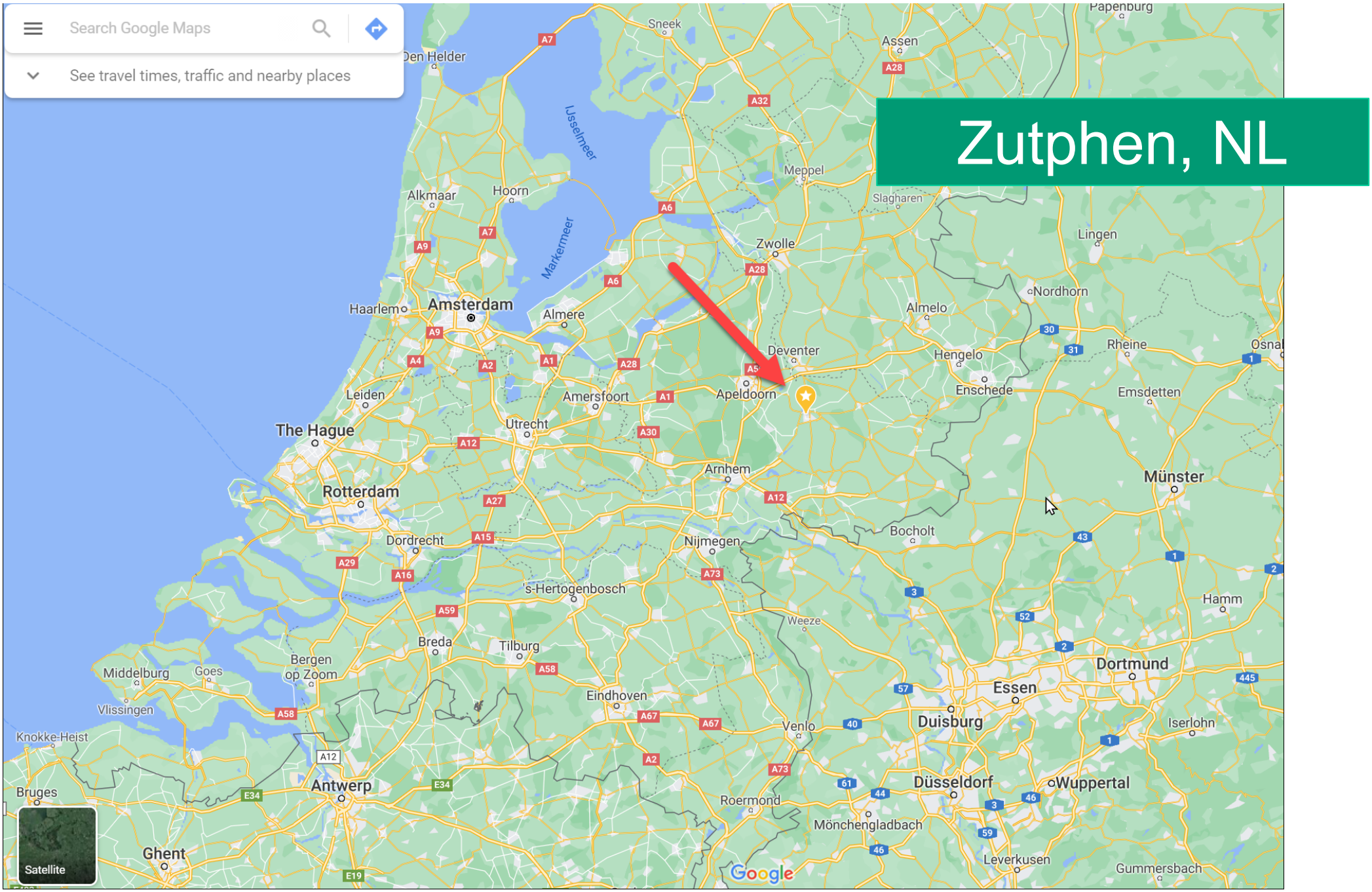
Rabobank

Angular Advanced
Introduction, Architecture

Peter Kassenaar
info@kassenaar.com


Search Google Maps
See travel times, traffic and nearby places
Zutphen, NL
Amsterdam
Rotterdam
The Hague
Utrecht
Arnhem
Nijmegen
Apeldoorn
Deventer
Zwolle
Amersfoort
Enschede
Münster
Dortmund
Essen
Duisburg
Düsseldorf
Wuppertal
Antwerp
Ghent
Bruges
Eindhoven
Tilburg
Breda
Dordrecht
Leiden
Haarlem
Almere
Hoorn
Alkmaar
Meppel
Assen
Lingen
Nordhorn
Almelo
Hengelo
Rheine
Emsdetten
Bocholt
Hamm
Iserlohn
Venlo
Roermond
Mönchengladbach
Leverkusen
Gummersbach
s-Hertogenbosch
Weeze
Bergen op Zoom
Middelburg
Vlissingen
Goes
Knokke-Heist
IJsselmeer
Markermeer
Satellite
Google

Peter Kassenaar

- Trainer, author, developer – since 1996
- Specialty: *“Everything JavaScript”*
- JavaScript, ES6, Angular, NodeJS, TypeScript, jQuery, Vue.js, React

www.kassenaar.com

info@kassenaar.com

VANDUUREN MEDIA

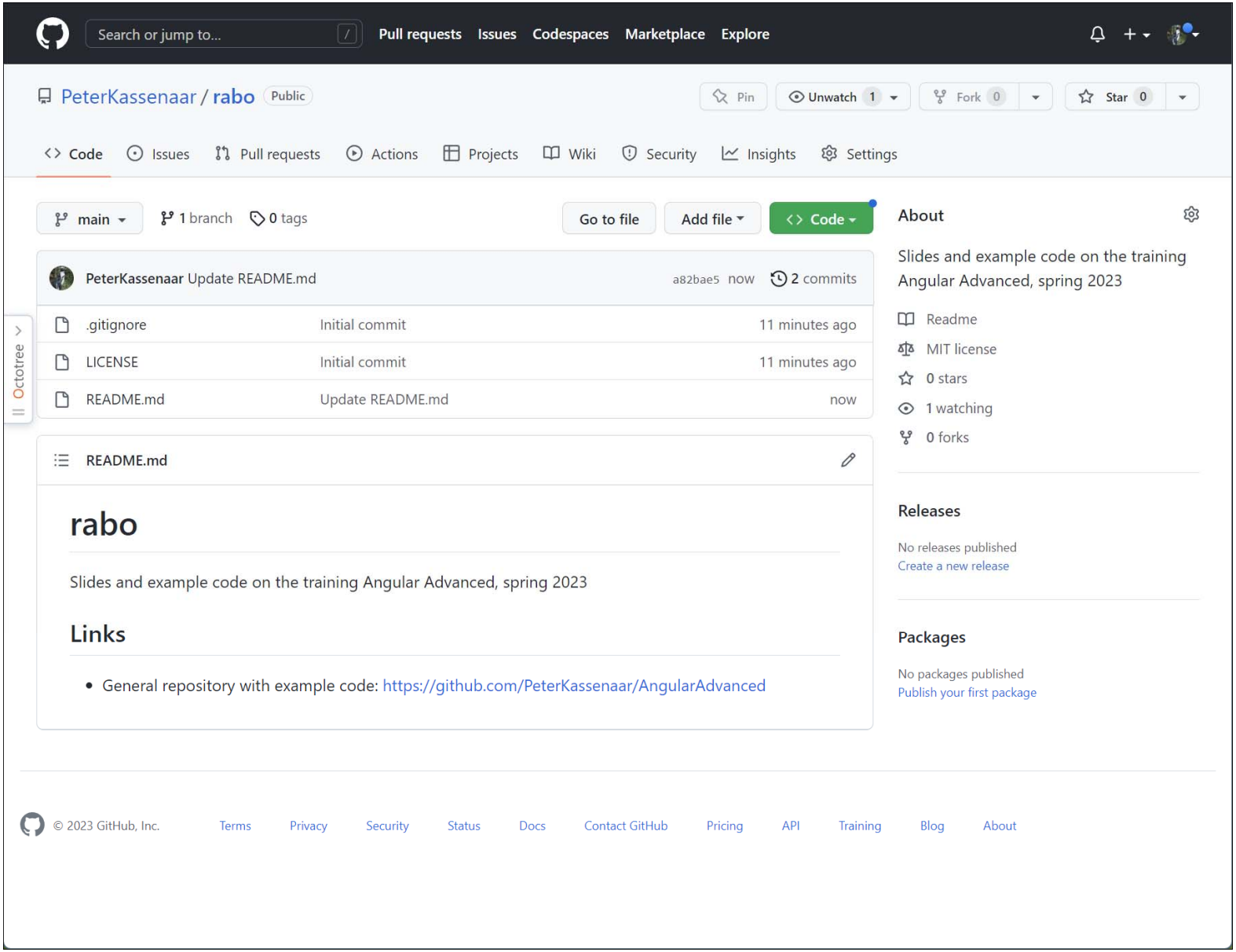
Twitter: @PeterKassenaar

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

github.com/PeterKassenaar/rabo



About you…

Introduce yourself shortly

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?

Agenda – 6, 7, 8 February 2023

~09:00 start – Morning session

~ 10:00, 11:00 Short Break

~12:00 Lunch

~12:45 Afternoon session

~ 14:00, 15:00 Break

~16:00-16:15 - end

Wednesday: if possible,
wrap up a bit early

Material

Software	(Angular + Editor + Browser + libraries)
Handouts	(PDF, Github)
Workshops	(in the presentations)
Websites	(online)

angular.io/

Short recap

- Assumed familiar: Fundamentals
  - Concepts, context & architecture
  - Angular CLI basics
  - Components, Data binding
  - Services
  - Live API’s
  - Component communication / event buses
  - Routing
    - Basics
    - Routing Parameters

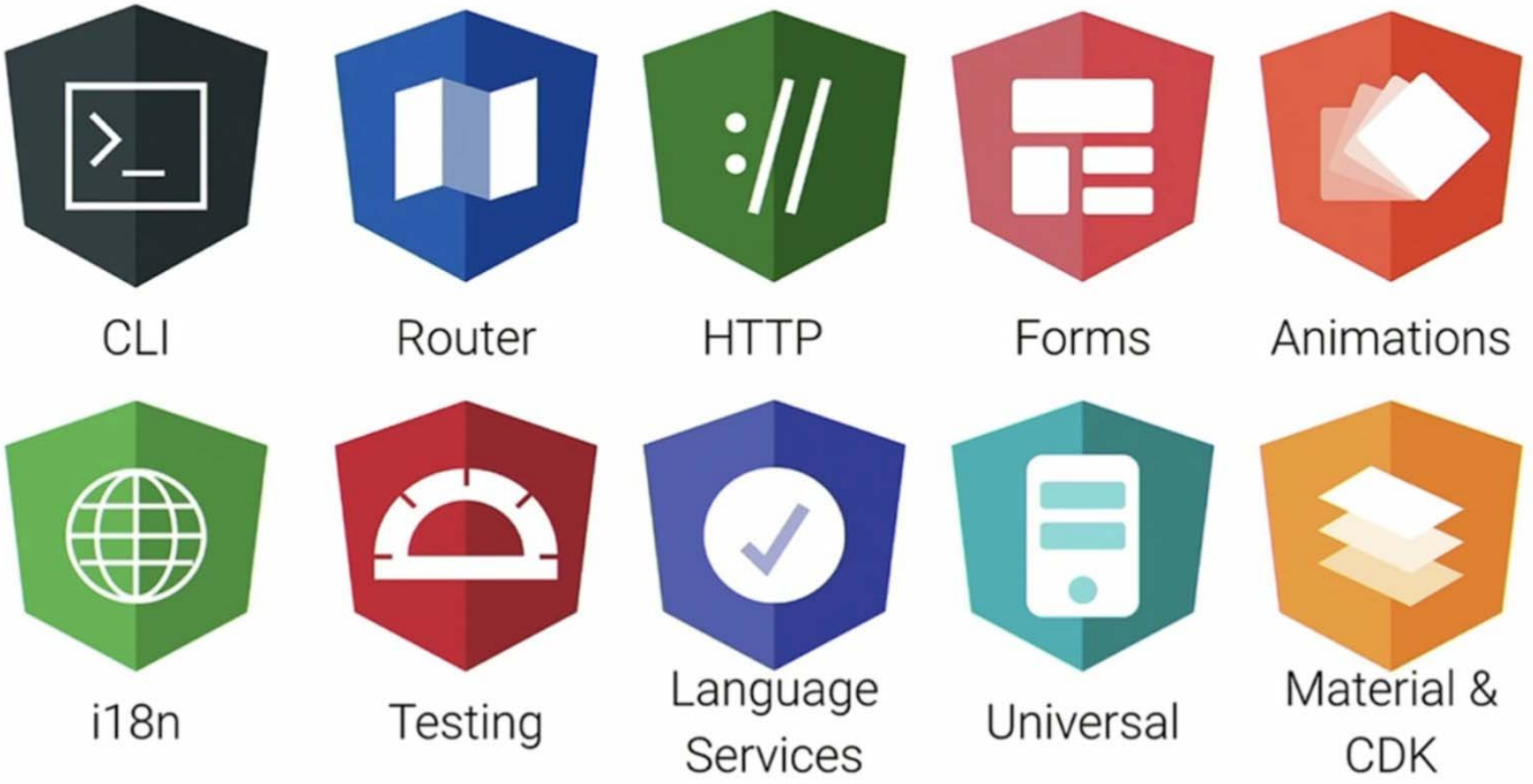
“Advanced”

Broadening?

or…

deepening?

Why devs and enterprises like Angular…



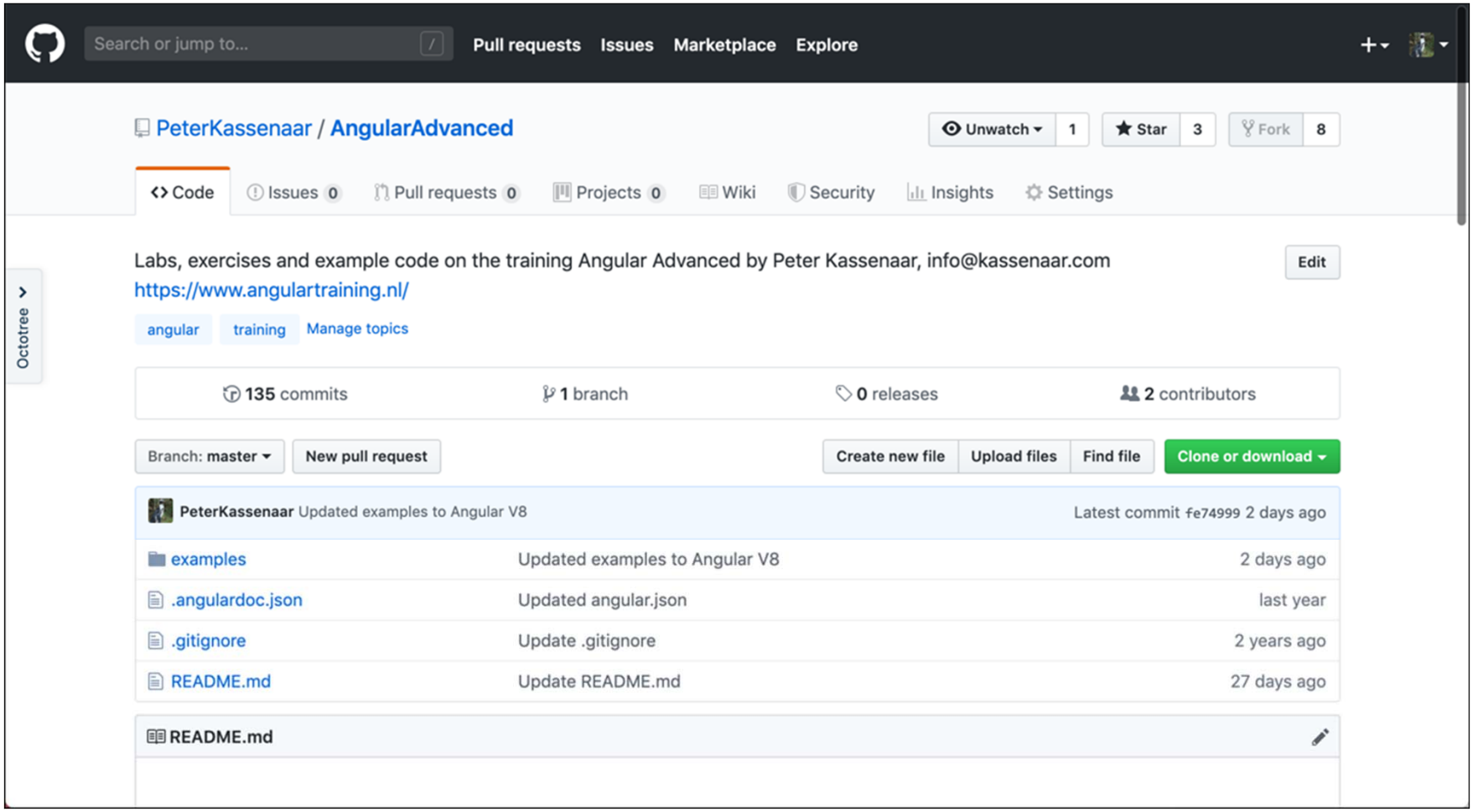
Agenda - 3 days - Thematic

- Day 1: Architecture
  - Angular CLI (tips & tricks)
  - Composing Applications with multiple modules
  - Routing and lazy loading modules
    - Loading Strategies
  - Advanced components: Smart/View ~, Dynamic ~
- Day 2: Store & Observables
  - Creating observables from scratch using RxJS, (un)subscribing, operators
  - Introduction - @ngrx/store
  - Concepts, State, Action, Reducer, Dispatcher, Effect, Http

Agenda - 3 days- Thematic

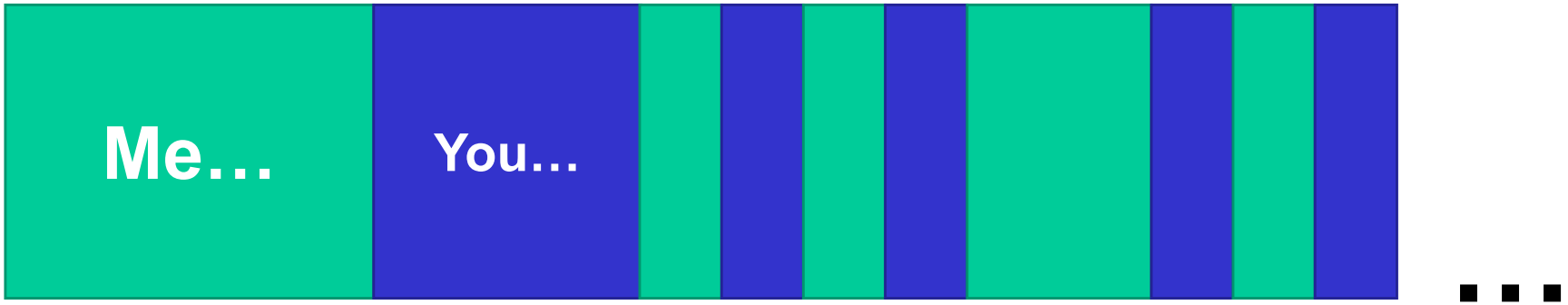
- Day 3: Miscellaneous
  - Unit testing
  - Dynamic Forms
  - Angular monorepo's / micro frontends?
  - Publishing Angular libraries to NPM
  - More on Angular Schematics
  - Your turn : Q & A, specific issues
  - …
- Overall : Best practices on coding & architecture

Generic ‘Advanced’ Github repo



https://github.com/PeterKassenaar/AngularAdvanced

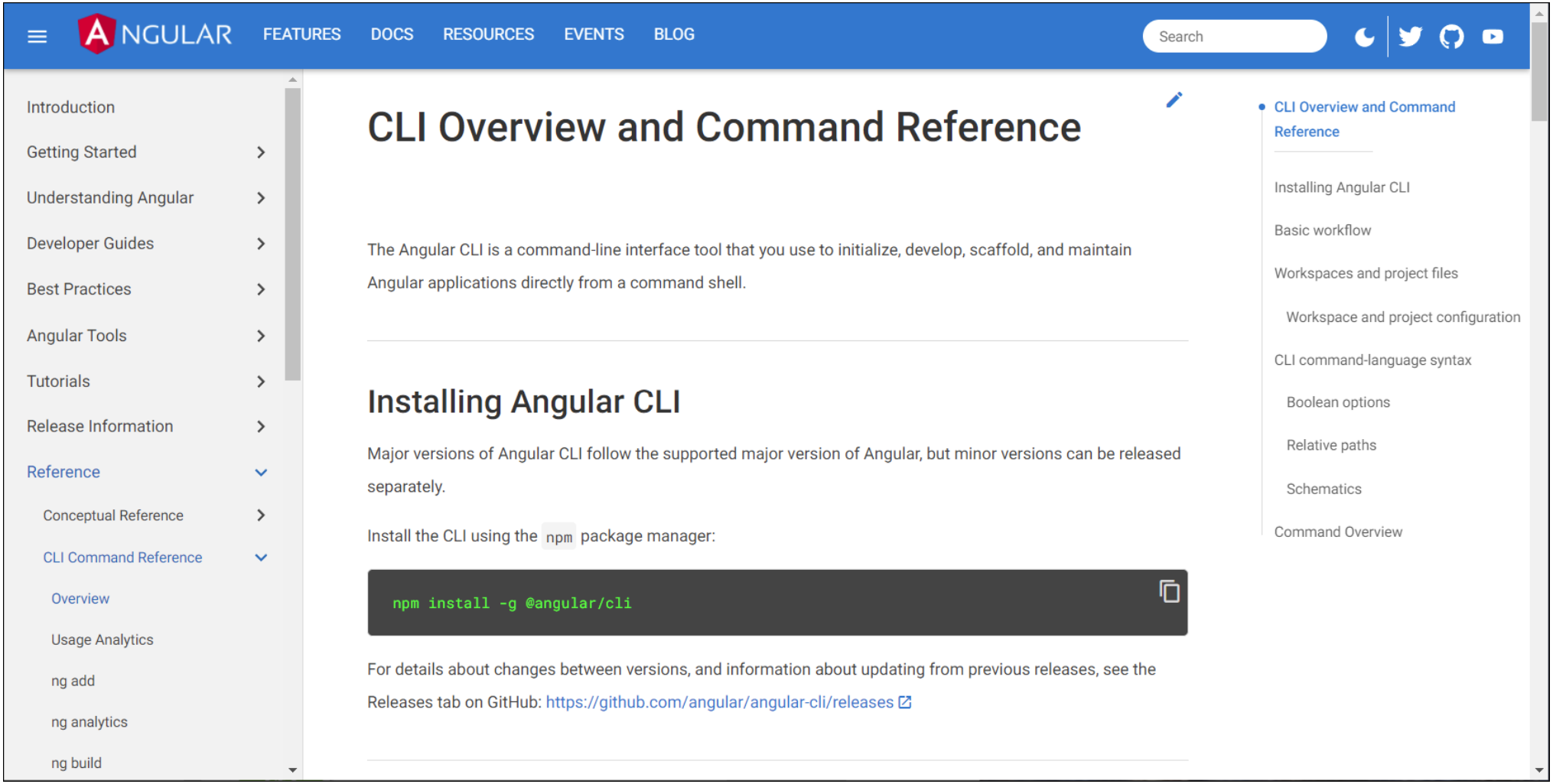
Overall process



Questions?

Angular CLI

Scaffold new projects, modules, components via command line...



https://angular.io/cli

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
  - ...

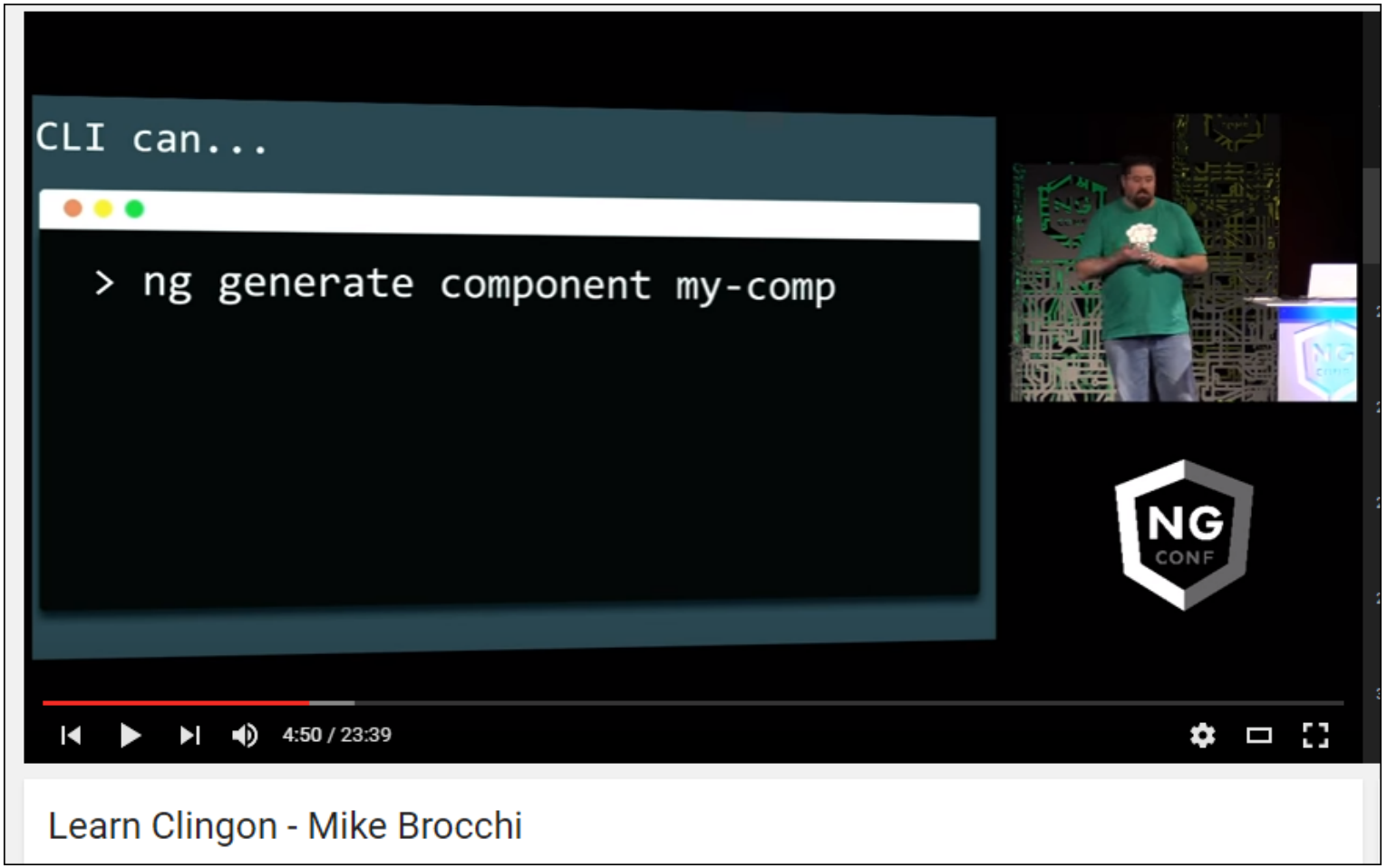
https://cli.angular.io

Install globally

```
npm install -g @angular/cli
```

CLI is also installed locally in project

```
npm start [--options]
```



https://www.youtube.com/watch?v=wHZe6gGI5RY

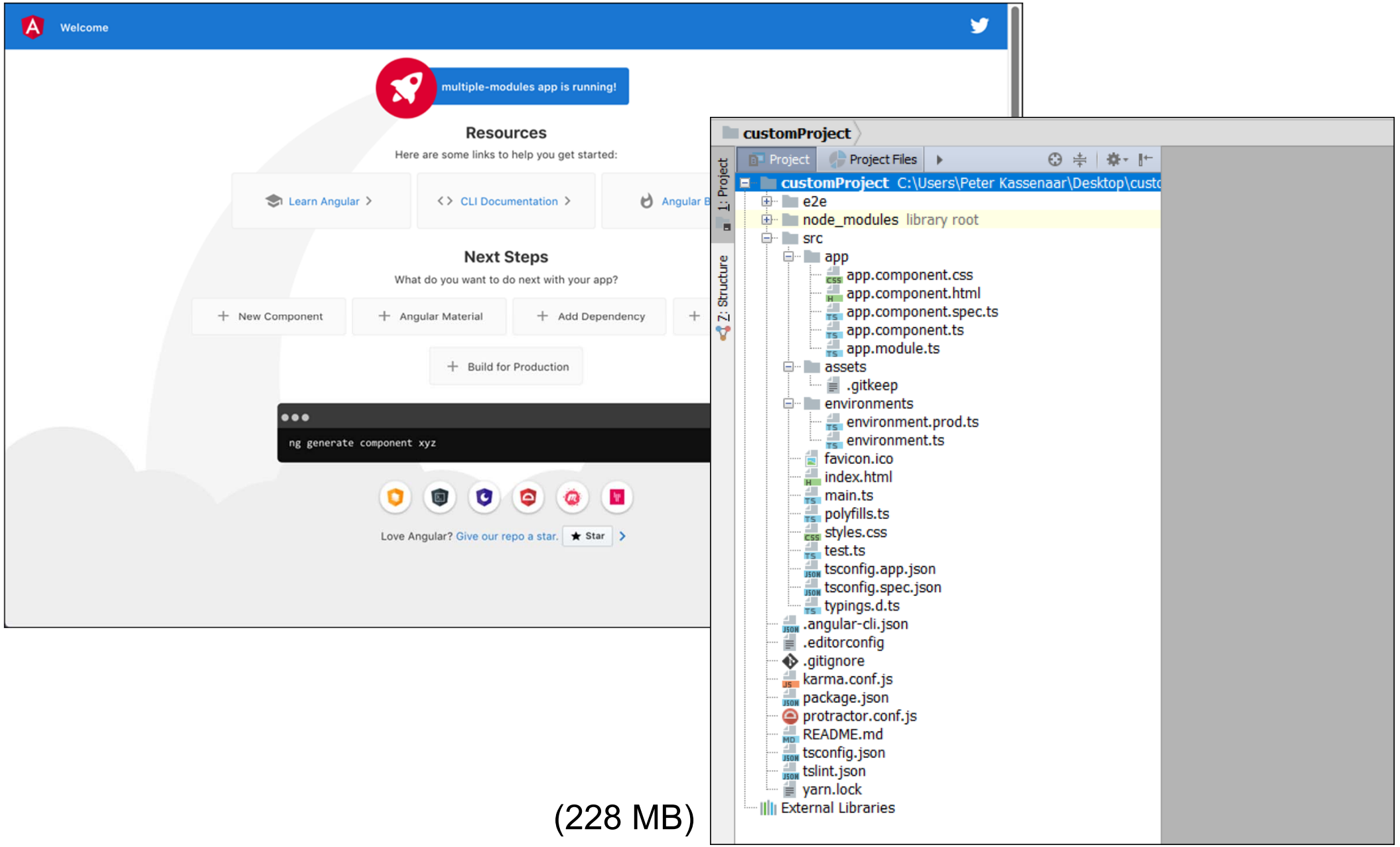
Main commands

ng new – create basic app

```
ng new PROJECT_NAME
cd PROJECT_NAME
ng serve // global command, npm start for local command
```

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

Default application



(228 MB)

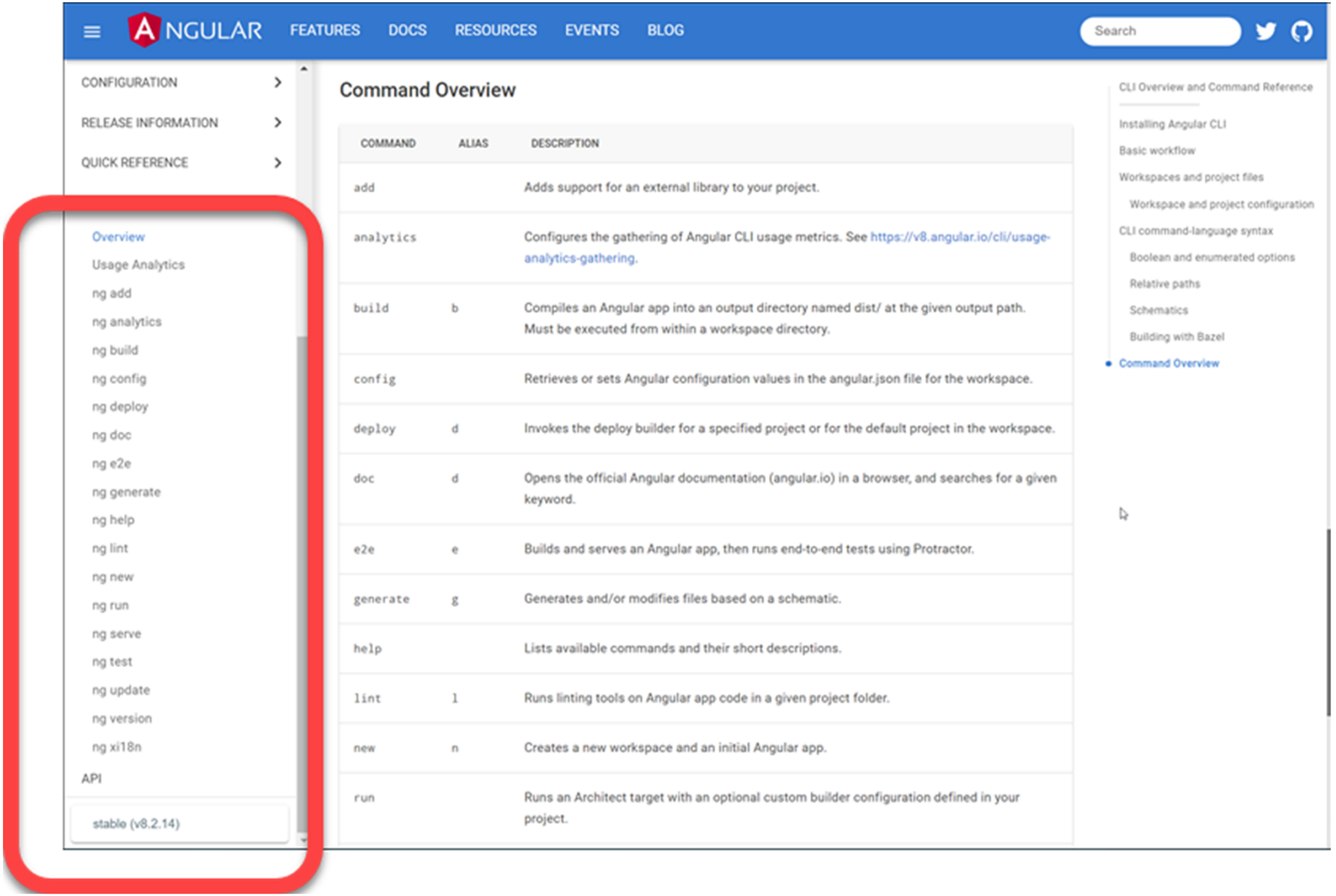
Some CLI tips & tricks

- `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

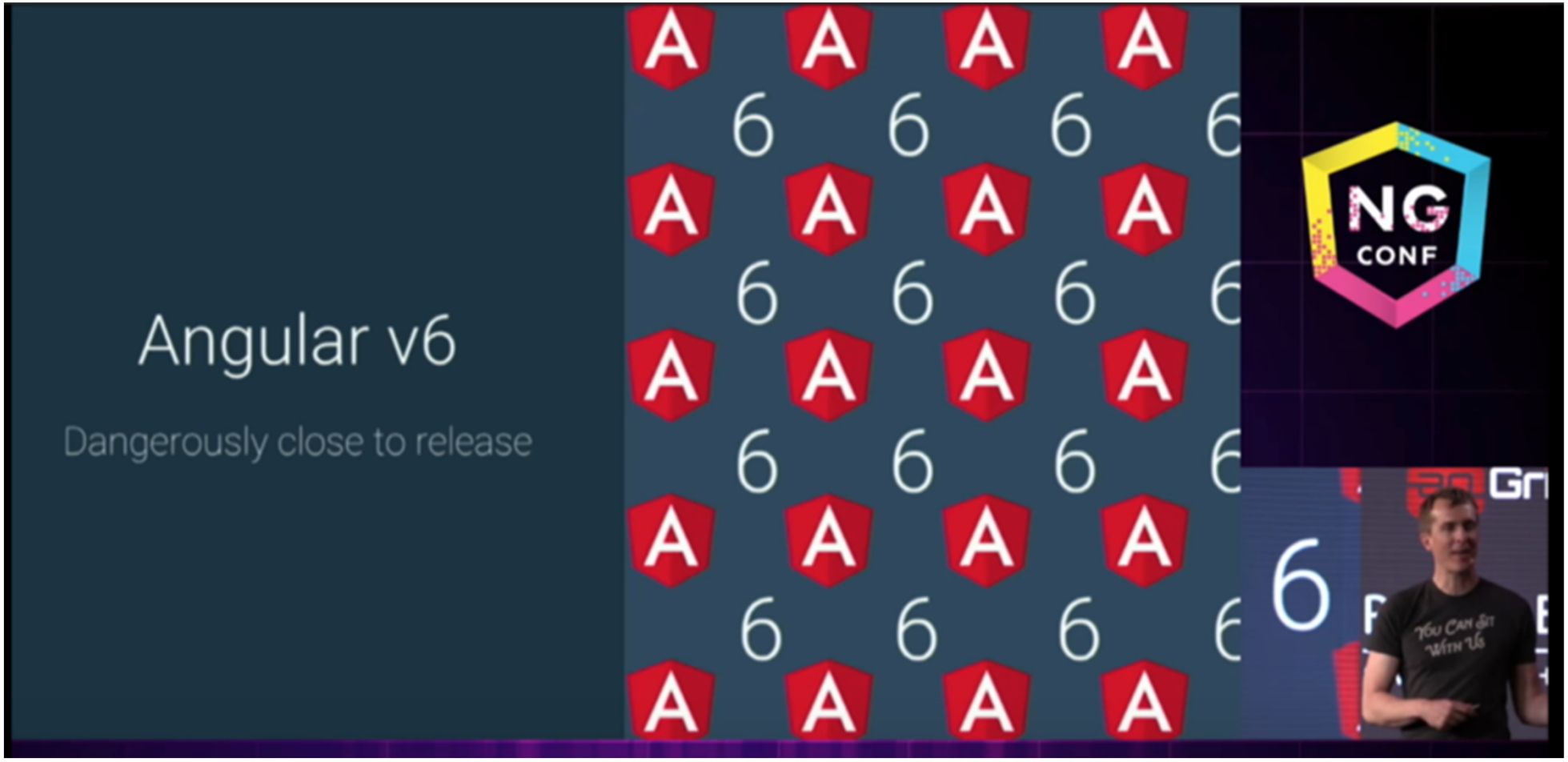
More ng tooling

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

Lots (!) of options

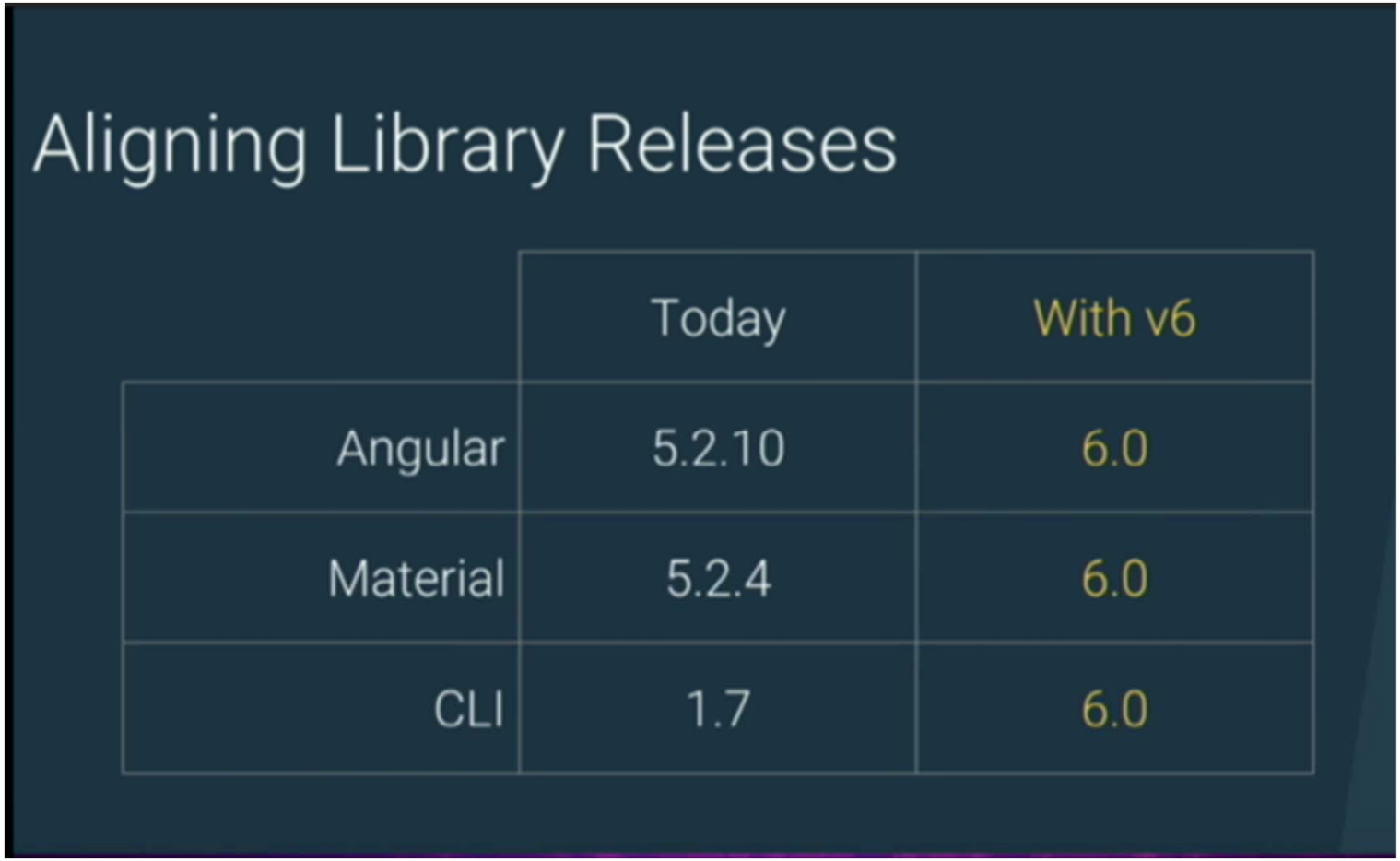


Angular CLI – modern version (May 2018)

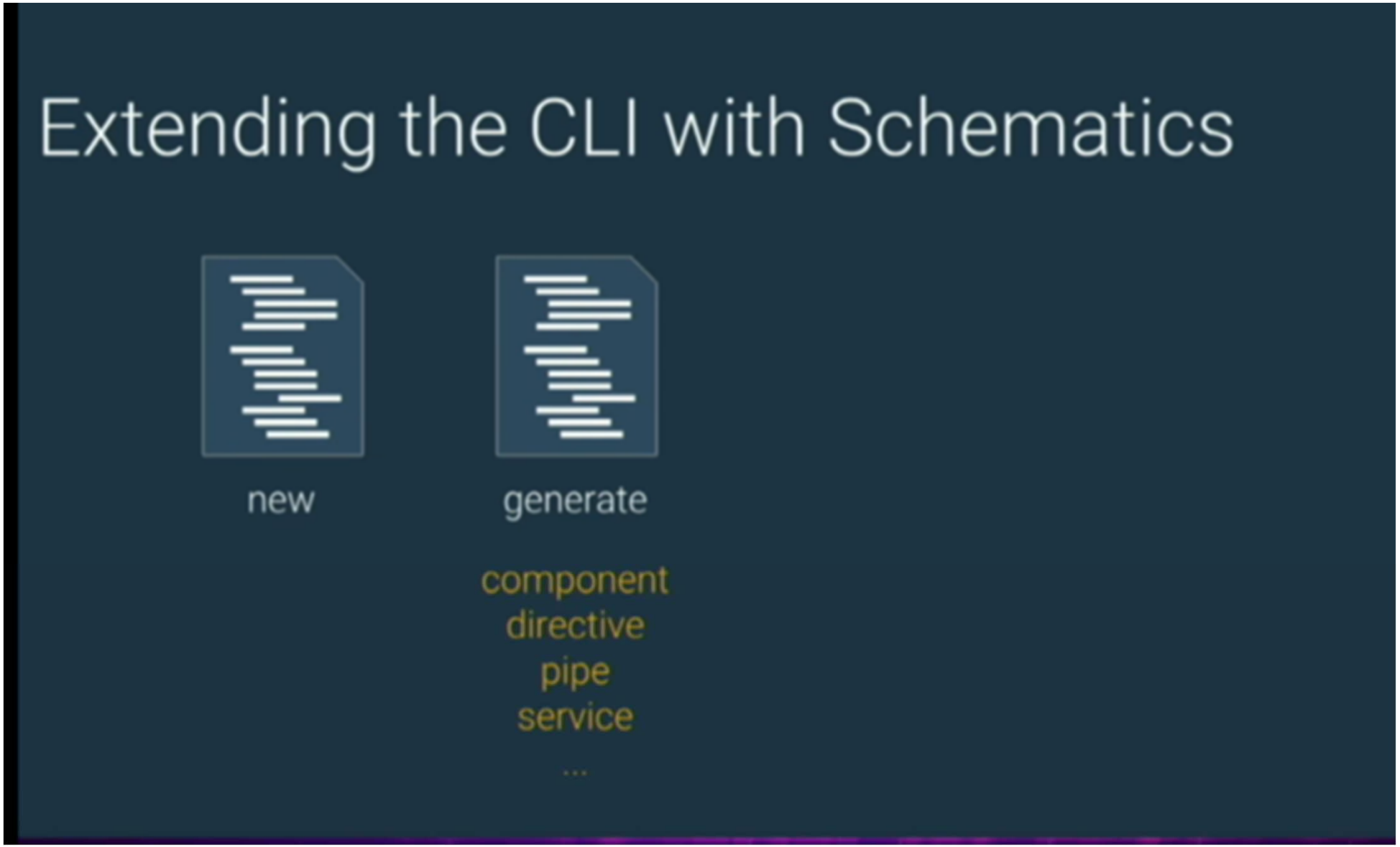


https://www.youtube.com/watch?v=dIxknqPOWms

About version numbering



New CLI Options



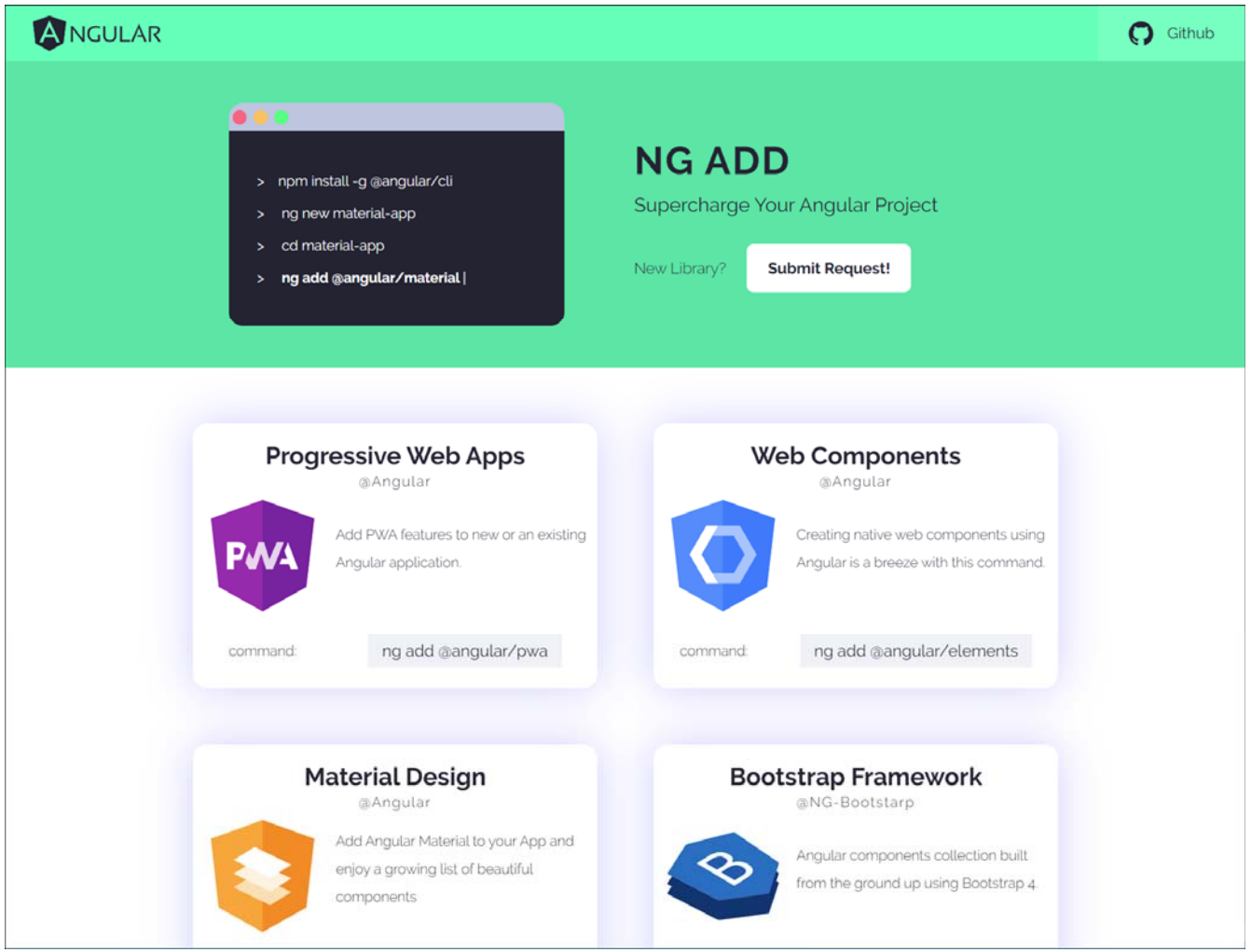
Extending the CLI with Schematics

component
directive
pipe
service
...

Info on the Angular CLI keynote

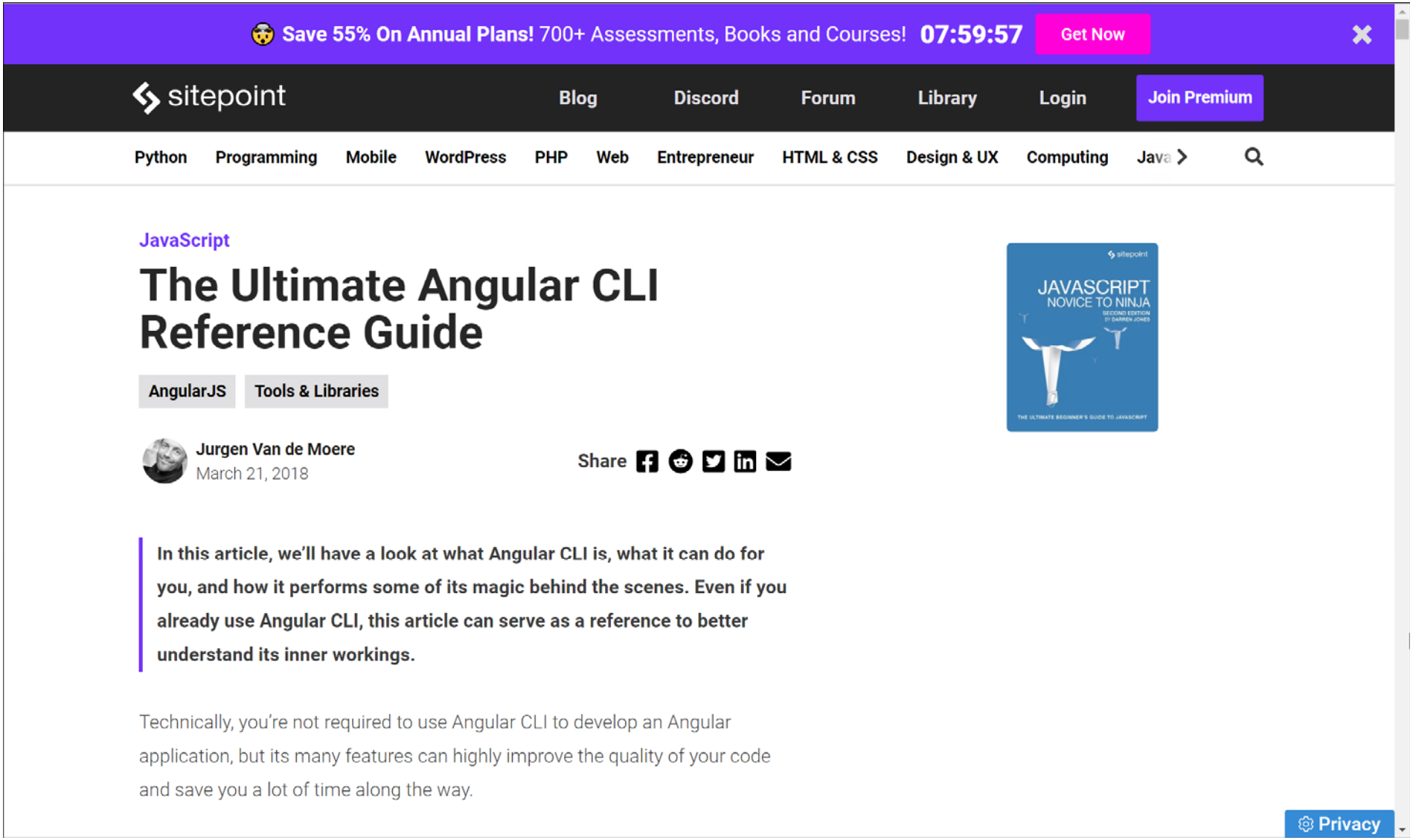
https://www.youtube.com/watch?v=dIxknqPOWms

Ng add overview



https://ng-add.web.app/

If you want to learn more - background information

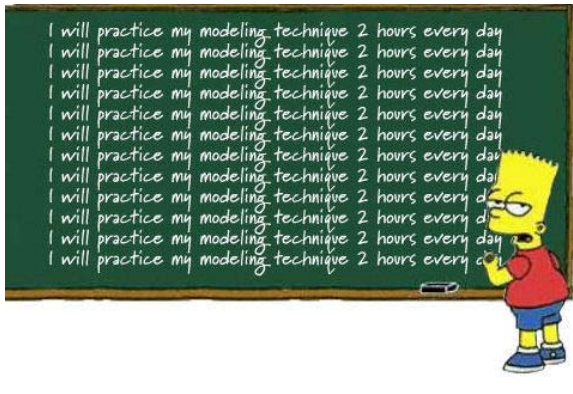


https://www.sitepoint.com/ultimate-angular-cli-reference/

Mini Workshop

- Generate a new, blank project with Angular CLI
- Generate a new component or a new service with it
- Add some new CLI packages, using `ng add` for instance
  - `@angular/material`
  - `ngx-bootstrap` (https://valor-software.com/ngx-bootstrap/#/)
  - See how/where they are installed
  - What files are affected?
  - https://ng-add.web.app/
- Study some additional CLI/ ng commands at https://angular.io/cli

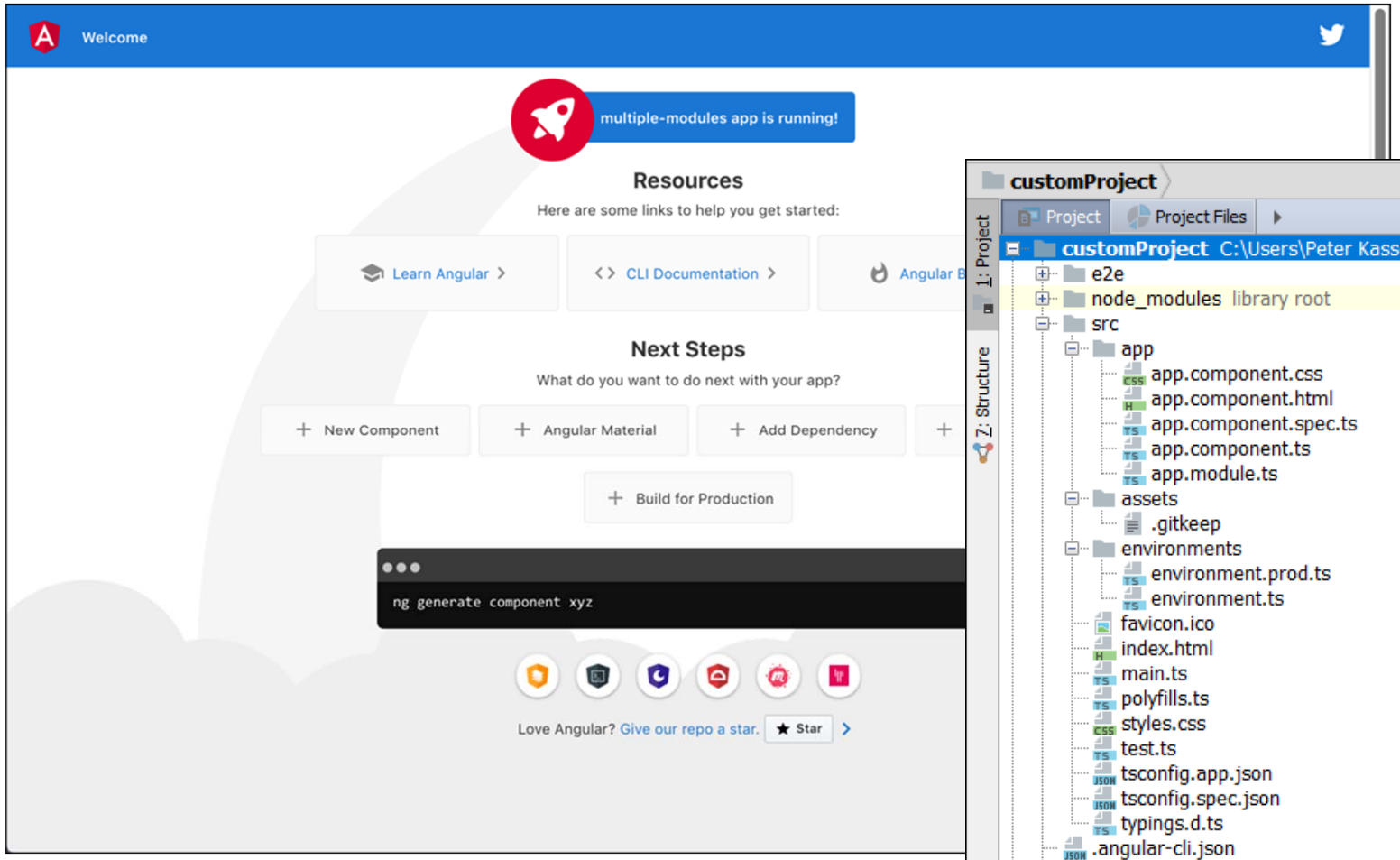
  For instance:
  - What can you generate with `ng generate`?
  - What options are available to `ng build`?
  - What other commands do seem useful to you?

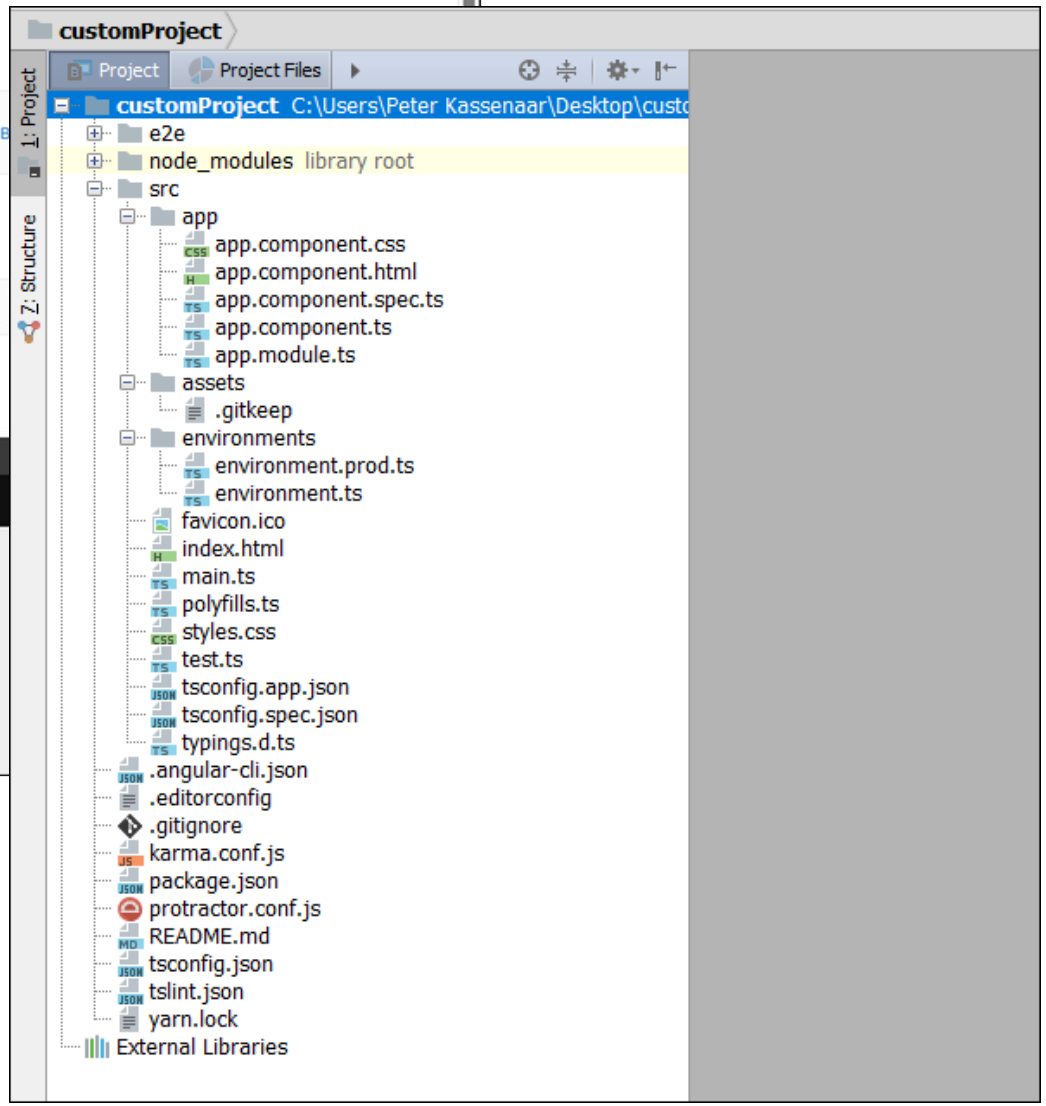


Multiple modules

Splitting your application into separate, reusable modules

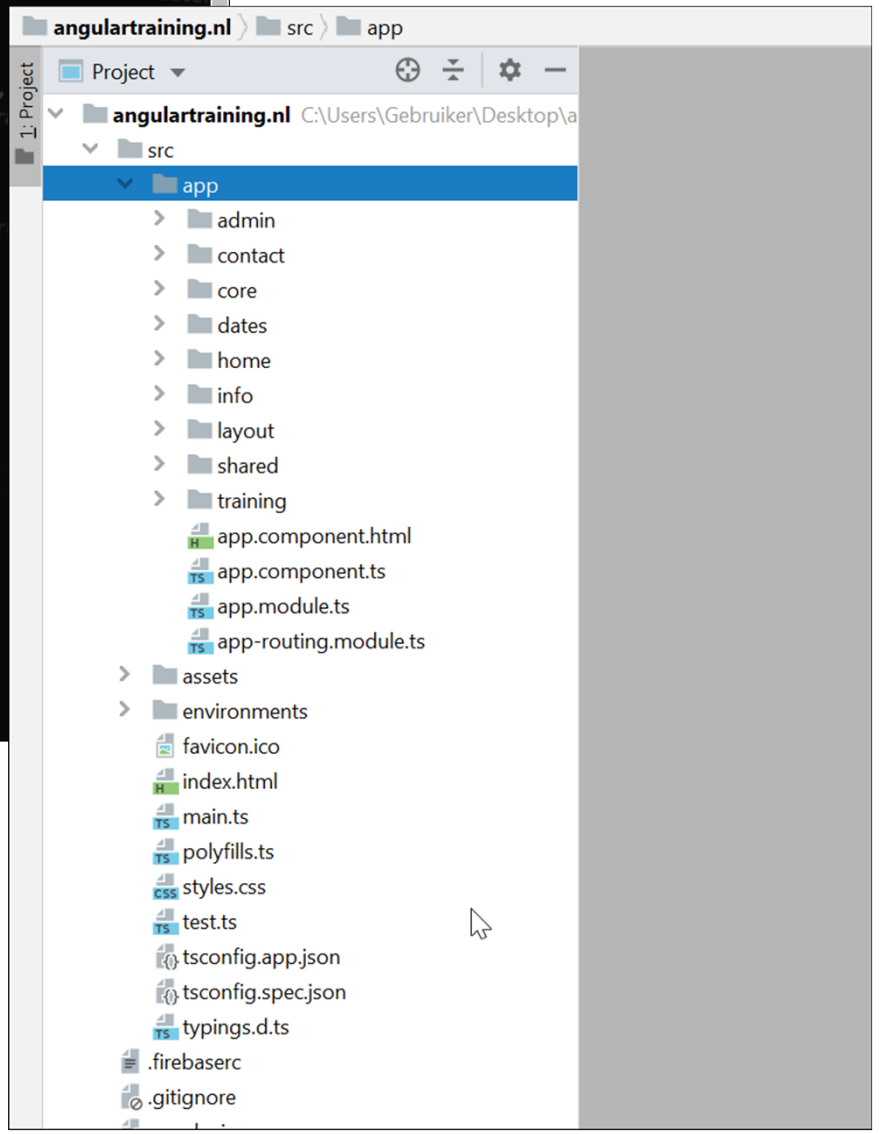
Default application – 1 module





(228 MB)

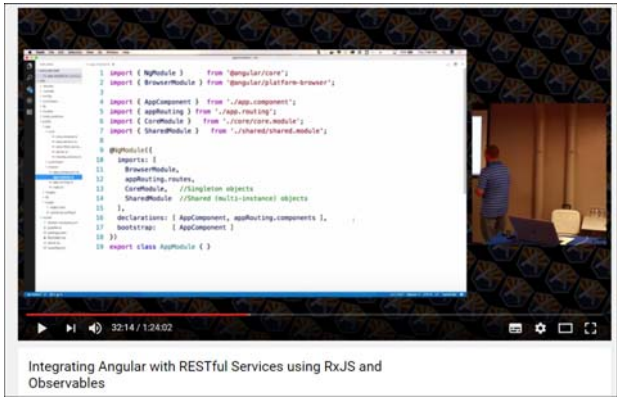
Bigger applications – multiple modules



Angular Modules - naming

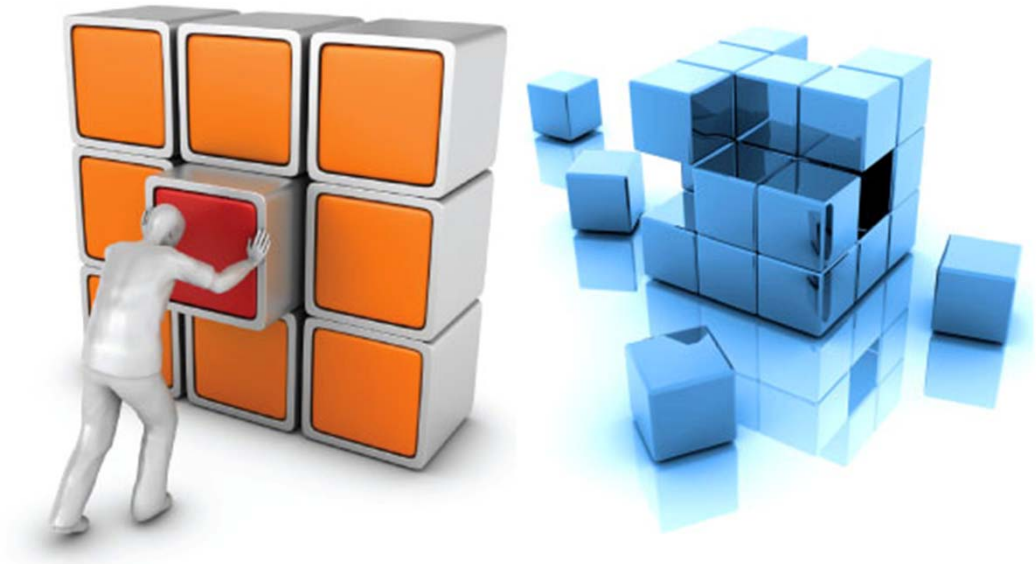
- Divide your app into *logical* and often *reusable* pieces of code
- Keyword : code organization

- 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.

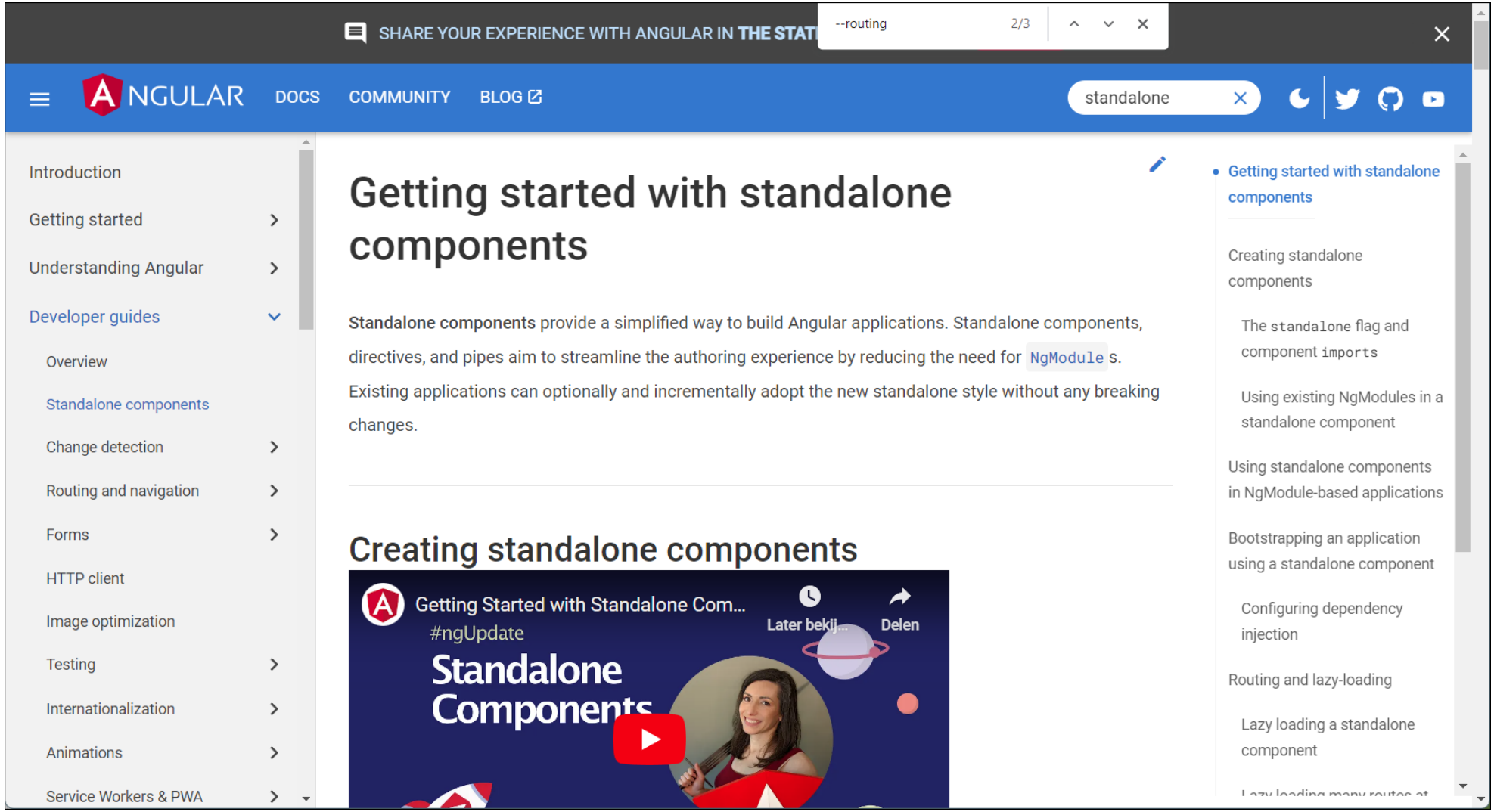


Since Angular 15 – Standalone Components

- Applications and components without an `ngModule`

"Standalone components provide a simplified way to build Angular applications. Standalone components, directives, and pipes aim to streamline the authoring experience by reducing the need for `NgModules`. Existing applications can optionally and incrementally adopt the new standalone style without any breaking changes."

More info on standalone components



https://angular.io/guide/standalone-components

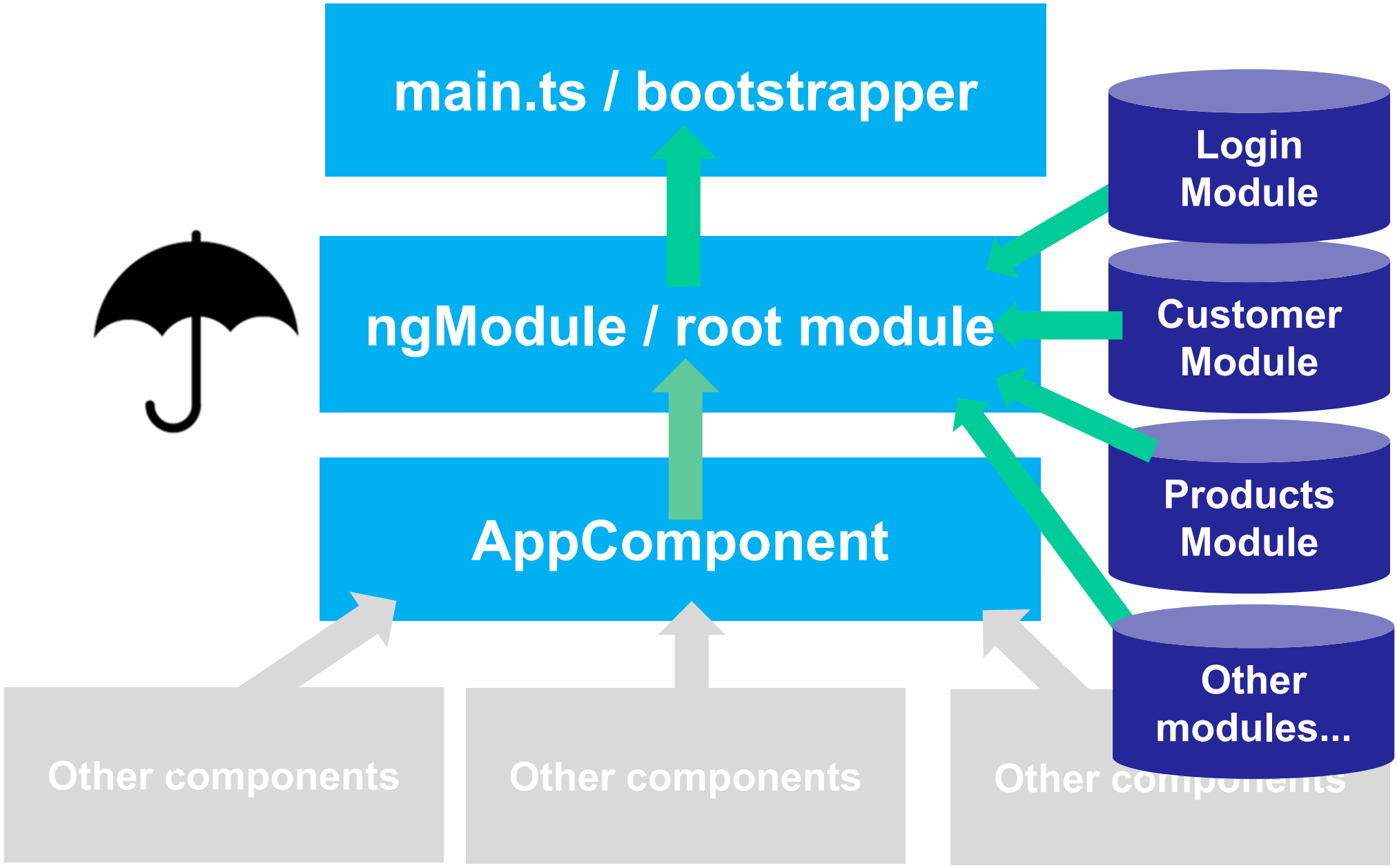
Verdict (personal, opinion!)

We're NOT using standalone components in this course

- Standalone components might come in handy, in new applications from scratch
- Due to confusion, personally I don't see it implemented *'optionally and incrementally'* in existing apps
- It is mostly done (IMO) to mimic Vue, React et all.

But, if you want more information…

- https://blog.angular.io/angular-v15-is-now-available-df7be7f2f4c8
- https://www.angulararchitects.io/aktuelles/angulars-future-without-ngmodules-lightweight-solutions-on-top-of-standalone-components/
- https://netbasal.com/angular-standalone-components-welcome-to-a-world-without-ngmodule-abd3963e89c5


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 `customers.module.ts`
   - `exports : [CustomersComponent],`
   - Otherwise it can’t be used in other components!
5. Provide new module to `app.module.ts`
   - `imports: [CustomersModule]`

Optional : SharedModule

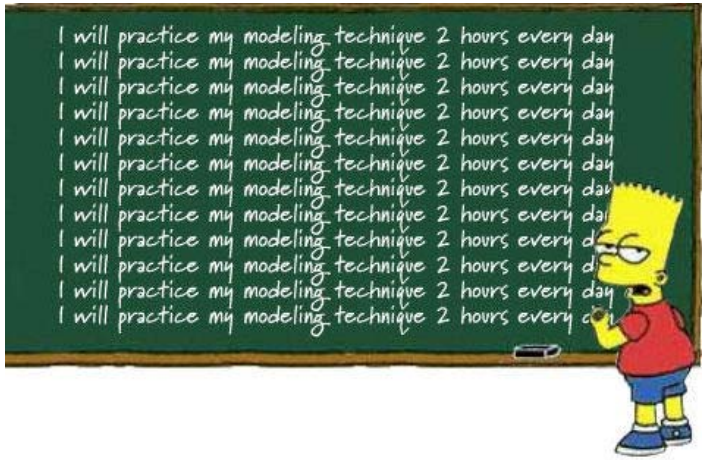
- Reuse components in multiple modules? Use a `SharedModule`
  - `ng g m shared` – shorthand notation
- Create components inside `SharedModule`
- Import `SharedModule` in other modules
- It doesn't have to be in `AppModule` if you don't use it directly!
- It *does* not add size to module bundles


Demo

Workshop

- Open `../100-multiple modules` (`npm install`, `npm start`)
- 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

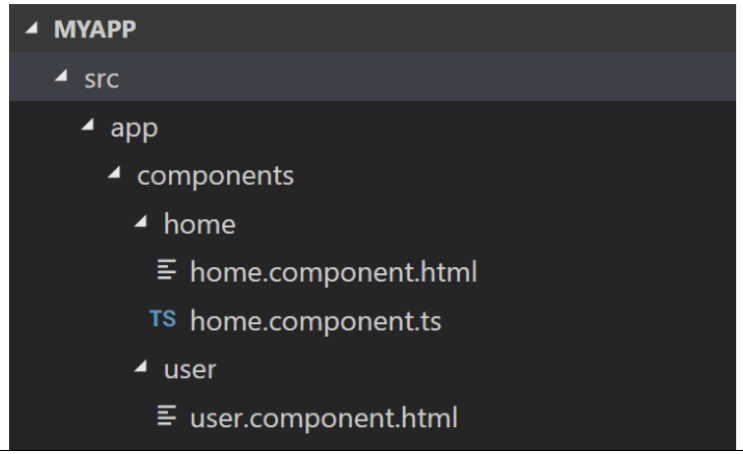
Why and how to structure Features in Modules in Angular

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