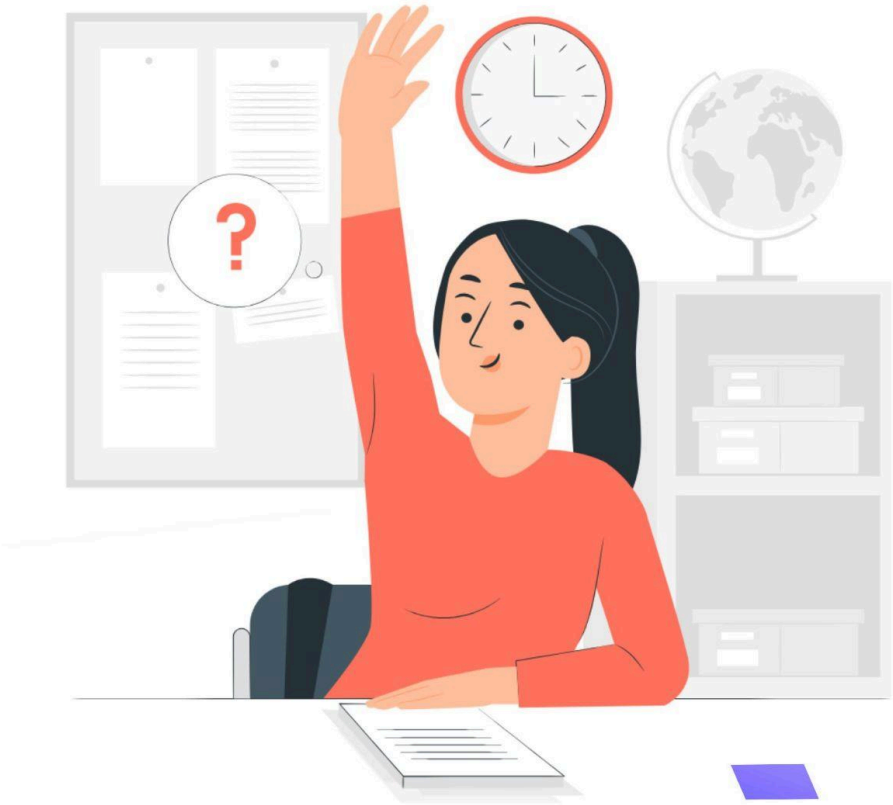
Angular for Beginners

Everything you need to know



Abdelfattah Ragab

Angular for Beginners

Everything you need to know

Abdelfattah Ragab

Introduction

Welcome to the book “Angular for beginners”.
In this book, I'll tell you all about Angular in a free discussion format.
What is Angular, how can you build a real application with it and what are all the terms you hear every day like pipes, interceptors, lazy loading and so on.
You will learn everything in a few minutes and have a good understanding of what Angular can do and how you can use it for your applications. By the end of this book, you will realize that Angular is so easy, and you will be interested in moving on and creating your own Angular applications.
Let's get started.

Move forward

When you start a new topic, you should explore it broadly and focus on moving forward without getting lost in the details. I also recommend that you gather information on the same topic from different sources. Take breaks from time to time, look for short answers to questions, and use different sources to get different perspectives. Don´t worry if you still don't understand after many attempts, just keep going because you will come across it again later.

What is Angular?

Angular is a web framework that empowers developers to build fast, reliable applications. Maintained by a dedicated team at Google, Angular provides a broad suite of tools, APIs, and libraries to simplify and streamline your development workflow. Angular gives you a solid platform on which to build fast, reliable applications that scale with both the size of your team and the size of your codebase.

How to use Angular to create real life applications?

You use Angular to create components, connect them together and deploy the application online. It's like building block toys.

How do you organize your Angular application?

The most important building block in Angular is the component. When you connect your component to a route, it becomes a page.
If you use it within another component, it becomes a shared component.
You can create a "Layout" folder and store the components of Header, Footer, Sidenav and so on in it.

You can create another folder "Components" and place the common components such as product card, rating etc. in it.
You can create another folder "Pages" and store the components for products, shopping cart, about, contact etc. in it.
You can create folders for all kinds of artifacts such as "Pipes", "Guards", "Interceptors", "Services", "Directives" and so on.

What is Angular CLI?

Angular CLI is an easy way to create all artifacts like components, services, guards and so on.
Open the terminal and run the commands:

ng g c pages/about

Creates the "about" component in the pages folder

ng g s services/products

Creates the "products" service in the services folder

ng g p pipes/truncate Creates the "truncate" pipe in the pipes folder

What is the Angular component?

An Angular component is a fundamental building block of an Angular application. It encapsulates a part of the user interface (UI) and defines how this part of the application behaves. Each component consists of three main parts:

Component class: This is a TypeScript class that contains the logic for the component.

Template: The template is an HTML file that defines the view for the component.

Metadata: Metadata is provided with the @Component decorator, which tells Angular how to process the component. This includes information such as the component's selector (the HTML tag used to embed the component), the path to the template and the styles associated with the component

Here's an example of what the products.component.ts file might look like:

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

@Component({
  selector: 'app-products,
  templateUrl:
'./products.component.html',
  styleUrls:
['./products.component.css']
```

```
})
export class ProductsComponent {

}
```

To generate it you need to execute the command
`ng g c pages/products`

Under the pages folder you will find the products folder component and it will have the following files:

- `products.component.html`: The HTML template for the component.
- `products.component.css`: The CSS styles for the component.
- `products.component.ts`: The TypeScript file that defines the component class.
- `products.component.spec.ts`: The unit test file for the component.

Events

In the html template you write normal html code in which you can use div, input, button and so on.
For each element you can handle events by defining the name of the handler method. In the Typescript class, you define the logic of the handler.
In the html:

```
<button (click)='onClick()" ></button>
```

In the TypeScript class:

```
onClick() {
   console.log('Clicked');
}
```

Routing

Angular routing allows you to manage the navigation and rendering of different views or components based on the current URL.
The key concepts are:
Routes: Routes are defined as an array of objects, where each object specifies a path and the corresponding component to be displayed when accessing this path. Usually it is defined in a file called *app.routes.ts*

```
import { Routes } from
'@angular/router';
import { AboutComponent } from
'./pages/about/about.component';
import { ProductsComponent } from
'./pages/products/products.component';

export const routes: Routes = [
  { path: 'about', component:
AboutComponent },
  { path: 'products', component:
ProductsComponent },
];
```

When you type the ‘/about’ path in the address bar, the browser will display the AboutComponent.
Router Outlet: The `<router-outlet>` directive acts as a placeholder where the routed component will be displayed.

For example I can define the app.component.html as follows:

```
<app-header></app-header>
<div class='main'>
  <router-outlet/>
</div>
<app-footer></app-footer>
```

Component Lifecycle

A component's lifecycle is the sequence of steps that happen between the component's creation and its destruction.
The components go through four phases: Creation, change detection, rendering, and destruction.
The creation has only one method, the constructor, and the destruction also has only one method, namely ngOnDestroy.
Creation is followed by the change detection phase, which begins with the ngOnInit method.
The ngOnInit method is executed exactly once before the component's own template is initialized. This means that you can update the state of the component based on its initial input values.