



# Angular Fundamentals

## Korte herhaling dag 1

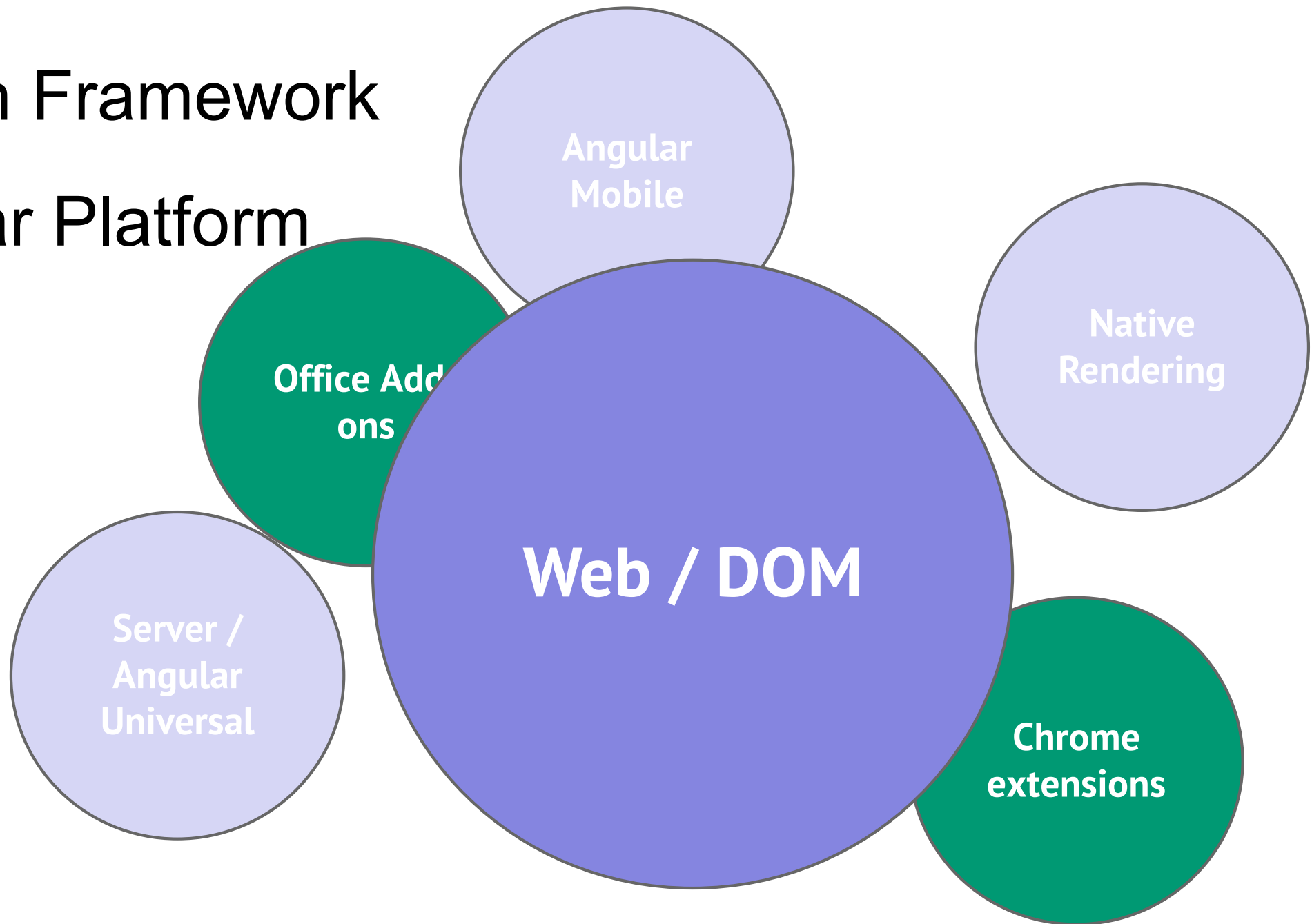


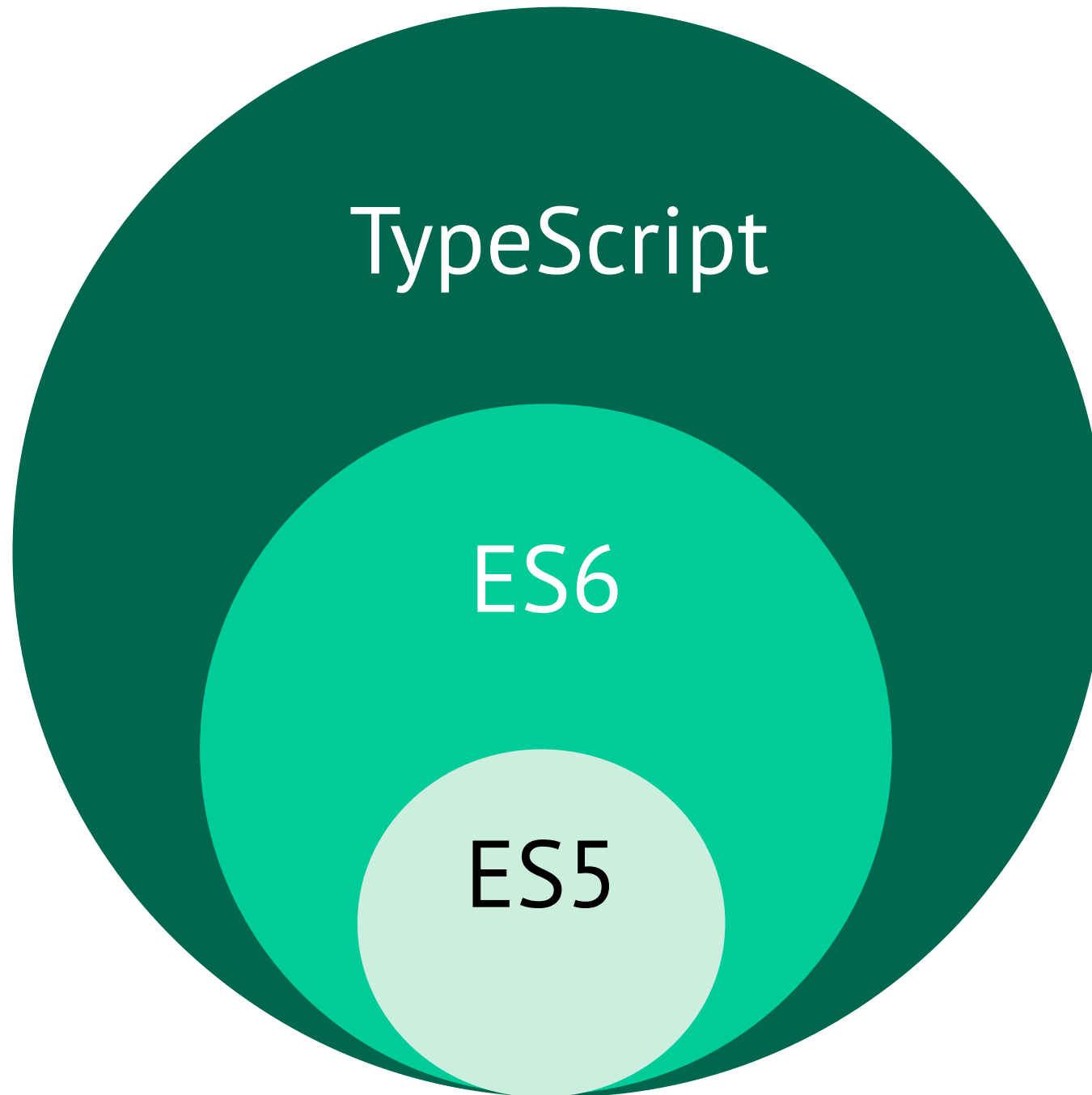
Peter Kassenaar –  
[info@kassenaar.com](mailto:info@kassenaar.com)

# Framework to Platform

|           | Scaffolding                             | Code completion &<br>Refactoring | Debugging |
|-----------|---|----------------------------------|-----------|
| Tooling   | Angular CLI                             | Language<br>Services             | Augury    |
| Libraries | Material 2                              | Mobile                           | Universal |
|           | Compile                                 | Change<br>Detection              | Renderer  |
| Core      | Components &<br>Dependency<br>Injection | Decorators                       | Zones     |

# Van Framework naar Platform





TypeScript

ES6

ES5

# Boilerplate code - stappenplan

1. Set up environment, boilerplate & libraries

– `npm install`

2. Schrijf Angular Root Component & Module voor de app

3. Bootstrap

4. Schrijf HTML-pagina (`index.html`)

5. Run applicatie

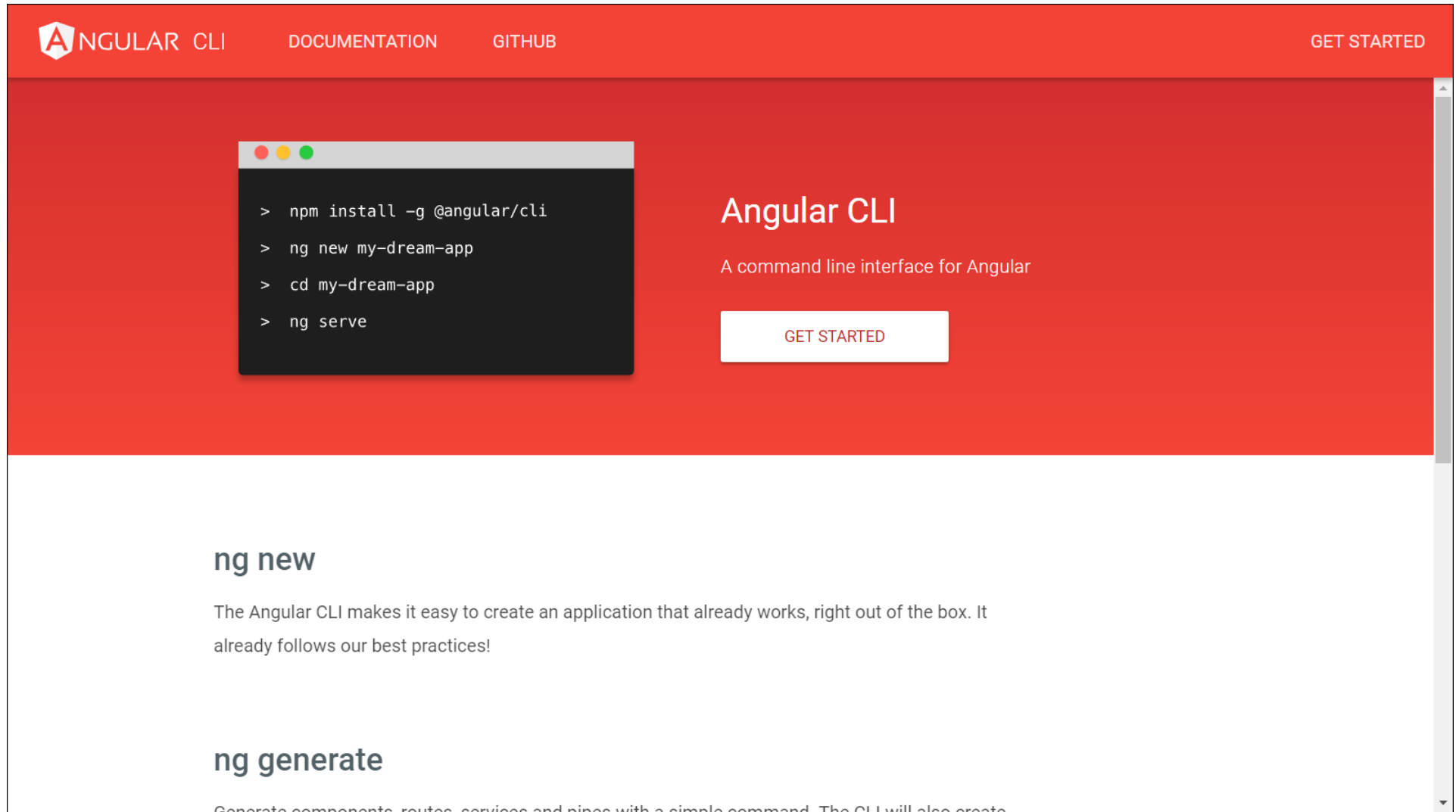
– `npm start`

# Components

- @Component decorator
- View
- Controller / Class

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4      moduleId: module.id,
5      selector: 'some-component',
6      templateUrl: 'some.component.html'
7  })
8  export class SomeComponent implements OnInit {
9      constructor() { }
10
11     ngOnInit() { }
12
13     // ... more logic
14 }
```

# Angular CLI



# Ng serve

- Vervult verschillende rollen (*kan* ook handmatig)
  1. TypeScript compilation
  2. (SASS compilation)
  3. Fire up webserver on <http://localhost:4200>
  4. Fire up Live Reload
  5. Analyzing and bundling of application by firing up WebPack
  6. Watch for changes and recompile, rebundle, etc.



# Data binding

- Nieuwe notatiewijzen in HTML-views/partials.
  1. Simple data binding `{{ ... }}`
  2. Event binding `(...) = "..."`
  3. One-way data binding `[...] = "..."`
  4. Two-way data binding `[(ngModel)] = "..."`

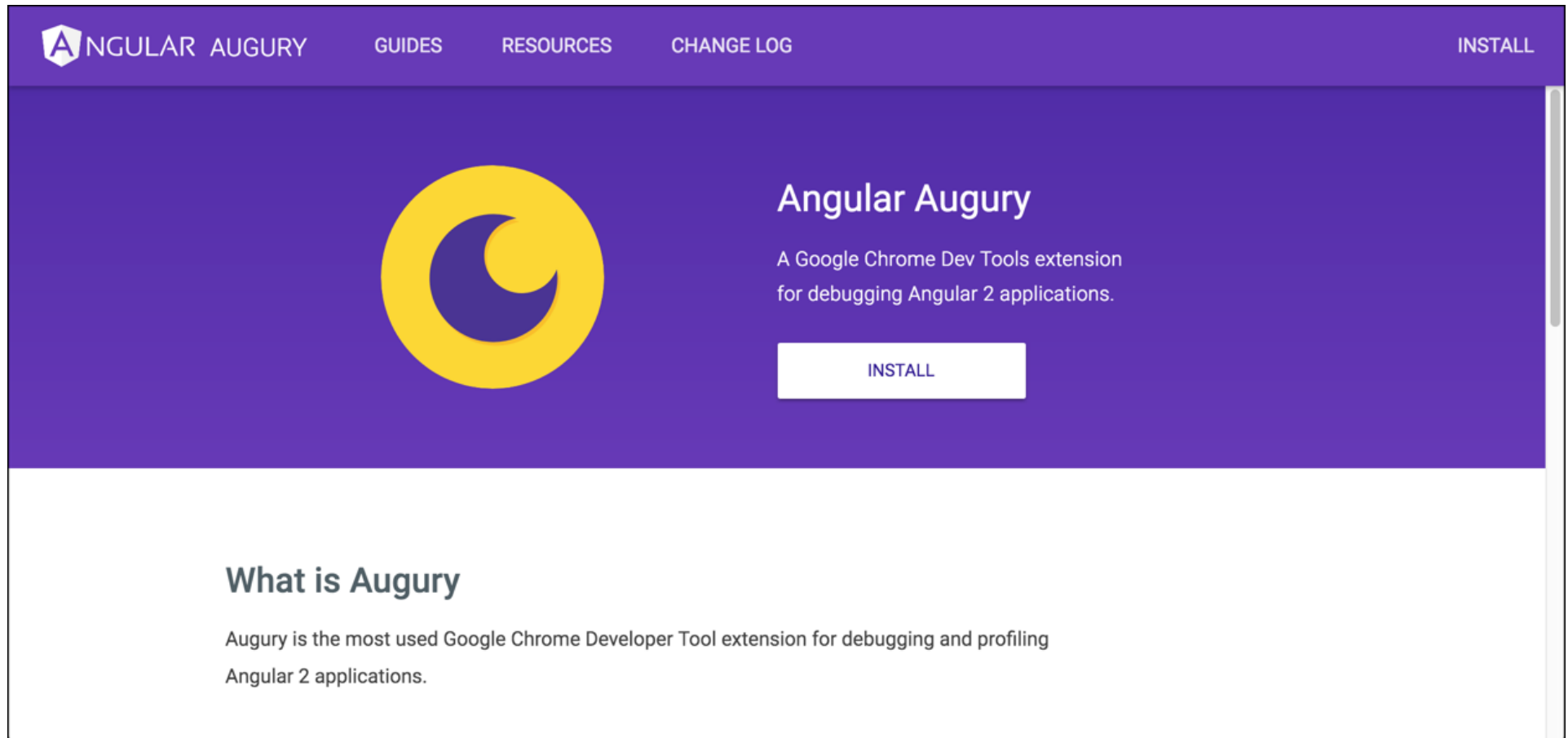
# Event binding

- Niet-DOM events binden: `@HostListener()`
- Event van View → Controller?
  - `$event` doorgeven als parameter
- Local Template Variable, #
- `@ViewChild('varName')` voor doorgeven DOM-element aan controller/class

# Vandaag

- Afronden Attribute binding
- two-way Data Binding
- Services
  - Static Services
  - RxJS / Observables, Live API's
- Tree of components
  - Applicaties opbouwen met meerdere componenten
  - Data flow tussen de componenten
  - `@Input()` en `@Output()`
- Case

# Debugging/Analyse tool - Augury



<https://augury.angular.io/>

# Angular Style Guide

The screenshot shows the Angular website's navigation bar with links for FEATURES, DOCS, RESOURCES, EVENTS, and BLOG. A search bar is located on the right. The left sidebar contains a list of categories: GETTING STARTED, TUTORIAL, FUNDAMENTALS, TECHNIQUES (which is expanded to show Internationalization, Security, Setup & Deployment, Upgrading, Visual Studio 2015 QuickStart, Style Guide, and Glossary), and API. The main content area is titled 'Style Guide' and contains an introductory paragraph, a 'Style vocabulary' section with guidelines on 'Do', 'Avoid', and 'Why?', and a 'File structure conventions' section. The right sidebar provides a detailed table of contents for the 'Style Guide' section, listing items such as 'Style vocabulary', 'File structure conventions', 'Single responsibility', 'Rule of One', 'Small functions', 'Naming' (with sub-items like 'General Naming Guidelines'), 'Directive selectors', and 'Pipe names'.

ANGULAR FEATURES DOCS RESOURCES EVENTS BLOG Search

GETTING STARTED  
TUTORIAL >  
FUNDAMENTALS >  
TECHNIQUES >  
Internationalization (i18n)  
Security  
Setup & Deployment >  
Upgrading >  
Visual Studio 2015 QuickStart  
Style Guide  
Glossary  
API

stable (v4.4.6)

## Style Guide

Looking for an opinionated guide to Angular syntax, conventions, and application structure? Step right in! This style guide presents preferred conventions and, as importantly, explains why.

### Style vocabulary

Each guideline describes either a good or bad practice, and all have a consistent presentation.

The wording of each guideline indicates how strong the recommendation is.

**Do** is one that should always be followed. *Always* might be a bit too strong of a word. Guidelines that literally should always be followed are extremely rare. On the other hand, you need a really unusual case for breaking a *Do* guideline.

**Consider** guidelines should generally be followed. If you fully understand the meaning behind the guideline and have a good reason to deviate, then do so. Please strive to be consistent.

**Avoid** indicates something you should almost never do. Code examples to *avoid* have an unmistakable red header.

**Why?** gives reasons for following the previous recommendations.

### File structure conventions

Some code examples display a file that has one or more similarly named companion files. For example,

- Style Guide
  - Style vocabulary
  - File structure conventions
  - Single responsibility
    - Rule of One
    - Small functions
  - Naming
    - General Naming Guidelines
    - Separate file names with dots and dashes
    - Symbols and file names
    - Service names
    - Bootstrapping
    - Directive selectors
    - Custom prefix for components
    - Custom prefix for directives
    - Pipe names
    - Unit test file names
    - End-to-End (E2E) test file names
    - Angular *NgModule* names

<https://angular.io/guide/styleguide>

## Dag 3

- Bespreking Case
- Multiple components
  - Event bus / Service bus
- POST-ing data / Live CRUD-operations
- Routing
- Forms
- Examen