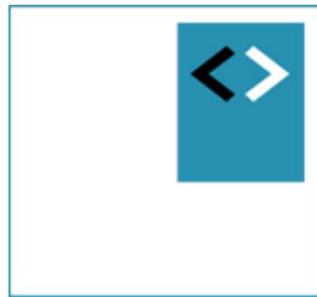


# WARMTEBOUW.

## *Angular Advanced* Introduction, Architecture



Peter Kassenaar  
[info@kassenaar.com](mailto:info@kassenaar.com)

# Peter Kassenaar

- Trainer, author, developer – since 1996
- Specialty: *"Everything Frontend"*
- JavaScript, ES6, Angular, NodeJS, TypeScript, jQuery, Vue.js, React, Flutter, Ionic

[www.kassenaar.com](http://www.kassenaar.com)

[info@kassenaar.com](mailto:info@kassenaar.com)

VANDUUREN  
MEDIA



ING

OHRA

euricom  
A DIMENSION DATA COMPANY

s a n o m a

delta lloyd

zenito  
BETERE ZEKERHEID  
VOOR ONDERNEMERS

Atos



OBERON INTERACTIVE

woonbron

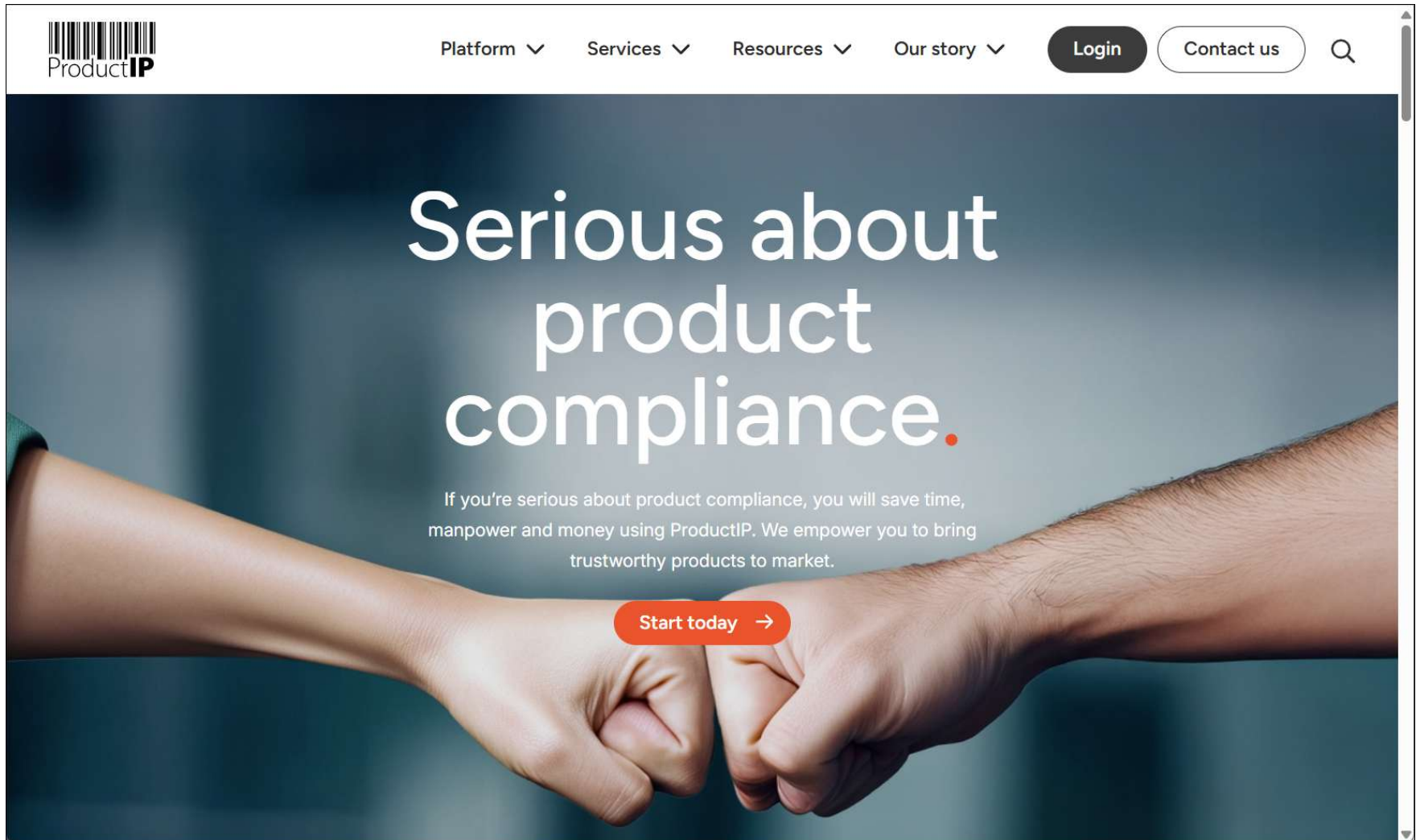
ROC West-Brabant

the eforum  
FACTORY





# Senior frontend developer ProductIP, (50%)



[www.productip.com](http://www.productip.com)



[peterkassenaar](#)



[PeterKassenaar](#)



[PeterKassenaar](#)



[pkas06](#)

CLS TRAININGEN

085 0600 151

Zoeken...

OPLEIDINGEN ▾LEERVORMENMAATWERK & IN COMPANYLAST MINUTESTIPS & TRICKSCONTACT

Meer dan 500 opleidingen  
in heel Nederland

Alle trainingen kunt u ook online volgen. Live, met trainer en vanuit huis!

Trainingsaanbod ▾

Zoek op trefwoord

## Aanbod CLS Trainingen


Ons huidige aanbod

CLS Trainingen maakt gebruik van cookies. Wanneer u doorgaat met het gebruiken van de website, gaan wij er vanuit dat u akkoord gaat met ons cookiebeleid.

akkoord

Ons cookiebeleid

<https://cls.nl/>



Angular Advanced

# CLS TRAININGEN

085 0600 151 Zoeken

Angular Advanced

OPLEIDINGEN ▾LEERVORMENMAATWERK & IN COMPANYLAST MINUTESTIPS & TRICKS

TERUG NA

Angular is een platform dat het mogelijk maakt om gemakkelijk webapplicaties te bouwen. Angular combineert verklarende templates, dependency injection, end to end tooling en best practices om ontwikkelingsuitdagingen op te lossen. Met behulp van Angular kunnen applicaties worden gebouwd voor zowel mobiel als desktop.

Omschrijving


Inhoud

Plaats en data

Tijdens de cursus komen de volgende onderwerpen uitgebreid aan bod:

- Angular-applicaties met meerdere (feature) modules;
- routing en lazy loading;
- named router outlets;
- authentication en routing guards;
- advanced components – content projection
- unit testing met Jasmine en Karma;
- smart component/view component design pattern;
- state management met @ngrx/store;
- optioneel: Angular en real-time applicaties met Firebase;
- optioneel: PWA's met Angular;
- optioneel: third party-libraries gebruiken (Angular Material, Ionic, PrimeFaces, Angular Maps, etc).

Angular Advan



DIRECT I

### Deze cursus in

- Locaties door he
- Professionele do
- Inclusief certifica
- 9.4 op Springest
- Ook bij u op loca

# Warmtebouw “Wish List”

## Wensenlijst

Hier zijn wat onderdelen waar we wat meer over willen weten.

- signal(), computed(), linkedSignal(), effect()
- Signals vs Behavioursubject
- Signal-based inputs, outputs & queries
- resource(), httpResource(), rxResource()
- Routing, lazy loading & functional guards
- DestroyRef + takeUntilDestroyed()
- @for met track, @if, @switch, @defer
- Smart & dumb components
- Zoneless change detection
- Nieuwe Angular 20 features (selectorless, signal forms, host type checking)
- RxJS interop: toSignal(), toObservable()
- Functional interceptors
- Standalone architecture
- Valkuilen en Performance
- Angular 21+



# [github.com/PeterKassenaar/warmtebouw](https://github.com/PeterKassenaar/warmtebouw)

PeterKassenaar / warmtebouw

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

warmtebouw Public

Pin Watch 0 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

File	Commit Message	Time
.gitignore	Initial commit	7 minutes ago
LICENSE	Initial commit	7 minutes ago
README.md	Update README.md	6 minutes ago

README MIT license

## warmtebouw

Slides and sample code on the training Angular, Warmtebouw, February 2026

### Links

- General Angular Fundamentals Repository: <https://github.com/PeterKassenaar/angular-fundamentals>
- General Angular Advanced Repository: <https://github.com/PeterKassenaar/angular-advanced>
- ...

About

Slides and sample code on the training Angular, Warmtebouw, February 2026

- Readme
- MIT license
- Activity
- 0 stars
- 0 watching
- 0 forks

Releases

No releases published  
[Create a new release](#)

Packages

No packages published  
[Publish your first package](#)

© 2026 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

**About you...**



## **Introduce yourself briefly**

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 – 25, 26 February 2026

~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

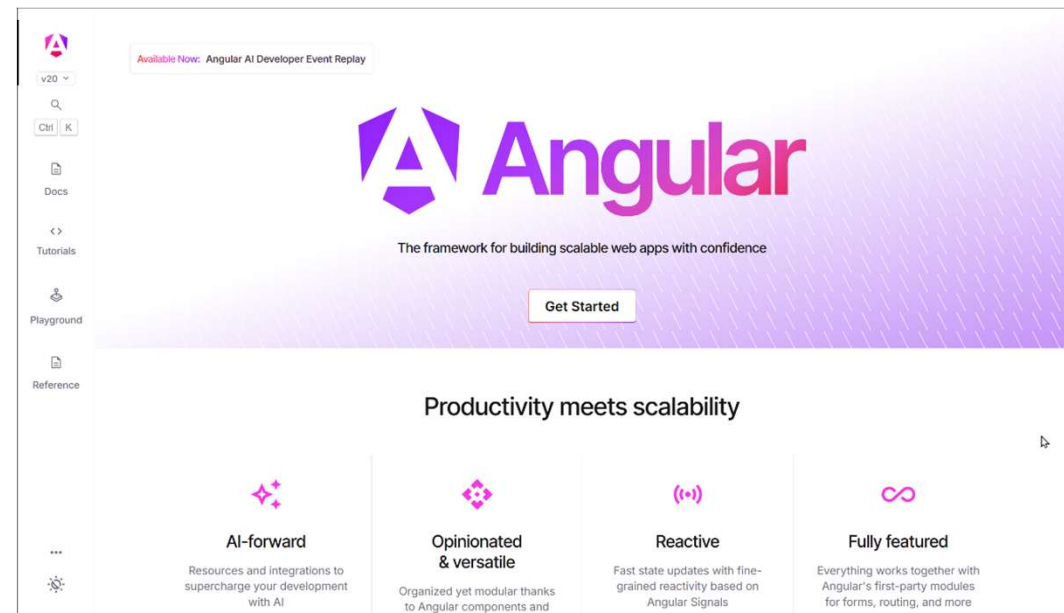
Last day: if possible, wrap  
up a bit early





# Material

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



[angular.dev/](https://angular.dev/)

# 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...



CLI



Router



HTTP



Forms



Animations



i18n



Testing



Language  
Services



Universal



Material &  
CDK

# Agenda - 2 days - Thematic

- Introduction
- Day 1: **Angular New Features + refresher**
  - Standalone components and module-less defaults
  - Template syntax: `@for`, `@if-then-else`, `@switch()`
  - Signals
    - Types of signals, computed, input/output
  - Dependency Injection
  - New `provideHttpClient()`
  - Components: Smart/View ~, content projection

# Agenda - 2 days- Thematic



- Day 2: **Miscellaneous**
  - Routing, Lazy Loading, Guards
  - Unsubscribing with `DestroyRef`
  - RXJS Patterns
  - ...
- Overall : Best practices on coding & architecture

# 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](https://github.com/PeterKassenaar/AngularAdvanced)



# Generic 'Advanced' Github repo

The screenshot shows the GitHub repository page for 'AngularAdvanced' by PeterKassenaar. The repository is public and has 19 forks and 13 stars. The main branch is 'master'. The repository contains a file tree with the following files and their commit history:

File	Commit Message	Commit Date
examples	feat: add examples on i18n	3 weeks ago
.angulardoc.json	Updated angular.json	8 years ago
.gitignore	Added Angular 13 cache directory to .gitignore	5 years ago
README.md	feat: update README.md	4 months ago

The README file is titled 'Angular Advanced' and contains the following text:

Labs, exercises and example code on the training Angular Advanced by Peter Kassenaar.

Disclaimer! The examples are not production ready, they are just for the training. Some use older versions of Node.js. They are NOT all updated to the latest version of Angular yet.

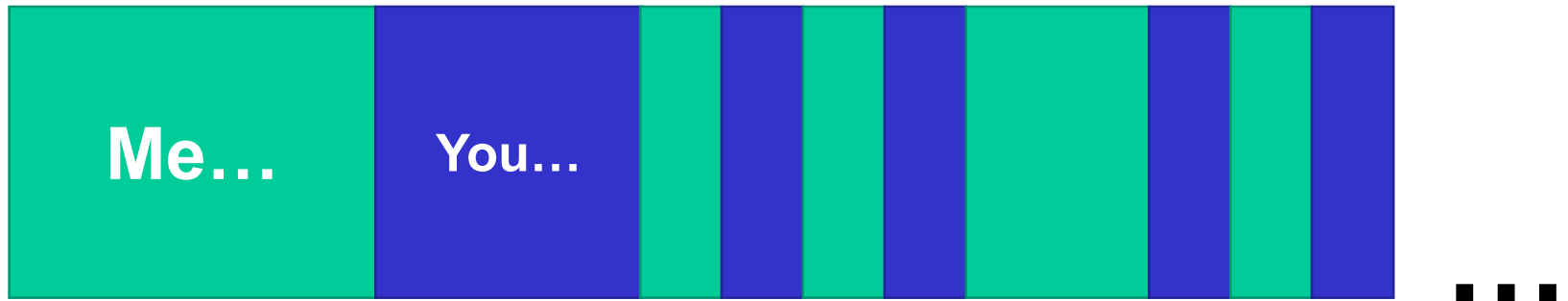
The repository also has a 'Contents' section.

On the right side of the repository page, there is an 'About' section with the following information:

- Labs, exercises and example code on the training Angular Advanced by Peter Kassenaar, info@kassenaar.com
- Website: [www.angulartraining.nl/](http://www.angulartraining.nl/)
- Tags: training, angular
- Readme, Activity, 13 stars, 1 watching, 19 forks
- Contributors: 2 (PeterKassenaar, herwinw)

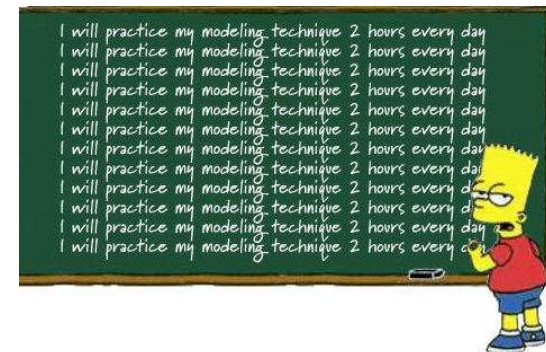
<https://github.com/PeterKassenaar/AngularAdvanced>

# Overall process



# On Workshops...

- ... are designated with a slide like this 🙌
- Are **between** the talks with theory
- Are **NOT** written out line-by-line. You have to think for yourself
- The time during the training **may be too short** to finish all the workshops
  - Do them **in your own time**
  - At least you know the **concepts** the workshop is about
  - Choice – **we have more to discuss**, I let that prevale
- The **example code** often (but not always!) contains the 'solution' to the workshop. Use it! It's there for you
  - But of course it would be nice if you can work with **your own project/data**



# Questions?





# Angular CLI

Scaffolding new projects, new options

# Angular CLI

- Command Line Interface for
  - Scaffolding (`ng new`),
  - Developing (`ng generate`)
  - Testing (`ng test`)
  - Deploying applications (`ng deploy`)
- Local installation
  - SO: no online environment like Stackblitz or CodeSandbox

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

Developer Tools > Angular CLI

# The Angular CLI

The Angular CLI is a command-line interface tool which allows you to scaffold, develop, test, deploy, and maintain Angular applications directly from a command shell.

Angular CLI is published on npm as the `@angular/cli` package and includes a binary named `ng`. Commands invoking `ng` are using the Angular CLI.

[Try Angular without local setup](#) ✓

If you are new to Angular, you might want to start with [Try it now!](#), which introduces the essentials of Angular in the context of a ready-made basic online store app for you to examine and modify. This standalone tutorial takes advantage of the interactive [StackBlitz](#) environment for online development. You don't need to set up your local environment until you're ready.

### Getting Started

Install Angular CLI to create and build your first app.

[Get Started](#)

### Command Reference

Discover CLI commands to make you more productive with Angular.

[Learn More](#)

### Schemas

Create and run schematics to generate and modify source files in your application automatically.

### Builders

Create and run builders to perform complex transformations from your source code to generated build outputs.

<https://angular.dev/tools/cli>

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Gebruiker> ng version

Angular CLI
Angular CLI: 17.0.9
Node: 18.16.0
Package Manager: npm 9.6.6
OS: win32 x64

Angular:
...

Package      Version
-----
@angular-devkit/architect    0.1700.9 (cli-only)
@angular-devkit/core         17.0.9 (cli-only)
@angular-devkit/schematics   17.0.9 (cli-only)
@schematics/angular          17.0.9 (cli-only)

PS C:\Users\Gebruiker> |
```


Current version? `ng version`

Latest version: make sure you have at least Node 20+

## New in Angular CLI v.17+

- **Modern output format** using ESM
  - dynamic import expressions to support lazy module loading.
- Faster **build-time performance**
  - both initial builds and incremental rebuilds.
- Newer **JavaScript ecosystem**
  - tools as `esbuild` and Vite.
- **Integrated SSR** and prerendering capabilities

# ESBuild – extremely fast builder

 **esbuild**

Try in the browser

Getting Started

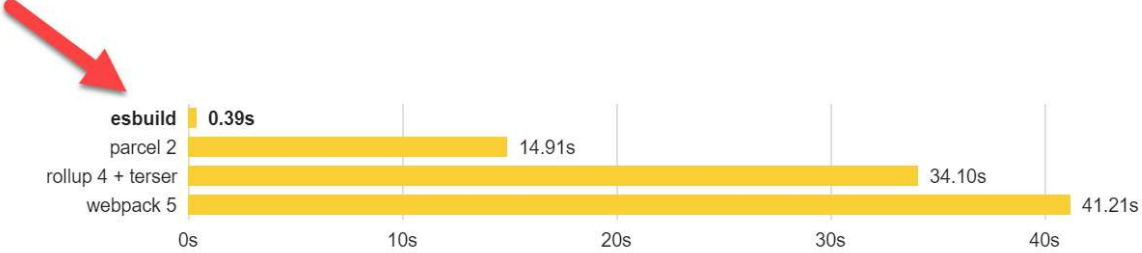
- Install esbuild
- Your first bundle
- Build scripts
- Bundling for the browser
- Bundling for node
- Simultaneous platforms
- Using Yarn Plug'n'Play
- Other ways to install

API

- Overview
- General options
- Input
- Output contents
- Output location
- Path resolution
- Transformation
- Optimization
- Source maps
- Build metadata
- Logging

# esbuild

*An extremely fast bundler for the web*



Tool	Time
esbuild	0.39s
parcel 2	14.91s
rollup 4 + terser	34.10s
webpack 5	41.21s

Above: the time to do a production bundle of 10 copies of the [three.js](#) library from scratch using default settings, including minification and source maps. More info [here](#).

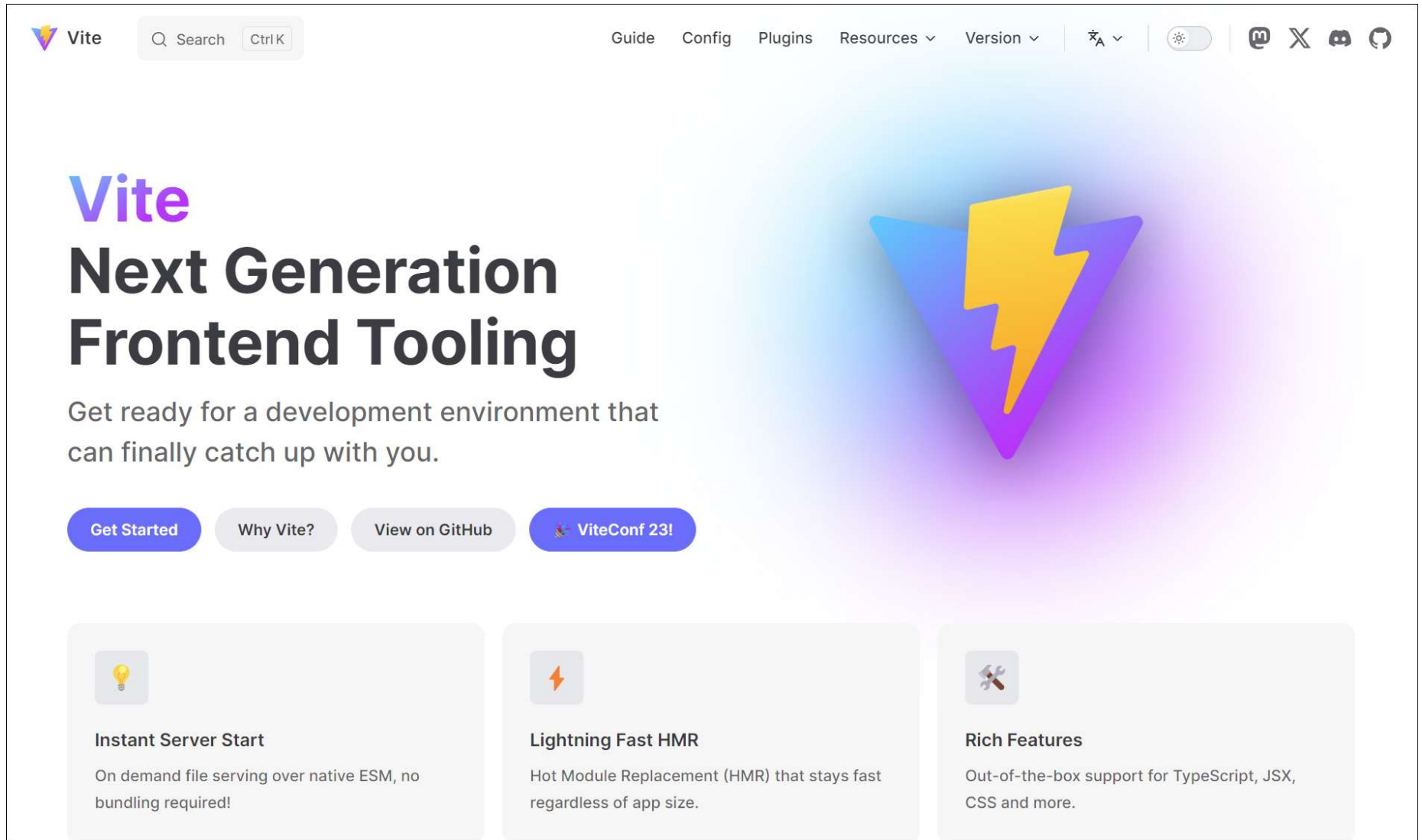
Our current build tools for the web are 10-100x slower than they could be. The main goal of the esbuild bundler project is to bring about a new era of build tool performance, and create an easy-to-use modern bundler along the way.

Major features:

- Extreme speed without needing a cache

<https://esbuild.github.io/>

# Vite.js – modern bundler



The screenshot shows the Vite.js website homepage. At the top, there is a navigation bar with the Vite logo, a search bar, and links for Guide, Config, Plugins, Resources, and Version. Below the navigation bar, the main heading reads "Vite Next Generation Frontend Tooling". To the right of the heading is a large, stylized Vite logo featuring a yellow lightning bolt inside a blue and purple triangle. Below the heading, a subheading states "Get ready for a development environment that can finally catch up with you." Underneath this, there are four buttons: "Get Started", "Why Vite?", "View on GitHub", and "ViteConf 23!". At the bottom, there are three feature cards. The first card, "Instant Server Start", features a lightbulb icon and describes on-demand file serving over native ESM. The second card, "Lightning Fast HMR", features a lightning bolt icon and describes Hot Module Replacement. The third card, "Rich Features", features a wrench icon and describes out-of-the-box support for TypeScript, JSX, CSS, and more.

Vite

Q Search CtrlK

Guide Config Plugins Resources Version

Vite

Next Generation Frontend Tooling

Get ready for a development environment that can finally catch up with you.

Get Started Why Vite? View on GitHub ViteConf 23!

**Instant Server Start**  
On demand file serving over native ESM, no bundling required!

**Lightning Fast HMR**  
Hot Module Replacement (HMR) that stays fast regardless of app size.

**Rich Features**  
Out-of-the-box support for TypeScript, JSX, CSS and more.

<https://vitejs.dev/>

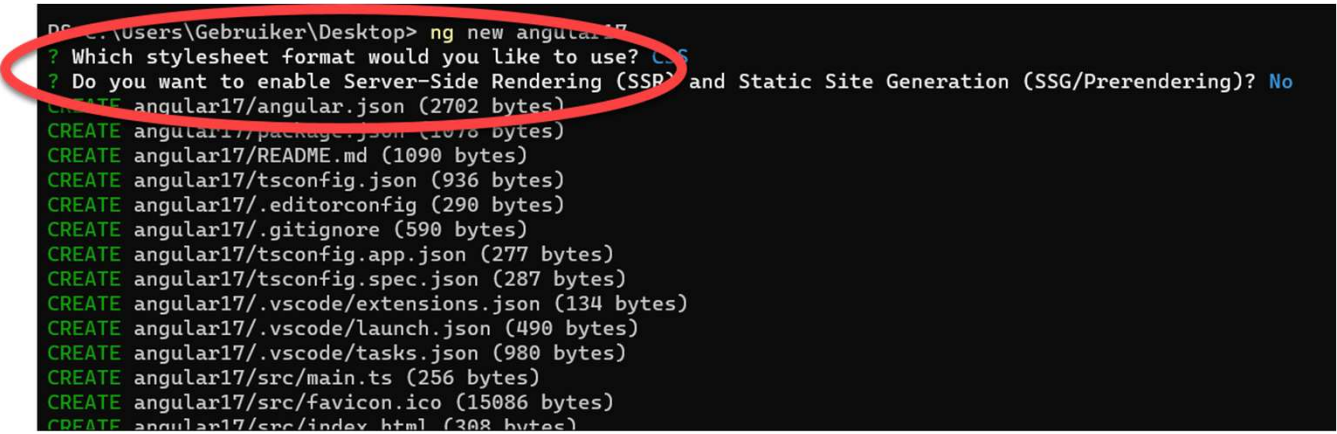


# New in CLI

- Option syntax: Unix/POSIX conventions
- Boolean options:
  - `--some-option` sets `--some-option` to `true`.  
Alternative: `--some-option=true`
  - `--no-some-option` sets `--some-option` to `false`.  
Alternative: `--some-option=false`
  - Example: `ng new --no-create-application`
- Array options:
  - Space separated or repeated
  - `--option value1 value2 ...`
  - `--option value1 --option value2 ...`

# Creating a new application – ng new

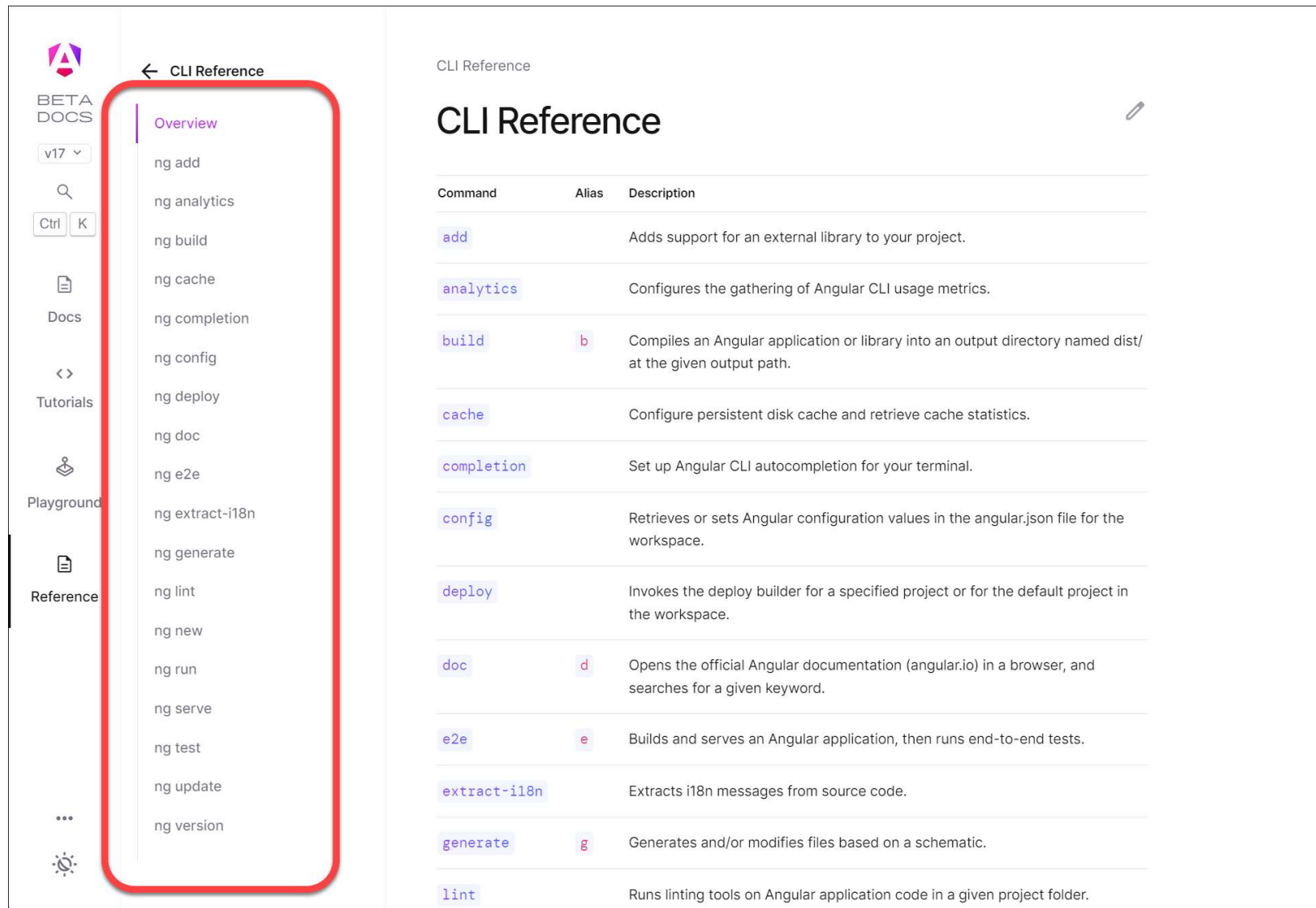
- Creating an application: like before, using `ng new application-name`
- **New options**
  - NO more Routing Y/N – now enabled by default
  - Picking a CSS variant – same
  - Enable pre-rendering SSG / SSR (default: No)
  - Picking an AI option to store defaults (`gemini.md`, `claude.md`, etc).



```
PS C:\Users\Gebruiker\Desktop> ng new angular17
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
CREATE angular17/angular.json (2702 bytes)
CREATE angular17/package.json (1070 bytes)
CREATE angular17/README.md (1090 bytes)
CREATE angular17/tsconfig.json (936 bytes)
CREATE angular17/.editorconfig (290 bytes)
CREATE angular17/.gitignore (590 bytes)
CREATE angular17/tsconfig.app.json (277 bytes)
CREATE angular17/tsconfig.spec.json (287 bytes)
CREATE angular17/.vscode/extensions.json (134 bytes)
CREATE angular17/.vscode/launch.json (490 bytes)
CREATE angular17/.vscode/tasks.json (980 bytes)
CREATE angular17/src/main.ts (256 bytes)
CREATE angular17/src/favicon.ico (15086 bytes)
CREATE angular17/src/index.html (308 bytes)
```



# CLI reference



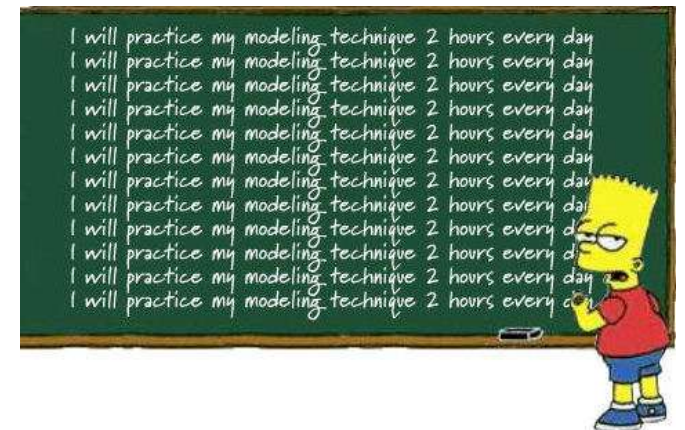
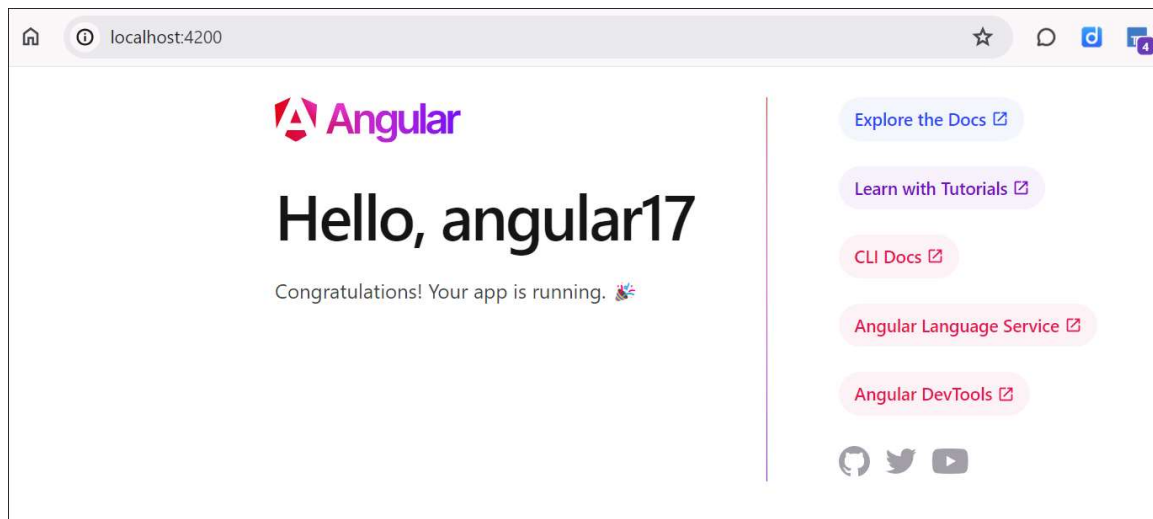
The screenshot shows the Angular CLI Reference page. On the left, a sidebar contains navigation links: BETA DOCS, v17, Search, Ctrl K, Docs, Tutorials, Playground, and Reference. The 'CLI Reference' section is highlighted with a red box. The main content area displays the 'CLI Reference' title and a table of commands.

Command	Alias	Description
<code>add</code>		Adds support for an external library to your project.
<code>analytics</code>		Configures the gathering of Angular CLI usage metrics.
<code>build</code>	<code>b</code>	Compiles an Angular application or library into an output directory named dist/ at the given output path.
<code>cache</code>		Configure persistent disk cache and retrieve cache statistics.
<code>completion</code>		Set up Angular CLI autocompletion for your terminal.
<code>config</code>		Retrieves or sets Angular configuration values in the angular.json file for the workspace.
<code>deploy</code>		Invokes the deploy builder for a specified project or for the default project in the workspace.
<code>doc</code>	<code>d</code>	Opens the official Angular documentation (angular.io) in a browser, and searches for a given keyword.
<code>e2e</code>	<code>e</code>	Builds and serves an Angular application, then runs end-to-end tests.
<code>extract-i18n</code>		Extracts i18n messages from source code.
<code>generate</code>	<code>g</code>	Generates and/or modifies files based on a schematic.
<code>lint</code>		Runs linting tools on Angular application code in a given project folder.

<https://angular.dev/cli>

# Workshop

- Install angular CLI – make sure you have the latest version
  - `ng version`
- Create a new application, using the default values
- Open the application in your editor and run it
  - `ng serve -open`
- Check components, see what has changed already





# Standalone Components

Towards an `ngModule`-less future

# Standalone components

- **Traditionally:** NgModule-based components
  - All components belong to an `@NgModule()`.
  - An `@NgModule()` acts as a container for similar functionality (Customers, Products, Login, and so on)
- **Modern applications:** standalone components
  - Components don't have to belong to a module
  - You can mix & match!
  - Better performance
  - More component-level imports
  - In preview since Angular 13+, default in Angular 17+



# What are standalone components?

Regular Angular Components with the `standalone : true` option

Angular 19+? No `standalone:true` option, but standalone is default!

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-hello',
  standalone: true,
  imports: [],
  templateUrl: './hello.component.html',
  styleUrls: ['./hello.component.css']
})
export class HelloComponent {

}
```



# Using standalone components

Simply import them in the component where you want to use it

```
import { Component } from '@angular/core';
import { HelloComponent } from "../hello/hello.component";

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [ HelloComponent ],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

```
<h1>Hello, {{ title }}</h1>
<p>Congratulations! Your app is running. 🎉</p>
<app-hello/>
```



# Hello, angular-v17

Congratulations! Your app is running. 🎉

**This is** hello component!



[Explore the Docs](#)

[Learn with Tutorials](#)

[CLI Docs](#)

[Angular Language Service](#)

[Angular DevTools](#)



# Note

- Use `import {...} from ...`
  - Otherwise it will NOT work – but **NO ERROR** will be thrown!
  - The browser just sees an unknown tag and renders nothing
  - Most IDE's handle this automatically

```
1  import { Component } from '@angular/core';  
2  import {HelloComponent} from './hello/hello.component';  
3
```

1. usages • PeterKassenaar

# Using Standard Directives


- Standard directives are made available as **standalone directives**
- We have to **import** them in the component

```
<h2 *ngIf="username">
```


```
  Username: <code>{{ username }}</code>
```

```
</h2>
```

```
import { Component } from '@angular/core';  
import { NgClass, NgFor, NgIf } from "@angular/common";
```



```
@Component({  
  selector: 'app-hello',  
  standalone: true,  
  imports: [NgIf, NgClass, NgFor],  
  templateUrl: './hello.component.html',  
  styleUrls: ['./hello.component.css']  
})
```



```
export class HelloComponent {  
  username = 'info@kassenaar.com';  
}
```

## But...new control flow syntax

- NO imports are needed when using the **new control flow syntax**, like `@if`, `@for`, and so on.
- So this will work out of the box:

```
@if (username) {  
  <h2>  
    Username: <code>{{ username }}</code>  
  </h2>  
}
```

# Standalone Pipes, Directives

- The same is true for standalone Pipes, Directives and more
  - Just add the `standalone: true` flag in Angular 19-

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'capitalize',
  standalone: true
})
export class CapitalizePipe implements PipeTransform {

  transform(value: string):string {
    return value.toUpperCase();
  }
}
```

```
@Component({
  selector: 'app-hello',
  standalone: true,
  imports: [CapitalizePipe],
  templateUrl: './hello.component.html',
  styleUrls: ['./hello.component.css']
})
```





# Why Standalone components?

- Removing the need for NgModules – **no extra complexity**
  - Handy for beginners
- But: “I have to **import** everything in that component. Really?”
  - Yes – but IDE’s handle that automatically for you
- Now, **components** can be **lazy loaded** instead of complete modules
  - More modularization, less monolithic
  - Performance!


*The main benefit of standalone components is that they make it trivial to develop a fully lazy-loaded application, or migrate an existing application and make it fully lazy-loaded.*

# Mixing and matching

- No need for a 'big bang' when refactoring
- Using **standalone components in NgModule** based applications
- Standalone components act as other modules, so `import` them:

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import {HelloComponent} from "../hello/hello.component";


@NgModule({
  declarations: [classicComponent],
  imports: [
    CommonModule,
    HelloComponent
  ]
})
export class CustomerModule { }
```




# Using NgModules in standalone components

- Export the component from the module as usual
- Import the module in the component as (now) usual

```
@NgModule({  
  declarations: [  
    CustomerComponent  
  ],  
  imports: [  
    CommonModule,  
  ],  
  exports: [  
    CustomerComponent  
  ]  
})  
export class CustomerModule {
```



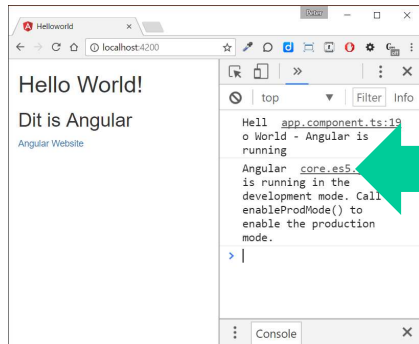
```
@Component({  
  selector: 'app-root',  
  standalone: true,  
  imports: [HelloComponent, CustomerModule],  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```



# Combined NgModules/Standalone



# Classic Angular apps



**main.ts / bootstrapper**

**ngModule / root module**

**AppComponent**

**Services**

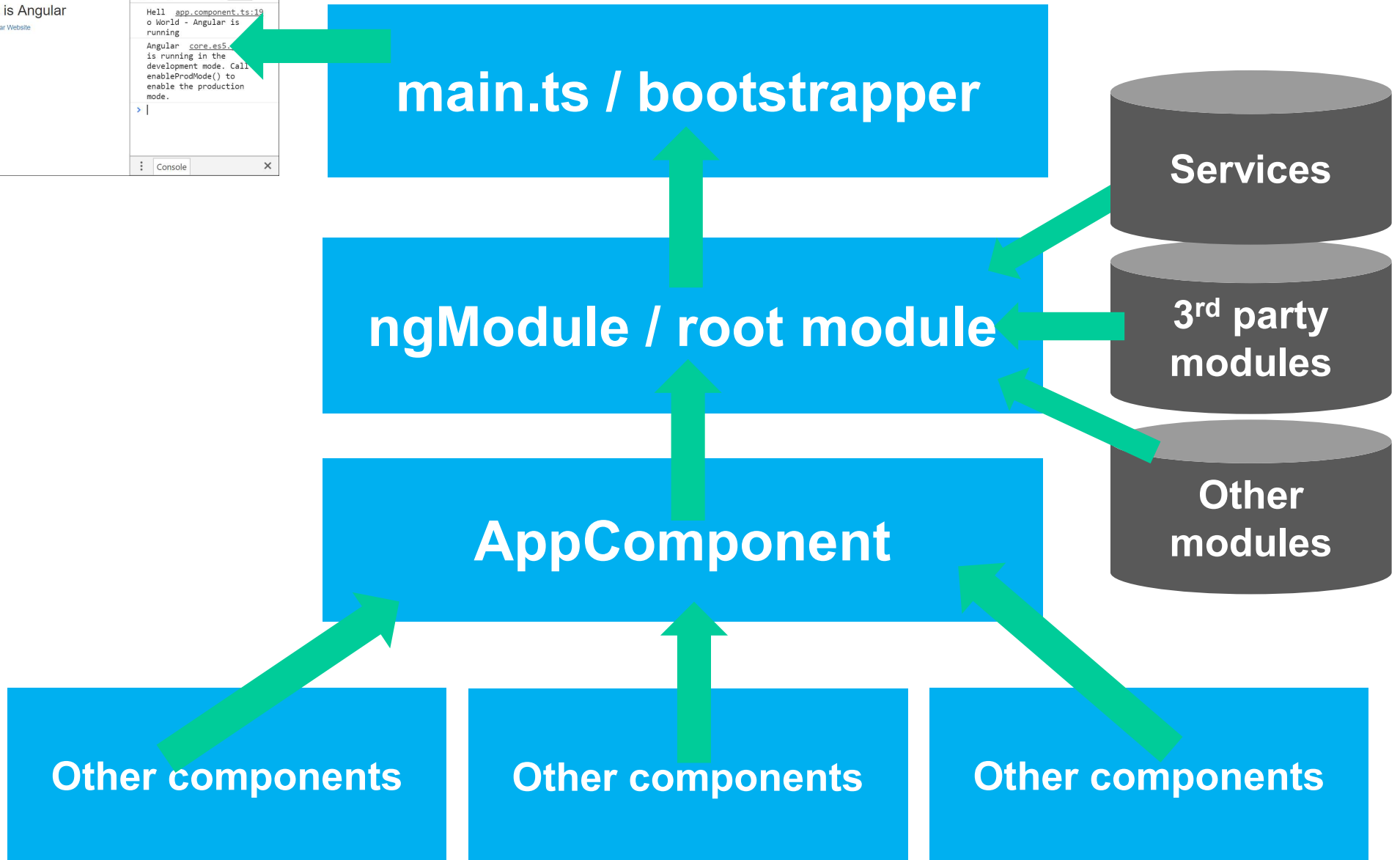
**3<sup>rd</sup> party  
modules**

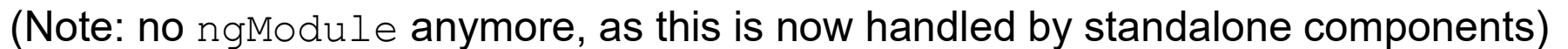
**Other  
modules**

**Other components**

**Other components**

**Other components**



[illegible]

(Note: no `ngModule` anymore, as this is now handled by standalone components)

# Workshop

- Open repo `ngx-new-features` and run it (`npm install, npm start`)
- Study the standalone components **structure and architecture**
- Going old school: create a new module
- Create a new component inside this new module and give it some UI.
- Include the module in the Main Standalone Component and show it besides other components
- Then the other way: Include the 'old' module in the component and show it's contents
  - You can mix and match!
  - No big bang needed

