



Angular – The Complete Guide



A Web Framework

For Building Highly Interactive Web Applications

Basics First

No Prior Angular Knowledge Required!

Course Requirements

Basic JavaScript & Web Development Knowledge

Stable Evolution

As Angular changes, this course changes with it

But in a stable, incremental, non-breaking way

Or dive right into the sections that are most interesting to you!

Deep Dives

Components
Directives
Pipes
Services & DI
Change Detection
HTTP
Forms
Routing
...

Start with no Angular knowledge

Essentials

And become an Angular developer!



Welcome To The Course

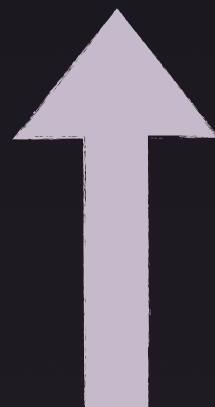
Maximilian Schwarzmüller

@maxedapps

maximilian-schwarzmueller.com

academind.com

Advanced Concepts



Angular Essentials

No prior Angular experience required!

Basic web development & JavaScript
experience is all you need



What Is Angular?

And why would you use it?

Maximilian Schwarzmüller – “Angular - The Complete Guide”



What Is Angular?



Angular is a **frontend JavaScript framework**

It helps with building interactive, modern web user interfaces



It's also a **collection of tools & features**

CLI, Debugging Tools, IDE Plugins



Why would you use Angular?

Maximilian Schwarzmüller – “Angular - The Complete Guide”

Why a framework?

Maximilian Schwarzmüller – “Angular - The Complete Guide”

You don't need it!

For trivial websites and web apps!

But that changes as your applications get more complex!

Why a framework?

Simplifies the process of building complex, interactive web user interfaces

Write **declarative code**

Separation of concerns via **components**

Object-oriented programming concepts & principles

Use **TypeScript**

With “Just JavaScript”

You write **imperative** code!

Step-by-step instructions (in code) that tell the browser what to do

Example

- 1) Find DOM element & store in variable
 - 2) Add event listener to element
 - 3) In triggered function: Find another element
 - 4) Set CSS visibility of that element to 0
 - 5) Create a new `<p>` DOM element
 - 6) Set text content of `<p>` to “Hello World!”
 - 7) Find element into which the `<p>` should be inserted
 - 8) Insert `<p>` element
- And so on ...

With Angular

You write **declarative** code!

Define the target UI states & how they change and let Angular do the rest

Example

- 1) Manage “activeTab” state property
- 2) Depending on “activeTab”, show different content
- 3) Change “activeTab” upon click on tab element

Why Components?

Modular Applications

Break up complex applications into simple building blocks

Split up responsibilities & concerns

Build a component once and re-use it as often as needed

Better Development Process

Assign different team members to different components

Share components & logic across the team

Reduce dependencies

Using TypeScript



TypeScript is a JavaScript superset – an extension of JavaScript

It extends the JavaScript syntax to support strong & strict typing

Unlike “vanilla JavaScript”, TypeScript enforces types and
prevents unexpected value type changes

TypeScript is **not an Angular feature** but can be used
independently – Angular embraces it though, hence all Angular
projects use TypeScript

Easier Development With TypeScript



With TypeScript, you can often catch errors early on during development

For this course, you **don't need to know TypeScript!**
You'll **learn it along the way** and there also is an **optional “TypeScript Crash Course” section** at the end of the course – you can always go through that section if you're struggling to follow along with TypeScript

Why Angular?

Highly scalable, well maintained, innovative & batteries included

Developed & used by **Google**

Improvements, fixes & features are added continuously

The framework keeps **evolving**

Backward-compatibility & stability is a number 1 priority

It's a all-in-one package – lots of **core features are built-in**



Angular: A Stable Yet Evolving Framework

Angular has backward-compatibility as a number 1 priority
At the same time, the Angular keeps innovating & aims to
improve and advance the framework over time

Angular: A Stable Yet Evolving Framework

2016

Angular 2 is released

2017

Angular 4 is released (v3 is skipped)

2017

Angular 5 is released

...

Angular 6 - 13 are released

2022

Angular 14 & 15 are released



The Angular team has a versioning & release policy where a **new major version** is released **every ~6 months**

BUT: Despite labeled as major versions, these releases do NOT break your code & “change the framework”

2023

Angular 16 is released



Angular 14 introduces “Standalone Components”

Angular 16 introduces “Signals”

Angular keeps innovating & changing - with **backward compatibility** as a top priority



Different Projects, Different Versions

Maximilian Schwarzmüller – “Angular - The Complete Guide”



Creating an Angular Project

Maximilian Schwarzmüller – “Angular - The Complete Guide”

About This Course

Deep Dives

Components
Directives
Pipes
Services & DI
Change Detection
HTTP
Forms
Routing
...

Essentials

Maximilian Schwarzmüller – “Angular - The Complete Guide”



Angular Essentials

Understanding Angular Project Structure

Working with Components

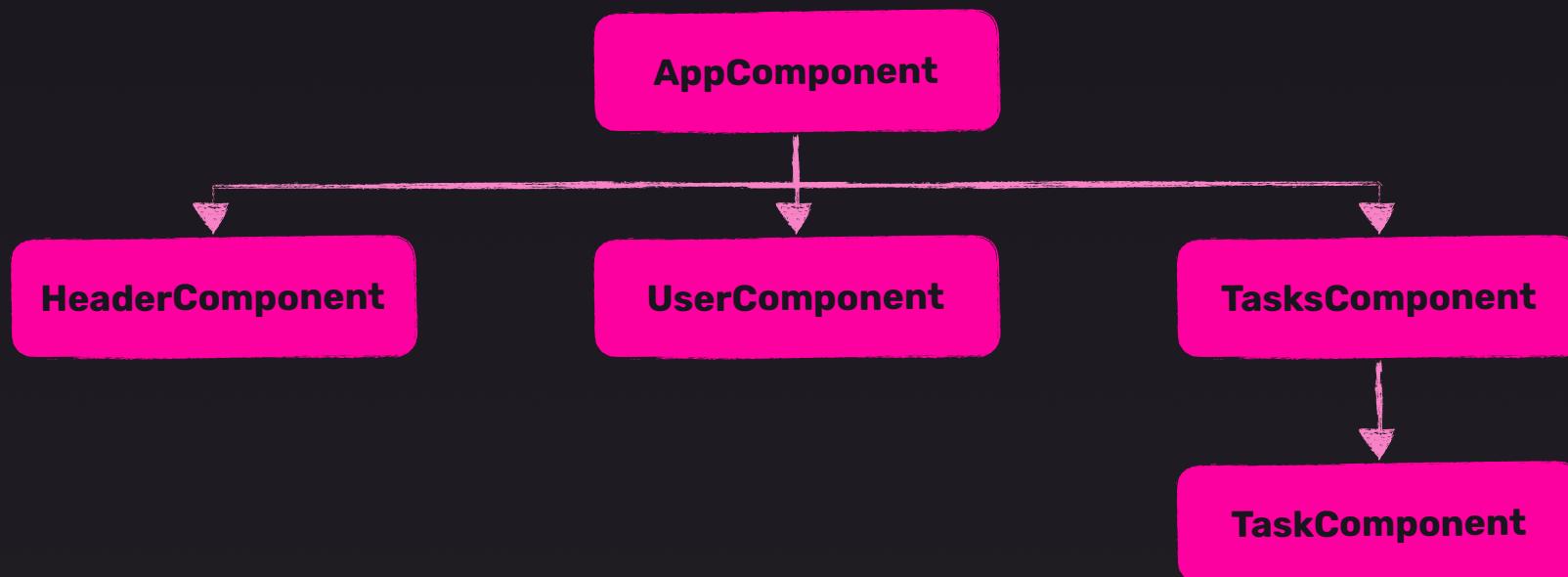
Handling User Events

Rendering & Updating Dynamic UI Content

Maximilian Schwarzmüller – “Angular - The Complete Guide”

You Build A Component Tree

One Angular Application = One Component Tree



There Are Two Approaches For Updating State



Option 1

Relying on Zone.js & Angular's change detection mechanism

Works automatically, no special instructions required

Supported since Angular 2



Option 2

Using Signals to notify Angular about value changes & required UI updates

Requires usage of special "signal" instructions & code

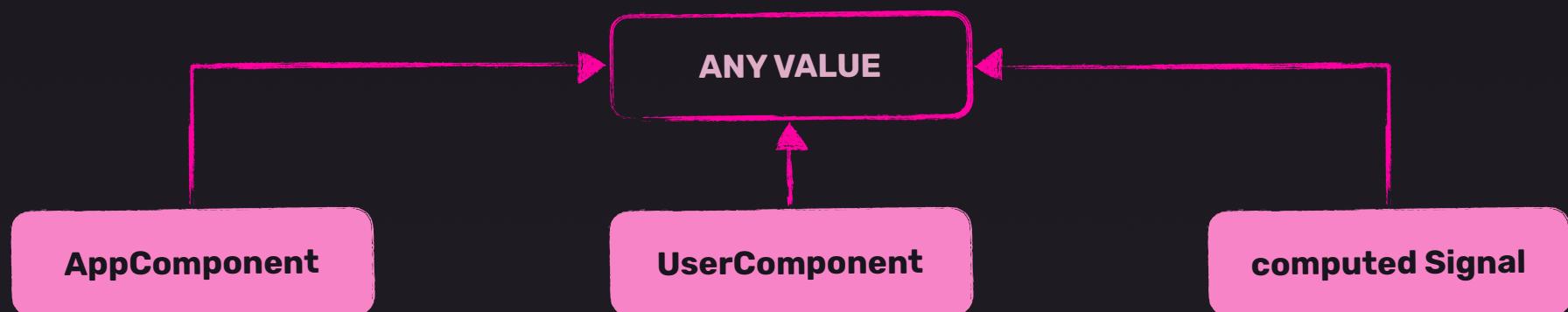
Supported since Angular 16

Signals Are Trackable Data Containers



Signal

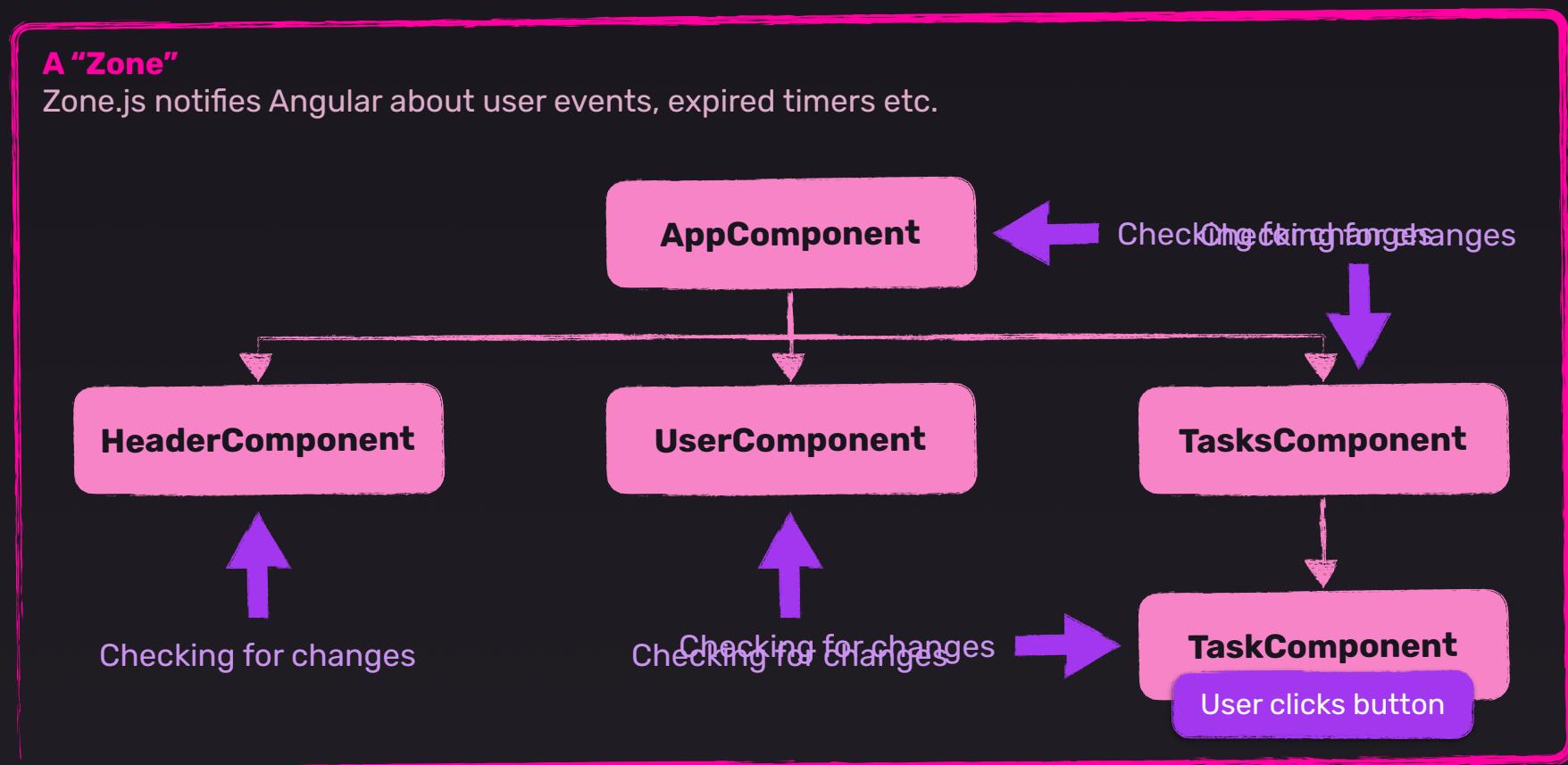
A signal is an object that stores a value (any type of value, including nested objects)



Angular manages subscriptions to the signal to get notified about value changes

When a change occurs, Angular is then able to update the part of the UI that needs updating

Angular's Change Detection Mechanism



Directives

With Angular, you can “enhance” elements by adding so-called **Directives** to them

<input ngModel>



NgModel Directive

An “element enhancement” that helps with extracting (or changing) user input values

Directives, unlike components, **don't have a template!**

Components



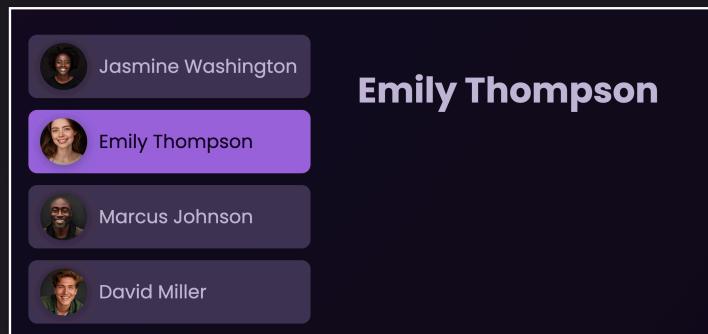
Directives

Components are directives! Directives with templates

Exercise

Create and use a “Tasks” component (app-tasks)

Receive & output the name of the selected user



HINT

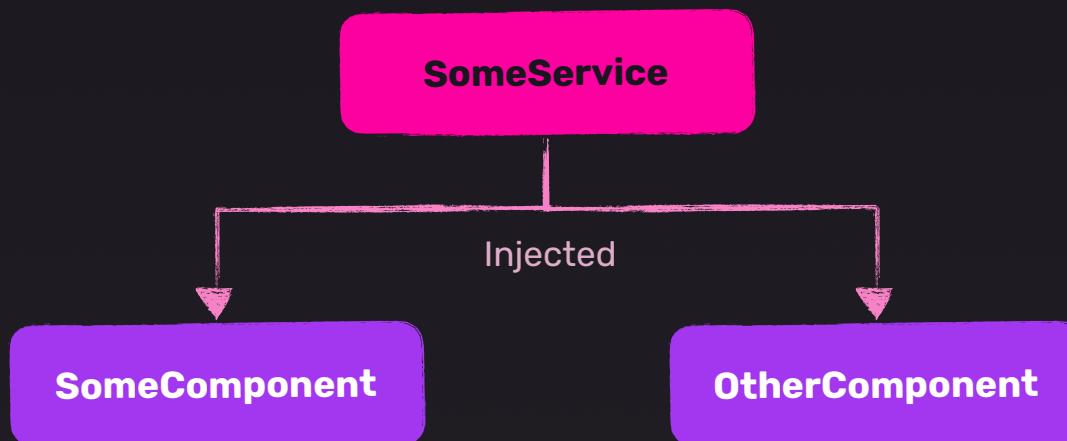
You can find a user with a specific id in the DUMMY_USERS array like this:

```
DUMMY_USERS.find(user => user.id === selectedUserId)!
```

Services



For centralized & sharable data- and logic-management



Two-Way-Binding



Change data AND listen to data changes at the same time

Example:
`<input />`

Bind entered value to
property

Bind property to entered
value



Angular Modules

Understanding Angular Modules (NgModule)

Using Modules Instead Of Standalone Components

Declaring & Exporting Components

Shared Modules

Angular Modules Exist For Historic Reasons

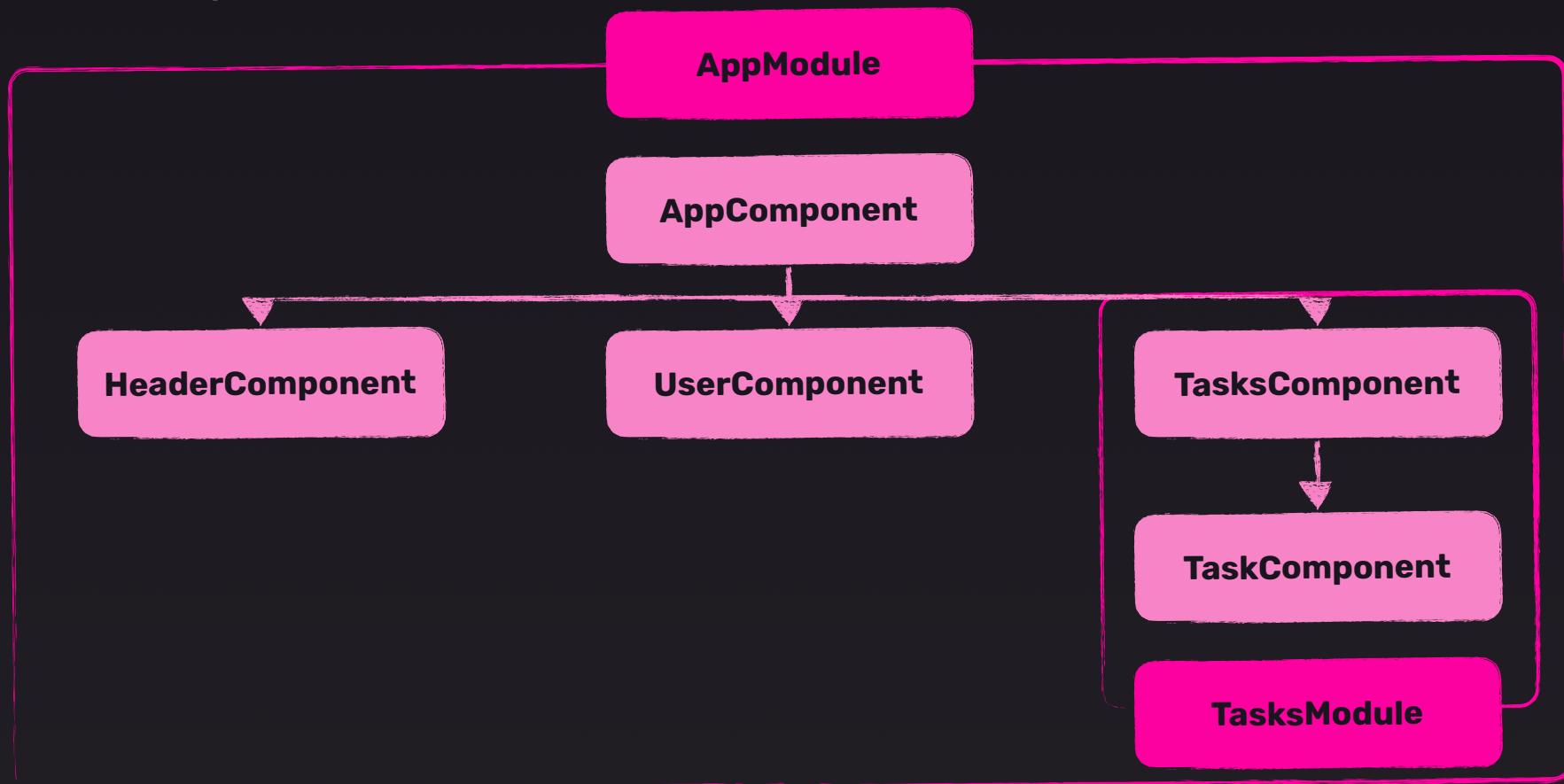
When Angular 2 was released in 2016, there were no “Standalone Components”

Today, Standalone Components are the recommended way of building components

But you can also still use “Module-based Components” with Angular Modules

Understanding Angular Modules

Angular Modules “make components (& more) available to each other”





Angular Debugging

Understanding & Fixing Error Messages

Debugging Logical Errors

Using the Angular Developer Tools



Angular Essentials - Practice Project

Apply Your Knowledge & Practice What You Learned

Maximilian Schwarzmüller – “Angular - The Complete Guide”

Exercise

Build an “Investment Calculator” Application

1

Add a “Header” component with a title & image

2

Add a “User Input” component that collects user input with two-way-binding

3

Add an “InvestmentResults” component that outputs results in a table

You can use signals or not

You can use standalone components or module-based components