# 01. Getting Started

## Welcome to the course!

---

# Angular - The Complete Guide: Introduction

### Welcome Message

Welcome to this Angular course!

It's great to have you on board.

My name is **Maximilian Schwarzmüller**. I am a professional Angular developer and an online course instructor. Through this course, I have taught hundreds of thousands of students how to use Angular effectively in their day-to-day work.

My goal is to help you understand how Angular works, how to use it in real-world projects, and how to become a proficient Angular developer—regardless of your current experience level.

---

### Who Is This Course For?

This course is designed for anyone interested in learning Angular:

- **No prior Angular experience is required.**
- A basic understanding of **web development** and **JavaScript** is sufficient.

We'll start from the **very basics** and gradually progress to more **advanced concepts**, ensuring that by the end of the course:

- You'll be confident using Angular in your own projects.
- You'll be able to **call yourself an Angular developer**.

---

### What Will You Learn?

Throughout the course, we will explore:

- The **core fundamentals** of Angular
- Its **essential features**
- More **advanced concepts** needed to build robust applications

You will gain both theoretical knowledge and practical experience to work on professional Angular projects.

---

### Let's Get Started

Before we begin writing code or creating Angular projects, let's address the most important question:

### What exactly is Angular, and why would you use it?

That's what we'll cover next as we kick off this journey into modern front-end development with Angular.

---

## What is Exactly is Angular?

---

# Lesson 2: What Is Angular and Why Use It?

### What Is Angular?

Angular is actually **two things**:

1. **A Front-End JavaScript Framework**
   Most importantly, Angular is a **front-end JavaScript framework**.
   This means it is a **collection of packages and conventions** that guide you in building advanced, feature-rich, and highly interactive **web user interfaces (UIs)**.

   Angular simplifies the process of building such UIs, making it **faster, more efficient, and more maintainable** than using plain JavaScript without any framework.

2. **A Complete Development Platform**
   In addition to being a framework, Angular is also a **comprehensive toolset** for web development. This includes:
   - A **Command Line Interface (CLI)** for creating Angular projects and spinning up local development servers
   - **Debugging tools** and **utilities** for improved development and testing
   - **IDE and code editor plugins** that enhance productivity with smart code completion and error detection

---

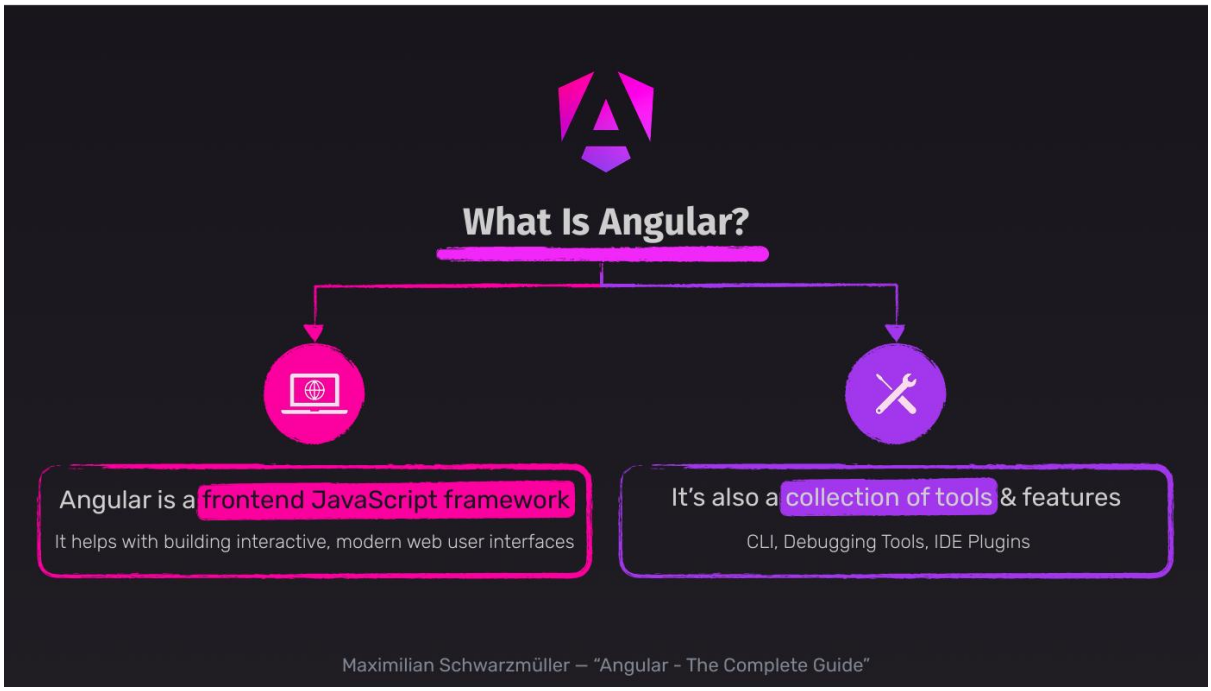
### Why Use Angular?

With its combination of a powerful framework and integrated tooling, Angular offers a **platform** for developing anything from:

- Simple, trivial applications
- To complex, enterprise-grade **web applications**

Angular is ideal when your goal is to build **scalable**, **modular**, and **maintainable** front-end applications—especially in professional and team-based environments.

---




# Why would you use Angular?

---

# Lesson 3: Why Use Angular Instead of Just JavaScript?

Now that you know what Angular is, a valid follow-up question would be:

### Why use Angular?

Why choose a framework like Angular instead of simply using **Vanilla JavaScript**, especially for basic or small-scale web applications?

---

### When You Might Not Need Angular

For **trivial web applications or simple user interfaces**, Angular—or any framework—might be unnecessary. Plain JavaScript could suffice.

---

## When Angular Shines

However, as your application becomes **more complex and interactive**, frameworks like Angular start to truly **shine**. They simplify the development process significantly through built-in structure, patterns, and tools.

Angular provides many powerful features, conventions, and architectural patterns that help you:

- Build large-scale applications efficiently
- Keep code **maintainable** and **modular**
- Collaborate effectively in teams

---

## The Four Core Benefits of Angular

### 1. Declarative Programming

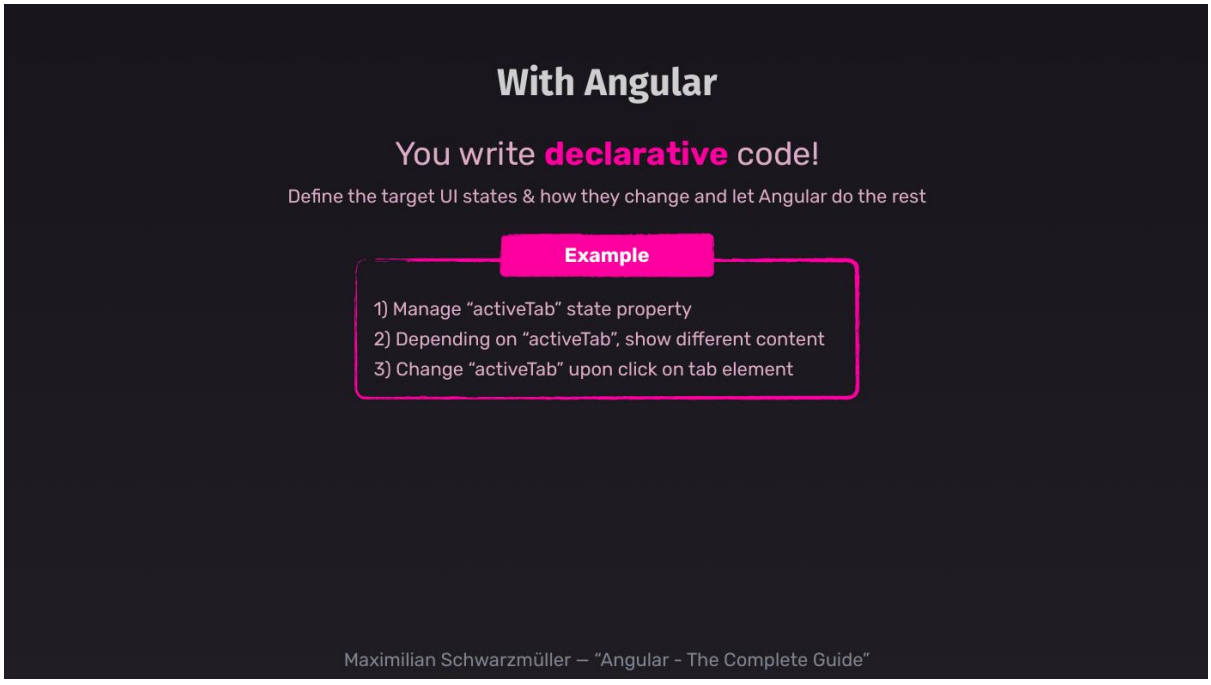
With Angular, you write **declarative code** instead of imperative code (as in Vanilla JavaScript).
This means you **describe the desired result**, and Angular figures out how to achieve it.

In Vanilla JavaScript, you often write code like this:

- Locate DOM elements manually
- Perform checks
- Conditionally update the DOM, step by step


In Angular:

- You define target states directly in the **HTML markup** using special Angular syntax
- Angular automatically takes care of **DOM updates** based on changes in application state




This shift to declarative programming reduces manual work and improves readability and maintainability.

## 2. Component-Based Architecture

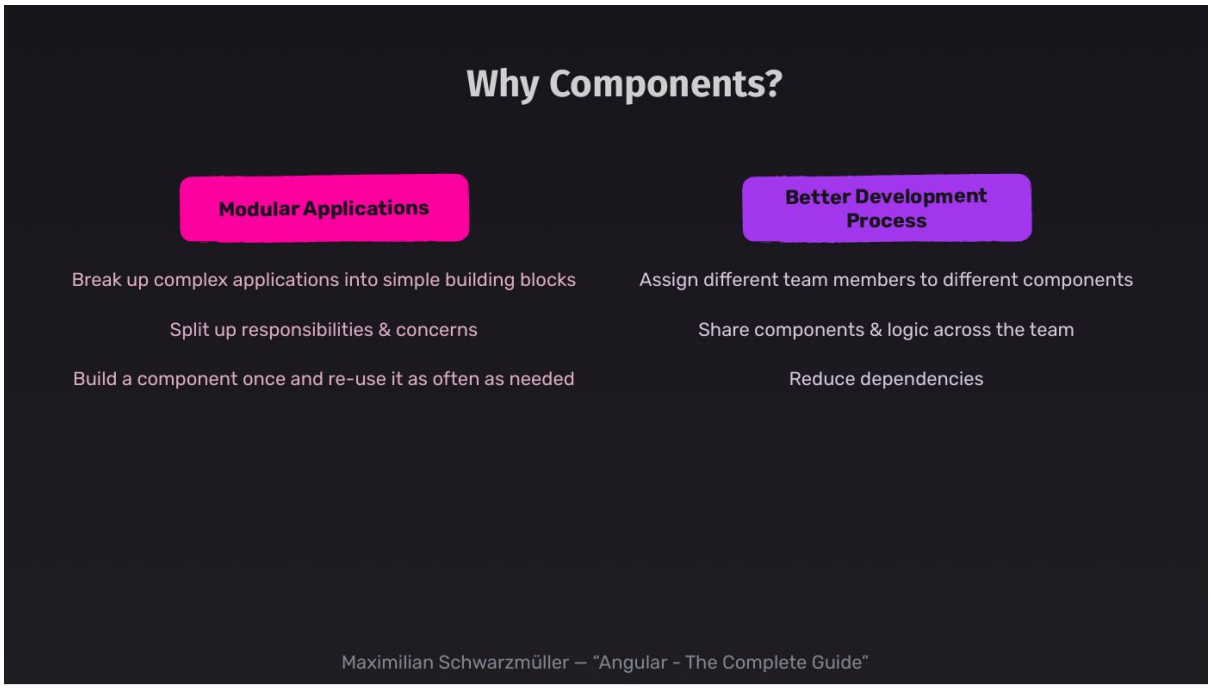
Angular encourages you to build your UI using **components**.

A **component** in Angular is:

- A custom, reusable building block
- Made up of HTML (markup), CSS (styling), and TypeScript (logic)

Benefits of components:

- Simplify complex UIs by **breaking them into manageable parts**
- Promote **code reusability**
- Make team collaboration easier by allowing different developers to work on different components independently




---

## 3. Object-Oriented Concepts (Like Dependency Injection)

Angular embraces **object-oriented programming principles**, especially:

- **Classes** to define components and services
- **Dependency Injection (DI)** to manage how different parts of your application are wired together

Don't worry if you're not familiar with OOP or DI—you'll learn these concepts throughout the course.

These patterns help make your application:

- Easier to **scale**
- More **testable**
- Better **structured**

---

### 4. TypeScript Support

Angular uses **TypeScript**, not plain JavaScript.

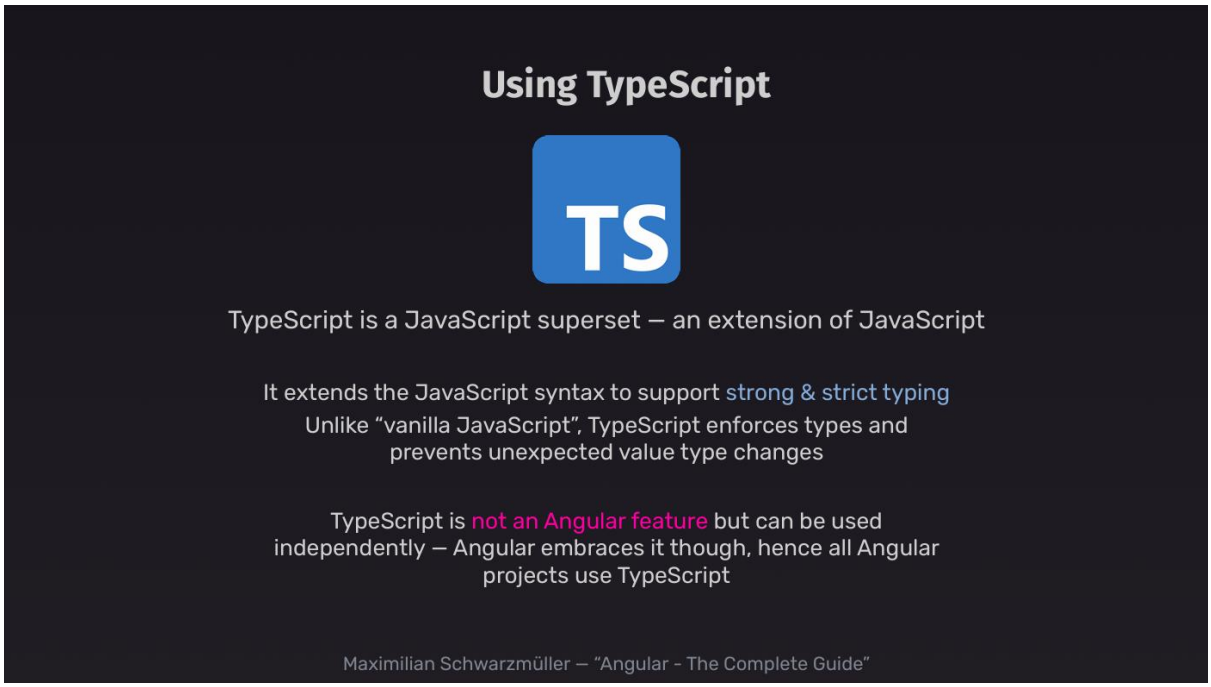
What is TypeScript?

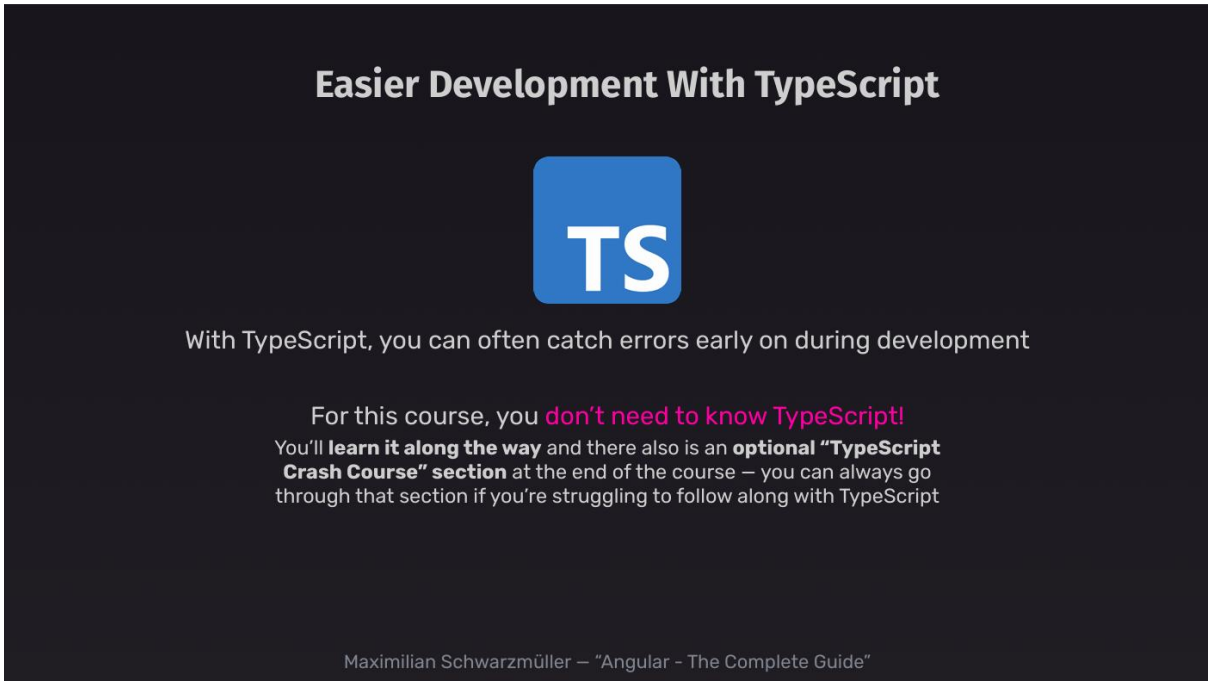
- A superset of JavaScript that adds **static typing**
- Allows early detection of bugs
- Helps you catch errors **during development**, rather than at runtime

Examples of TypeScript benefits:

- Warns when passing the wrong value type to a function
- Helps avoid using undefined variables
- Enhances developer tooling with **auto-complete**, **error hints**, and **refactoring support**

Good news: **You don’t need prior TypeScript experience**. You'll learn TypeScript naturally as you progress through the course.






---

### Summary

Angular provides:

- A **declarative** and structured way to write UI code
- A **component-based** system for organizing and scaling apps
- **Modern programming patterns** like DI
- The **safety and tooling benefits** of TypeScript

Together, these features make Angular an excellent choice for building **scalable**, **maintainable**, and **robust** web applications.

---

## Angular's Evolution & Stability

---

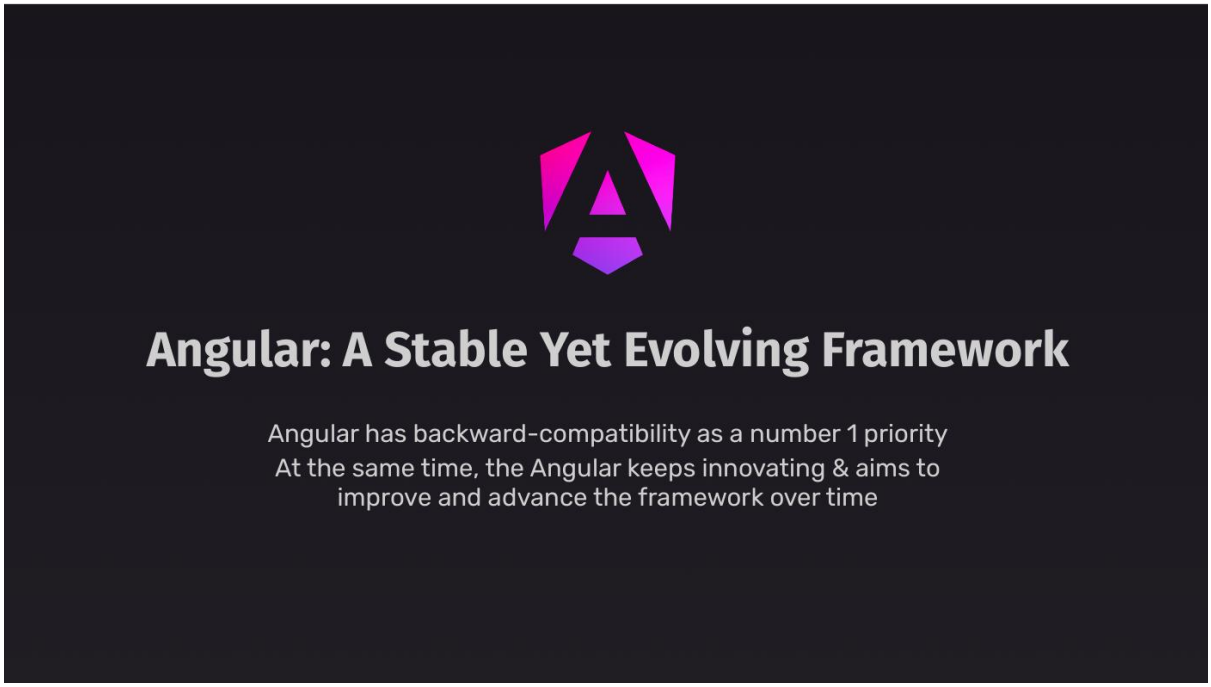
# Lesson 4: Angular's Evolution & Versioning

### Angular Keeps Evolving—But Carefully

Before we dive into writing Angular code, there's something important you should understand about Angular as a **framework**:

Angular continues to evolve and innovate—**but in a stable and backward-compatible way**.

This is important because, as a developer, you don't want to relearn Angular every year. Fortunately, you don't have to.




---

## A Brief History of Angular Versions

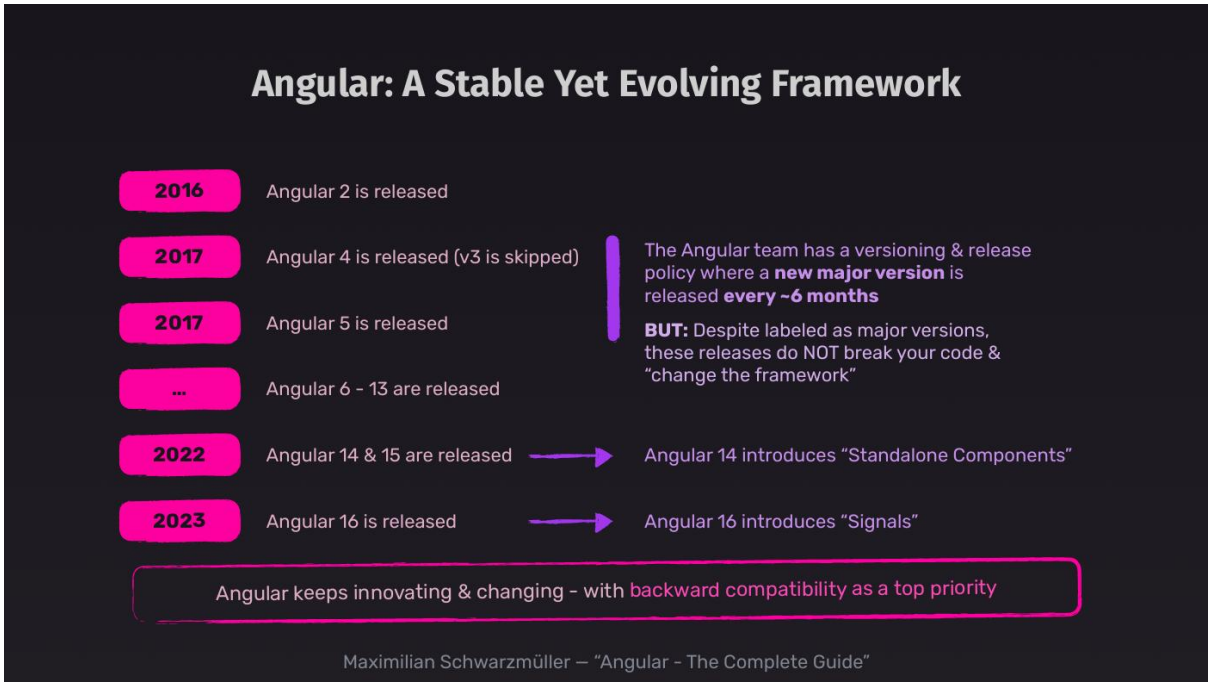
- Angular **2** was the first major release of the modern Angular framework (in 2016).
- It was a **complete rewrite** of AngularJS (also known as Angular 1).
- AngularJS and Angular 2+ are **totally different frameworks**, even though they share a name and were developed by many of the same team members.

Since Angular 2, the Angular team has followed a **predictable release cycle**:

- **Two major versions per year** (approximately every six months)
- Example versions: Angular 4, 6, 8, 10, up to **Angular 17** and beyond

⚠️ But **new versions don't mean major breaking changes**.

In fact, Angular has been one of the most **stable frameworks** on the market. Much of the code written for Angular 2 still works today with minimal changes.




---

## Backward Compatibility Matters

Even though Angular adds **new features over time**, these additions are introduced in a **non-disruptive** way:

- Existing code continues to work
- Features are often **optional** (you don't have to adopt them immediately)
- Deprecations are handled gradually, giving teams time to migrate

This means you can learn Angular **once** and continue to apply your knowledge across versions with confidence.

---

## Real-World Relevance

Not all companies and teams use the **latest version** of Angular.

In fact:

- Many projects are built using **older versions** (even Angular 9 or earlier)
- Teams may choose not to upgrade due to stability, time constraints, or dependencies

That's why **this course is designed** to help you regardless of the version you're using:

- Older content is preserved in the course
- Version-specific features are clearly labeled

- Compatibility notes are added to help you understand which features work in which versions

---

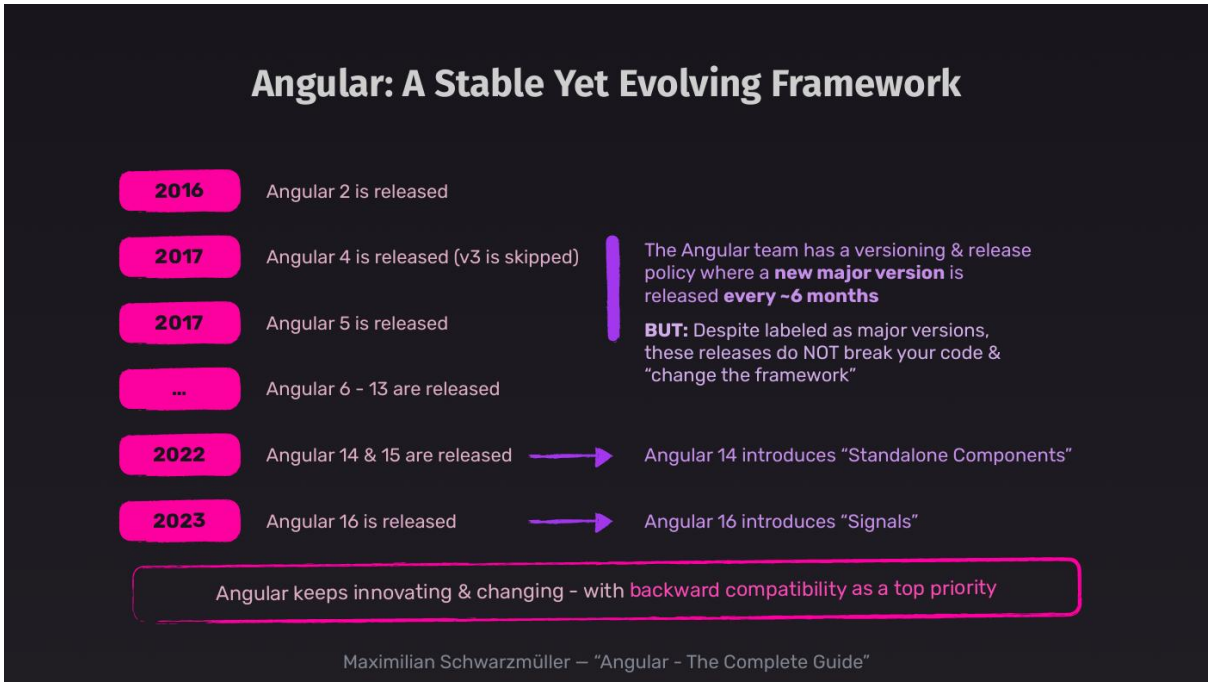
## Important Version Milestones

Two major releases introduced groundbreaking but **optional** features:

- **Angular 14** introduced **standalone components**
- **Angular 16** introduced **signals** for more reactive state handling

You'll learn about both of these in detail **throughout the course**.

These features are only available in **modern Angular versions**, but they're **not required**—you can still write Angular apps the "classic" way.




---

## Course Strategy

As the framework evolves, this course evolves too. Here's how:

- New features are added in updates
- Version compatibility is clearly marked
- You'll learn Angular in a **modern**, **future-proof**, and **practical** way

My goal is to ensure that:

- You're prepared for **any Angular version**

- You're comfortable working on **any project**—whether legacy or greenfield
- You understand both **classic** and **modern** Angular paradigms

---

### Let's Get Started

Now that you understand:

- How Angular has evolved
- Why it's a stable and future-ready choice
- That this course will guide you across versions

…it's time to dive in and **start building with Angular!**

---

# Creating a new Angular project

---

## Lesson 5: Setting Up Your First Angular Project

### Why You Need a Special Setup for Angular

To start learning and writing Angular code, you first need an **Angular project** where you can:

- Write and organize your code
- Run and test it in the browser

Unlike basic JavaScript projects, you **can't simply create an empty folder** with an `index.html` and `script.js` file and start coding. Angular projects involve advanced features that:

- Use **non-standard HTML syntax**
- Are written in **TypeScript**, which the browser doesn't understand directly

This means the code must be **compiled and optimized** before it runs in the browser.

---

### The Angular CLI (Command Line Interface)

https://angular.dev/tools/cli

To make this process easy, the Angular team provides the **Angular CLI**, a powerful tool used to:

- **Create** Angular projects
- **Run development servers**
- **Compile and optimize** your TypeScript and HTML templates
- **Build your app for production**

As an Angular developer, you'll use the Angular CLI constantly—often behind the scenes.

---

## Installing the Angular CLI

To install the Angular CLI, you need to perform two steps:

### ✅ Step 1: Install Node.js

Angular is not a Node.js framework—it runs in the **browser**.
However, the Angular CLI **depends on Node.js** to work.

1. Visit **nodejs.org**
2. Download and install the **LTS (Long-Term Support)** version
3. Complete the installation with the default settings

This installation will also include **npm** (Node Package Manager), which you'll need to install the CLI.

### ✅ Step 2: Install Angular CLI via npm

Once Node.js is installed, open your terminal or command prompt and run:

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

On **macOS or Linux**, you may need to prefix the command with `sudo`:

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

On **Windows**, don't use `sudo`—just run the command directly.

This command installs the Angular CLI **globally** so you can use it anywhere on your system.

---

## Creating a New Angular Project

Once the CLI is installed, follow these steps:

1. Open your terminal/command prompt
2. Navigate to the folder where you want your project

3. Run the following command:

```
ng new first-angular-app
```

- `ng` is the Angular CLI command
- `new` tells it to create a new project
- `first-angular-app` is your project's name (use **kebab-case**, no spaces)

---

## CLI Prompts During Project Creation

Depending on your **Angular version**, you may get different setup questions. Here's how to handle them:

- **Zoneless Angular App**: Choose **No** (or the default). We'll cover zoneless Angular later.
- **CSS Preprocessor**: Choose **CSS** for now (you can pick SCSS, SASS, etc. later if needed).
- **Server-Side Rendering (SSR)**: Choose **No** for now. We'll explore SSR later in the course.

If you're unsure about any question:

- Just **hit Enter** to accept the default option.

Once confirmed, the CLI will:

- Scaffold the entire Angular project
- Install all necessary dependencies
- Set everything up for immediate development

---

## ✅ Summary

By the end of this setup:

- You'll have a ready-to-use Angular project
- You'll be using TypeScript, even if you're new to it (no worries—you'll learn as we go!)
- Your project is CLI-powered, making development smoother and faster

Next up: **Running the Angular app and exploring the project structure**!

---

# Setting Up an Angular Development Environment

## Lesson 6: Setting Up Your Development Environment

### Choosing a Code Editor

To write and manage your Angular project, you'll need a **code editor or IDE**. You're free to use any editor you like, but if you don't have a preference, the **recommended choice is**:

### ✅ Visual Studio Code (VS Code)

- Free and open-source
- Cross-platform (Windows, macOS, Linux)
- Highly extensible through extensions
- Built-in terminal support

Download it from https://code.visualstudio.com and install it using default settings.

---

### Opening Your Angular Project in VS Code

Once installed:

1. Launch **VS Code**
2. Use `File > Open Folder` to open the folder you created with the `ng new` command
3. You'll see a file and folder structure like this:

The exact structure may vary depending on Angular CLI and version updates.

We'll **explore these files and folders** in detail in the next section.

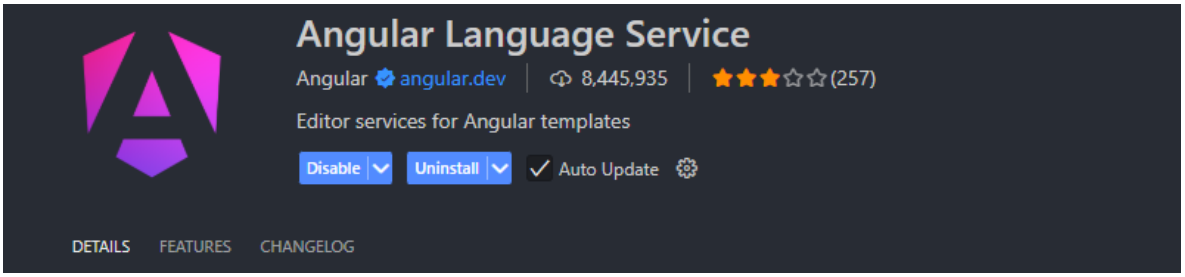
---

### Enhancing VS Code for Angular Development

To improve your development experience, install some **Angular-related extensions**.

### 🚀 Recommended Extensions:

1. **Angular Language Service**
   - Offers error detection, code completion, and syntax support specific to Angular.




2. **Angular Essentials (Optional)**
   - A bundle of helpful Angular tools and extensions.


To install:

- Click on the **Extensions** icon in the sidebar
- Search for `Angular`
- Install **Angular Language Service** first
- Optionally install **Angular Essentials**

Once done, return to the **Explorer view** and you're ready to begin development.

---

## Running Your Angular App Locally

Let's **preview the Angular app** in your browser by starting the development server.

### 🧪 Step-by-Step:

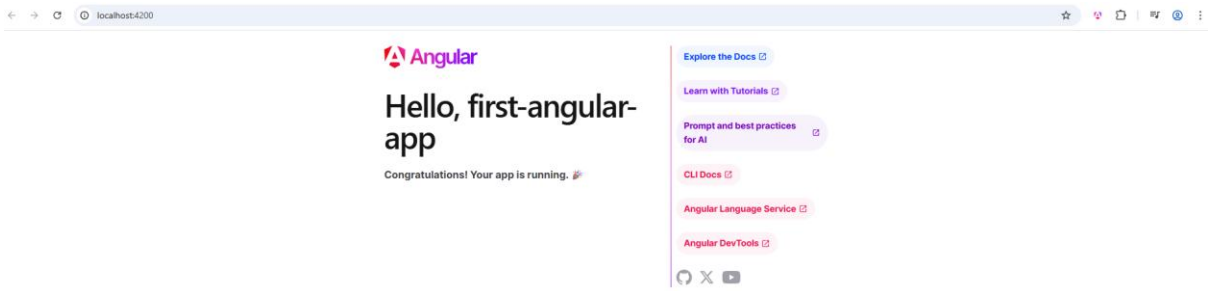
1. In VS Code, go to **Terminal > New Terminal**
2. This opens an integrated terminal **already navigated** to your project folder
3. Run:

```
npm start
```

This executes a built-in script from your Angular project which:

- Compiles the app
- Starts a local development server
- Opens your app in a browser at:
- `http://localhost:4200`

You should see the **default Angular welcome screen**.




The content may vary based on the CLI version—but as long as something loads, your setup is working.

---

## Live Reload and Development Workflow

With the development server running:

- Your app **auto-reloads** in the browser whenever you make changes and save files
- You don't need to manually refresh—Angular handles it for you

### 🛑 Stopping the Server:

- Press `Ctrl + C` in the terminal
- Or click the **X (Close)** icon in the terminal tab
- To restart later, simply run `npm start` again

Keep the server running while coding so you can see your changes live.

---

## ✅ Summary

In this lesson, you've:

- Chosen and set up your editor (preferably VS Code)
- Opened your Angular project
- Installed helpful extensions
- Learned to run your Angular app locally with `npm start`
- Understood the importance of **live reload** for development

---

Next, we'll dive into a **new demo project**, and more importantly, begin learning the **core building blocks of Angular**—components, templates, data binding, and more.

---

# About the course

## Lesson 7: How This Course Teaches Angular

### What to Expect from This Course

You might be wondering:

How does this course teach Angular?
What is the best way to approach the content?

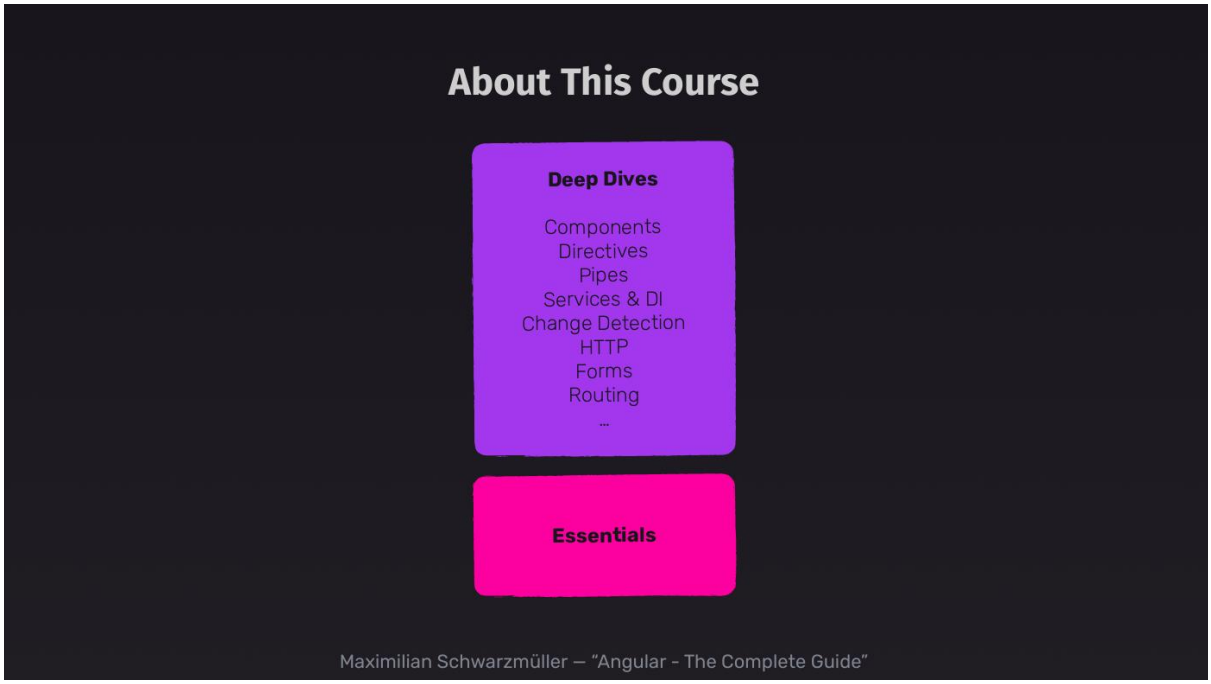
While you can always browse the **course curriculum** and jump into sections that interest you most, the course is designed with a **progressive learning path** in mind.

### Course Structure: Essentials First, Then Deep Dives

The course follows this logical flow:

1. **Start with the Angular Essentials**
   - These first few sections introduce the **core Angular concepts**
   - They are essential no matter what kind of project you're working on
   - You'll build a solid foundation in Angular
2. **Move into Deep Dives**
   After covering the essentials, we explore more advanced and detailed topics such as:
   - Components
   - Directives
   - Pipes
   - Services and Dependency Injection
   - Routing
   - HTTP
   - State management
   - Signals
   - And much more

This structure ensures a strong foundation before tackling more advanced features.




---

## Flexible Learning Path

Although the course is best followed **step by step**—lecture by lecture—you're not locked into a linear path.

- Already familiar with Angular basics?
  You can **skip ahead** to any advanced topic that interests you (e.g., Services, Routing, Signals).
- **Brand new to Angular?**
  Start at the beginning and **don't skip the essentials**.

---

## Two-in-One: Modern + Legacy Angular

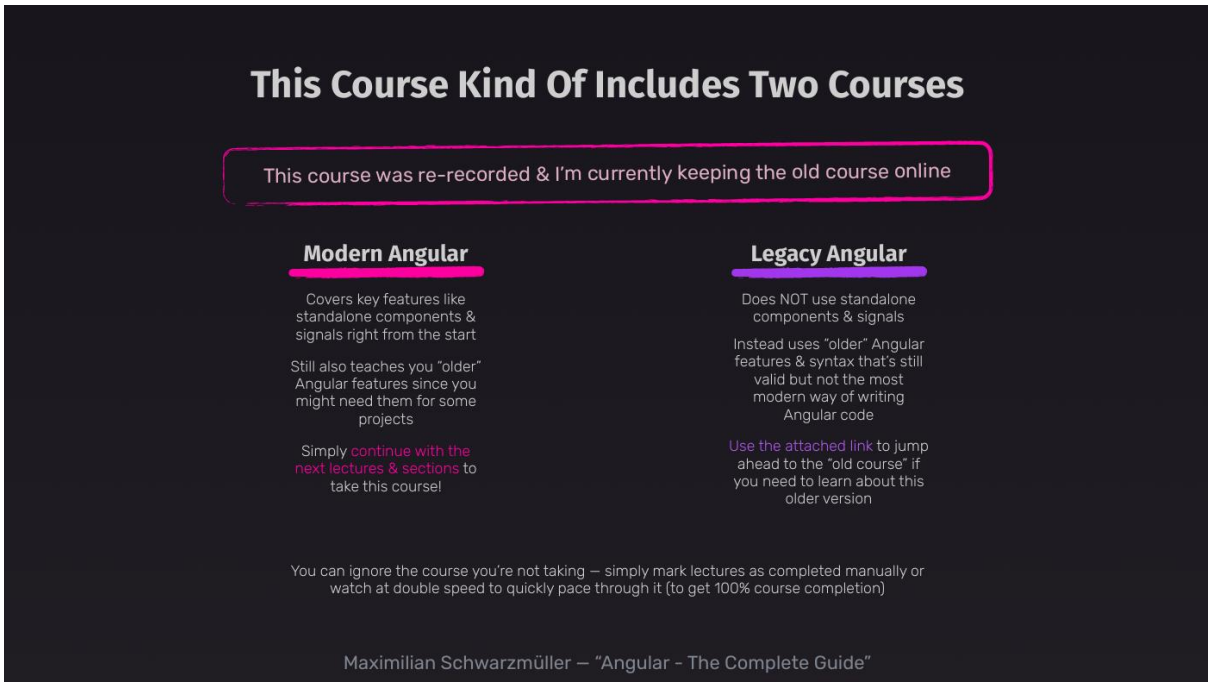
This course is **actually two courses in one**:

1. **Modern Angular Course**
   - Fully updated and re-recorded to teach Angular with:
     - **Standalone components**
     - **Signals**
   - These modern concepts are integrated from the **very beginning**
2. **Legacy Angular Essentials (Optional)**
   - Retained for learners working with **older Angular versions**
   - Includes the "classic" Angular setup using `NgModules`, etc.
   - You'll find a link to jump directly to that version if needed

For most students, **sticking to the modern course** is the best approach.

However, if you already know you'll be working on a project that does **not** use signals or standalone components, you can start with the **older essentials** section.

A link is provided in the course to jump directly to that content.




---

## ✅ Summary

- Learn Angular progressively: **Essentials → Deep Dives**
- Most effective when followed **step by step**
- Optionally skip sections if you're experienced
- Covers **both modern and older Angular** approaches
- Signals and standalone components are covered **early**
- Legacy sections are available if needed

---

Next up: we'll begin learning about **Angular components**, the most fundamental building block in any Angular application.

---

# Course Setup & Resources

This is a huge course with many demo projects, examples & concepts. Here are some resources that should help you with taking this course:
**Academind Learning Community**

As a student of this course you also get access to the **Academind Learning Community** - a free Discord channel you can join to find like-minded students. You can search for learning partners, ask questions, share projects and, in general, boost your learning experience by diving into this community. Of course, joining is not mandatory and you can leave at any time!

**IDE / Code Editor**

For following along with this course, you can use VS Code - a free, extensible IDE that works on all operating systems.

A **popular alternative** are the IDEs offered by Jetbrains. They offer a variety of IDEs for different programming languages and projects. For web development, Webstorm is a popular choice.

As a student of this course, you can get **one Jetbrains IDE for free** for six months! All you need to do, is select it and enter the following coupon code on the checkout page: **ACADEMIND_JETBRAINS.**

You find detailed instructions regarding the coupon code usage on this page.

*Please note: Using Jetbrains IDEs is optional (you could use VS Code)! If you do go for the Jetbrains deal, you'll need to enter your email address during the checkout process.*

**Course Code Snapshots**

When coding along, it's possible that you run into errors from time to time. In order to help you with those, I prepared a **course code repository** where you can find **multiple code snapshots for every course section.** You can use those snapshots to compare your code to mine and hopefully resolve any issues you might encounter.

**Course Slides**

Whilst the majority of the videos in this course shows you hands-on demos & examples, I also do use some slides in some of the videos. You can download those from the course code repository.

**TypeScript Introduction**

This course also uses TypeScript as a programming language since Angular uses TypeScript. The good news is: You don't need to know TypeScript to take this course! You'll learn it along the way.

But if you ever feel like you don't fully understand a certain TypeScript concept or you're having a hard time following along with the TypeScript code shown in the videos, you can dive into the dedicated **"TypeScript Introduction" course section** I prepared for you.