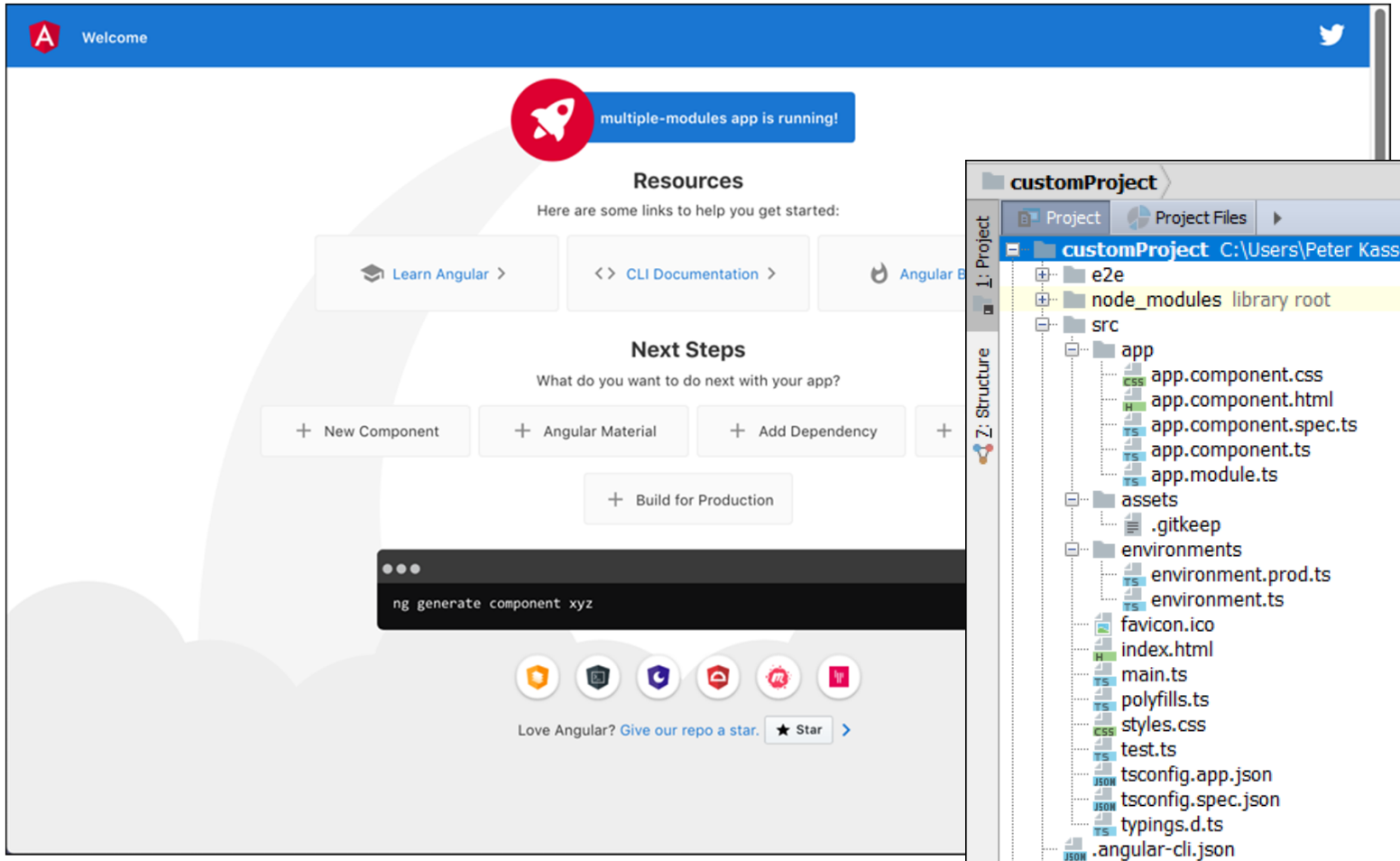


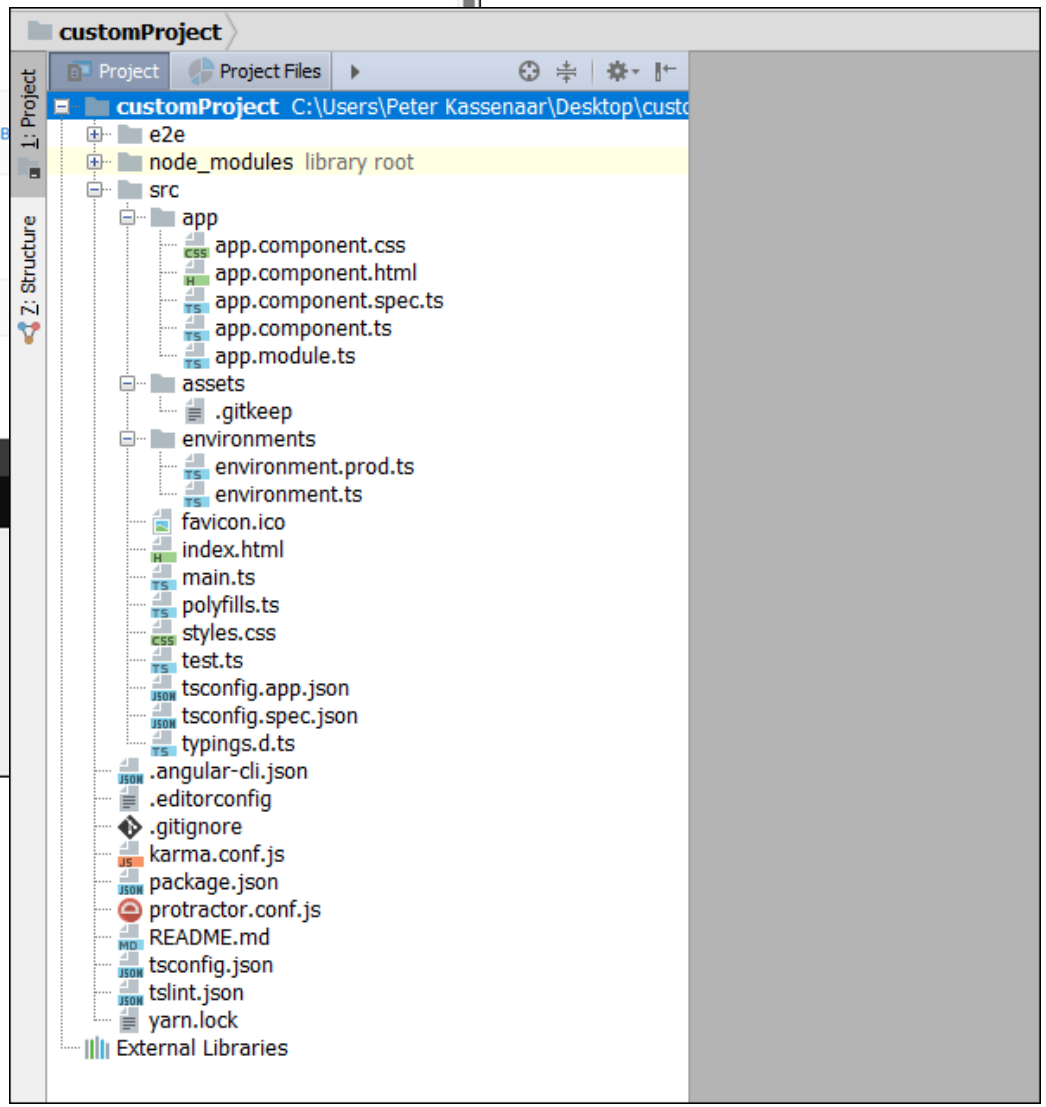

https://github.com/PeterKassenaar/AngularAdvanced

# Multiple modules

Splitting your application into separate, reusable modules

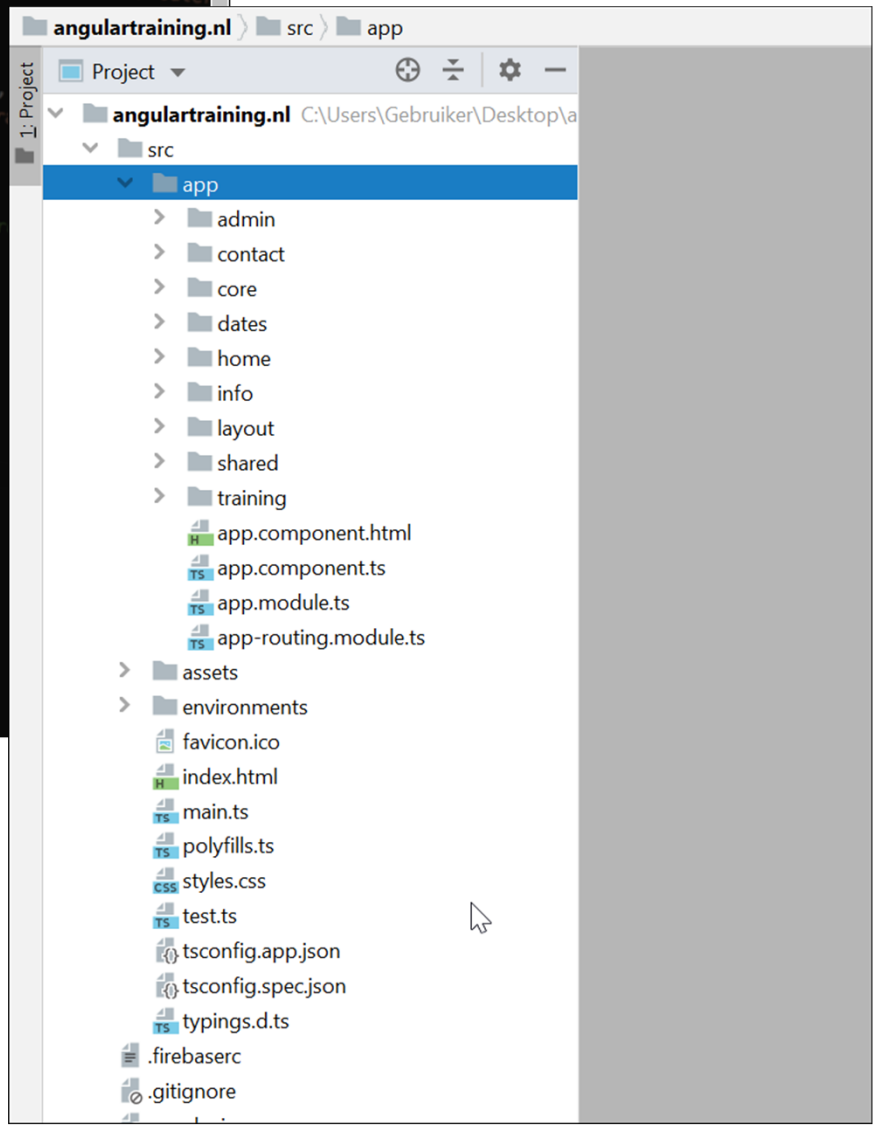
# Default application – 1 module




(228 MB)

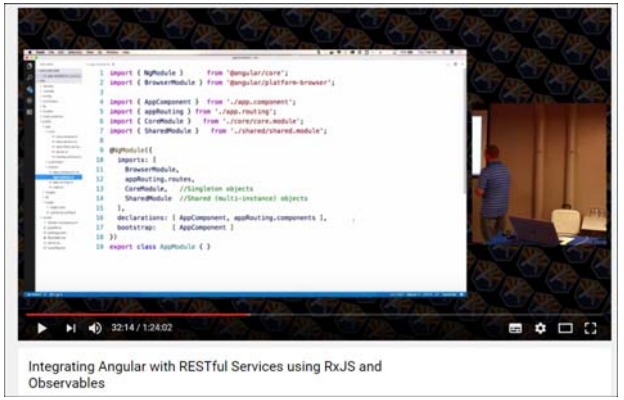
# Bigger applications – multiple modules

# Angular Modules

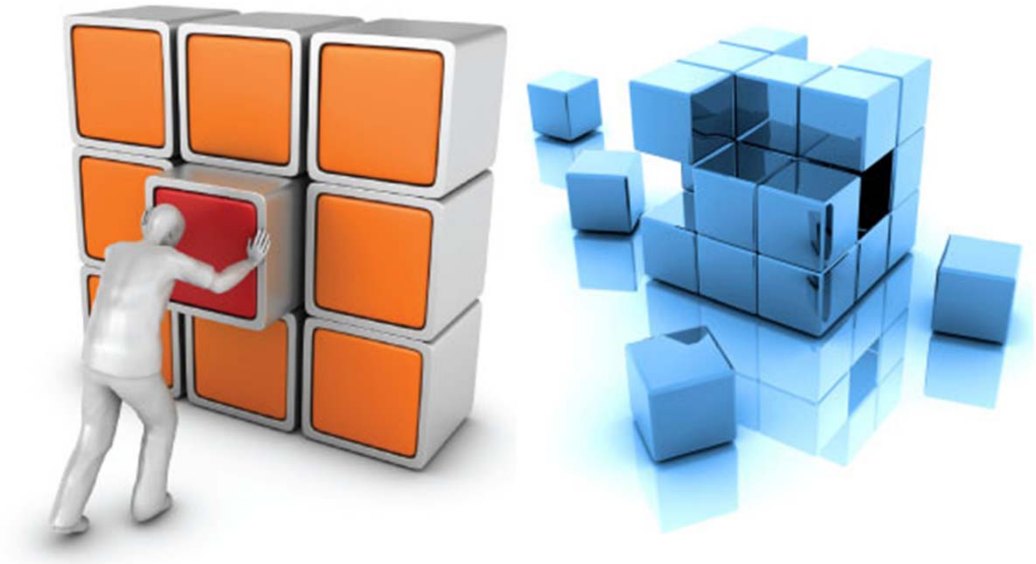
- Divide your app into *logical* and often *reusable* pieces of code
- Keyword : code organization

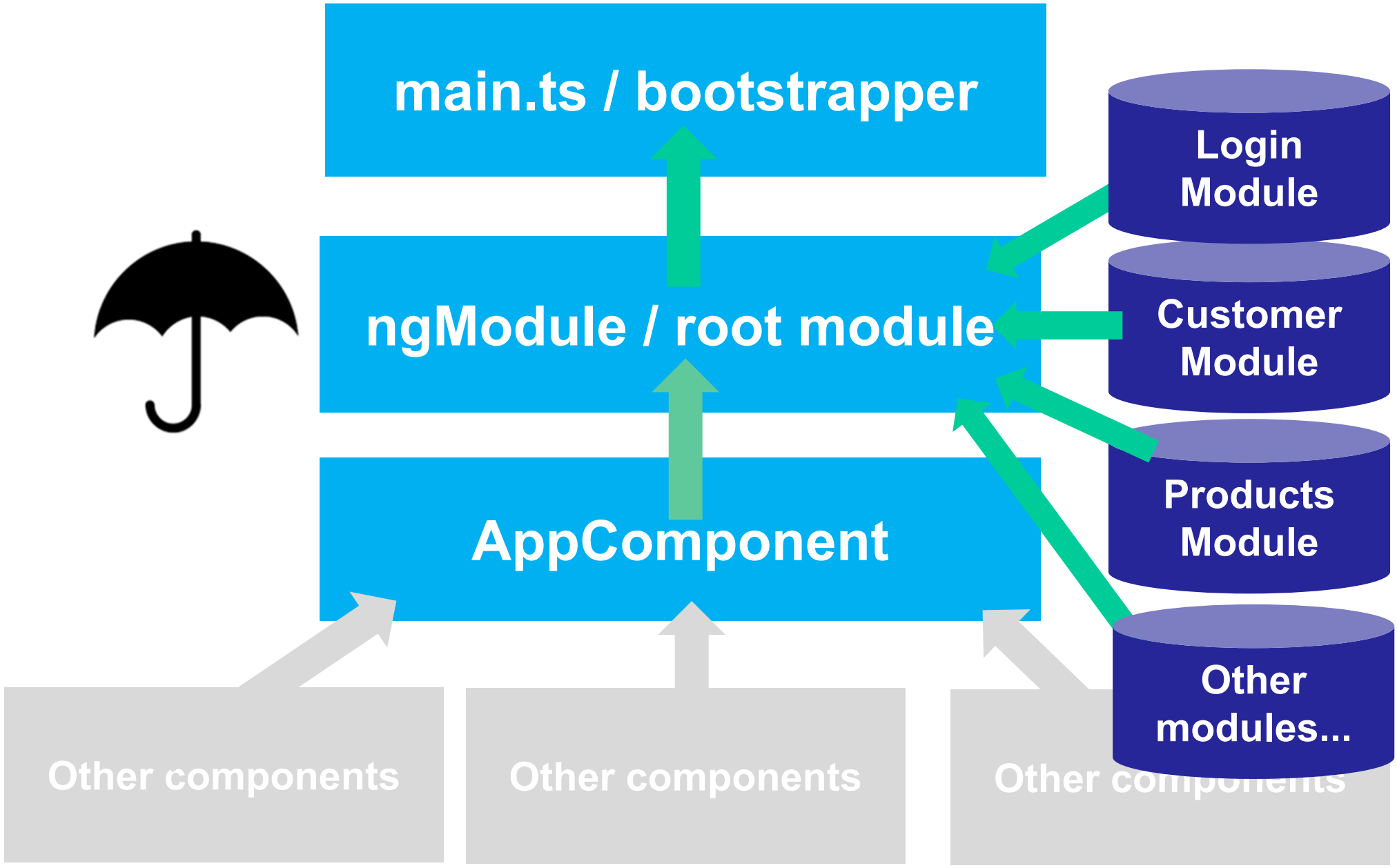
- Use one `AppModule` - the root of your app
- Use one `CoreModule` - containing all *singletons* in your app
- Use one `SharedModule` - containing all shared resources, possible multiple instances
- Use additional modules *per feature*
- https://www.youtube.com/watch?v=YxK4UW4UfCk



# Application – multiple Modules – why?

- *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

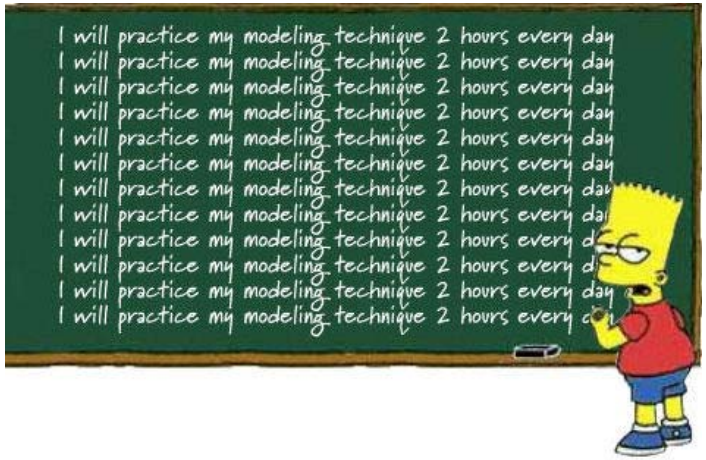
- Reuse components in multiple modules? Use a SharedModule
  - `ng g m shared` – shorthand notation
- Create components inside SharedModule
- Import SharedModule in other modules
- It doesn’t have to be in AppModule if you don’t use it directly!
- It *does* not add size to module bundles


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

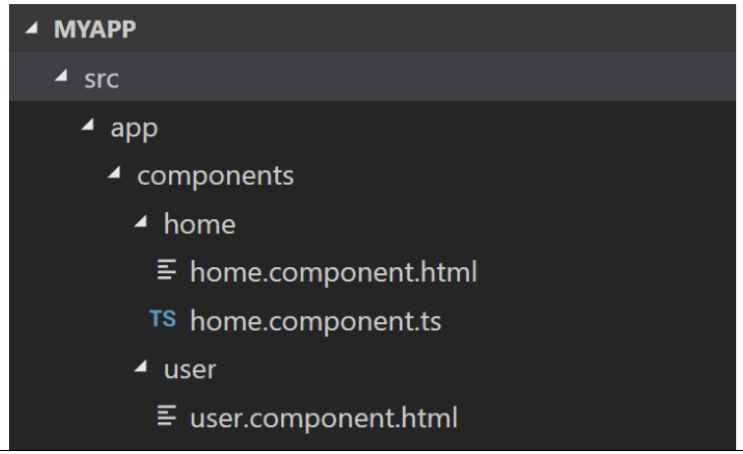
## Why and how to structure Features in Modules in Angular

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