Alliance For Digital Employability

Angular

8.2.8

F U N D A M E N T A L S

[Updated: 28/09/2019 | Curated by Kostas Minaidis & Aristeidis Bampakos]

What is Angular?

Angular is a platform and framework for building client applications in HTML and TypeScript.

Architecture:

The basic building blocks of an Angular application are NgModules, which provide a compilation context for a set of components and is dedicated to an application domain, a workflow, or a closely related set of capabilities.

Alliance For Digital Employability

SPA

Single Page Application

A web application that fits on a single page

- General Concepts:
  - All necessary code can be retrieved in a single page load or dynamically loaded as necessary.
  - As the user interacts with the Application, data is sent and received from the Server using Ajax requests or WebSockets.
  - Templating is often used in SPAs.
  - Routing: client-side routing dynamically loads a piece of the application without reloading: `https://spa.com/#/about`

    Routes are commonly placed in the fragment identifier section of the URL:* ***https://spa.com/#/about*
- Advantages:
  - Improved performance
  - Fluid user experience
  - Reduced page loads & smaller server transactions

SPA Resources

- Angular Single Page Applications (SPA): What are the Benefits?
- Are Angular applications SPAs?
- Why angular is called single page application
- What is SPA in AngularJS?

Alliance For Digital Employability

Angular

Prerequisites

(Background knowledge)

- Moderate knowledge of **HTML**, **CSS**, & **JavaScript**
- Basic concepts of **ES6**
  *Modern JavaScript
- **TypeScript** (Basics)
- Basic **MVC*** **concepts**
  *Model-View-Controller
- Basic **OOP*** **concepts**
  *Object Oriented Programming

Alliance For Digital Employability

Prerequisites

(Software)

- Node.JS v10.9.0 or later
- Node Package Manager (npm)
  Comes pre-installed with Node.JS
- Angular CLI:
  `npm install -g @angular/cli`

Resources

- **Basics**
  - Setting up the Local Environment & Workspace
  - Introduction to ES6+
  - Learn modern JavaScript
  - MVC Architectural Pattern | Mosh
  - Object-oriented Programming in JavaScript: Made Super Simple | Mosh

- **TypeScript**
  - TypeScript Tutorial for Angular & React Developers | Mosh
  - Introduction to TypeScript | Interactive Screencasts (55 minutes)
  - TypeScript Tutorial | Derek Banas
  - Understanding TypeScript Basics
  - TypeScript in 5 minutes
  - Learn TypeScript in 5 minutes
    A tutorial for beginners

Quickstart

Alliance For Digital Employability

1) Install the Angular CLI globally. You will need to execute this command only once on your system:

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

2) Use the newly installed CLI tool to create a new Angular App:

```
$ ng new my-app
```

Would you like to add Angular routing? (y/N)
Press y, to enable Routing and press Enter.
Which stylesheet format would you like to use?
Hit enter to go with the default CSS format.

3) Change into the newly created App directory:

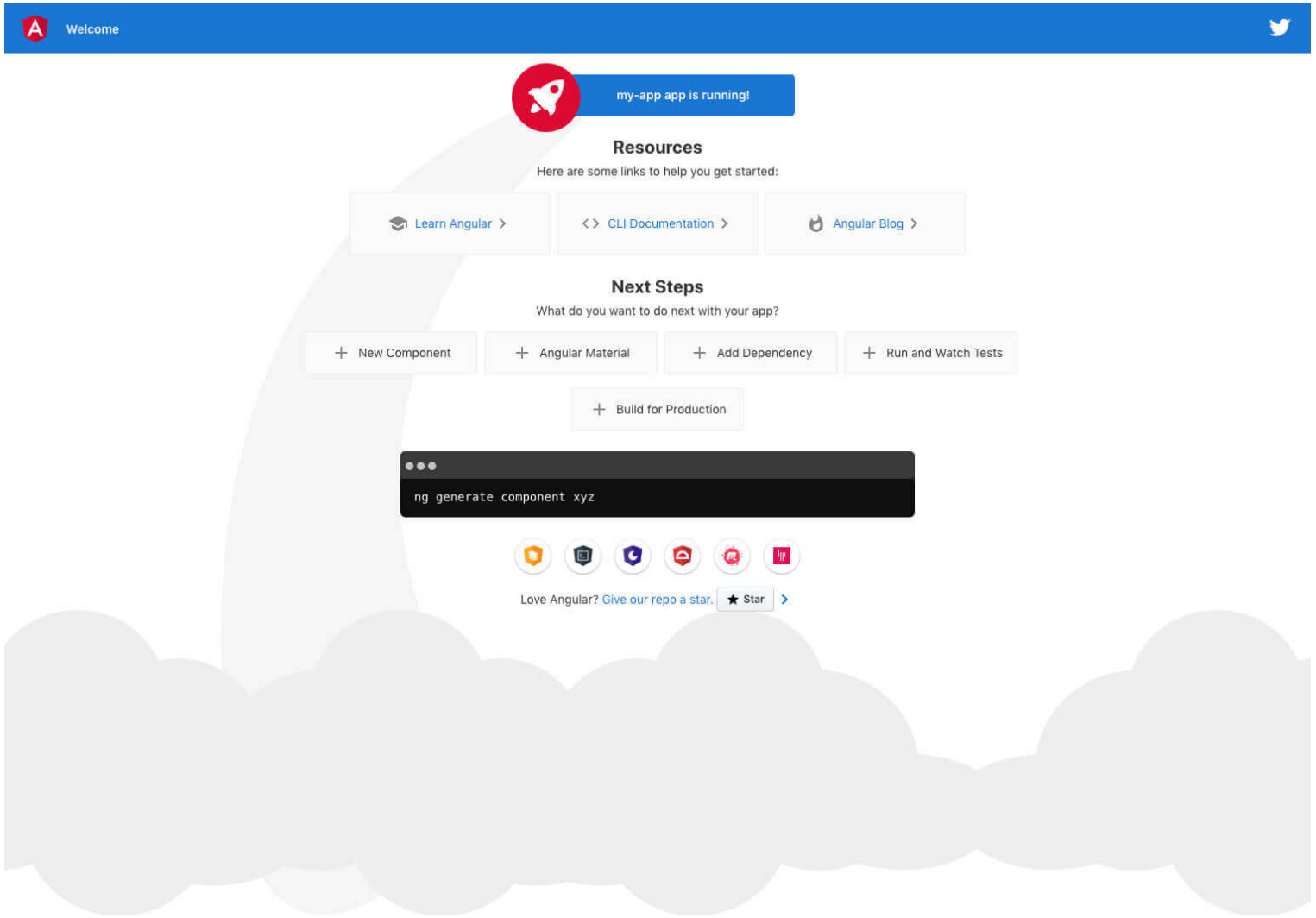
```
$ cd my-app
```

4) Start the development server
and watch the App running on your browser:

```
$ ng serve --open
```

Source: **https://angular.io/guide/quickstart**

This is the screen that you should be seeing in your browser, once all the steps have been completed successfully:



Let's complete our Quickstart guide,
by editing the main App page:

1) Open the following file in your editor:
`/my-app/src/app/app.component.ts`

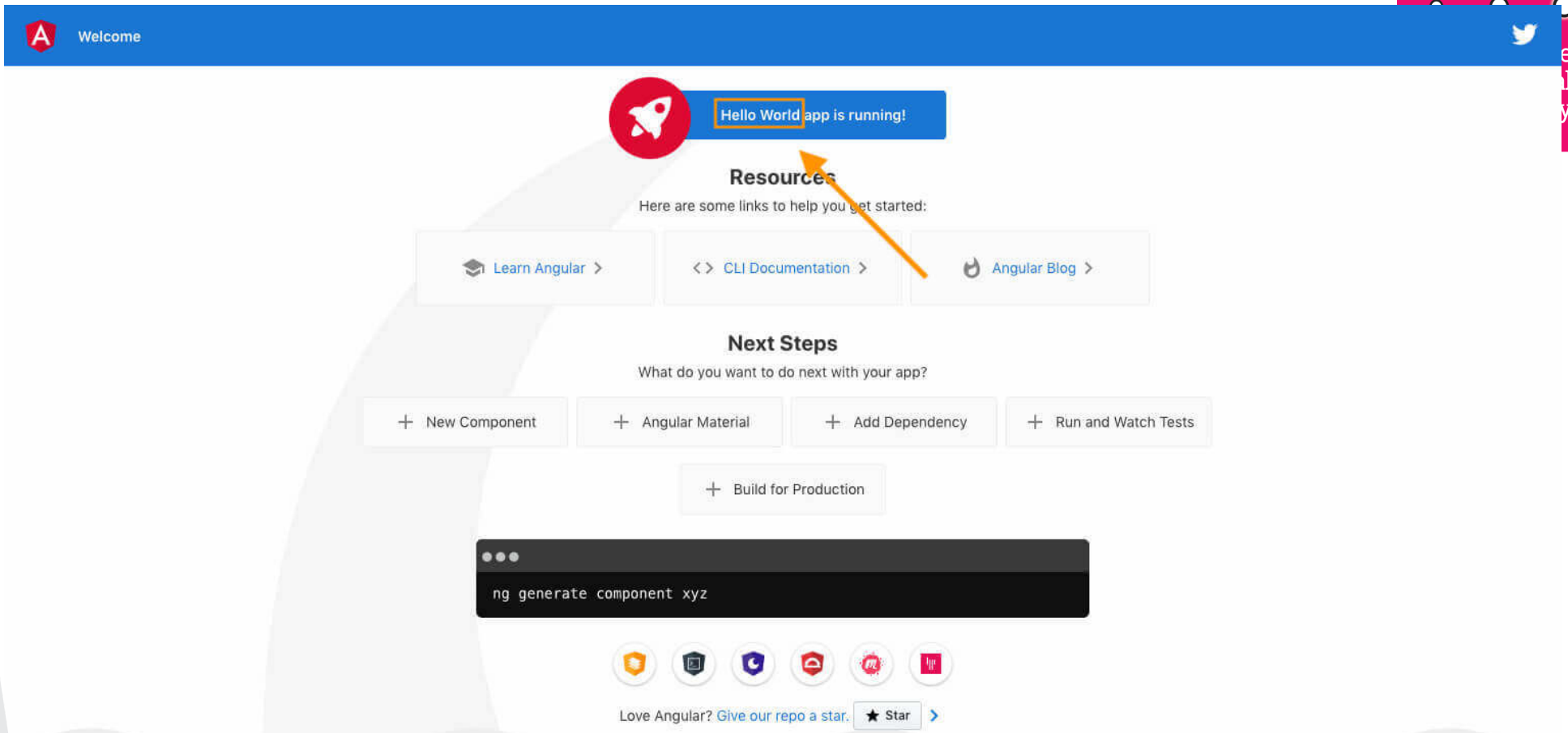
2) Find and replace the following line:

```
title = 'my-app';
```

With:

```
title = 'Hello World';
```

3) Watch how the Browser automatically reloads the page with the updated title:


Welcome
Hello World app is running!
Resources
Here are some links to help you get started:
Learn Angular
CLI Documentation
Angular Blog
Next Steps
What do you want to do next with your app?
New Component
Angular Material
Add Dependency
Run and Watch Tests
Build for Production
ng generate component xyz
Love Angular? Give our repo a star.
Star

Alliance For Digital Employability

Workspace & Project File Structure

- .editorconfig
- .gitignore
- README.md
- angular.json
- package.json
- package-lock.json
- tsconfig.json
- tslint.json
- node_modules/
- **src/**
  *This is the folder where we will be spending most of our time.

Official Reference / Interactive Overview

Alliance For Digital Employability

Angular Roadmap

Modules

Blocks of code dedicated to a specific set of capabilities.

Components

UI Code / MVC Views

Services

Reusable Code of Business Logic.

Routing

Handles multiple page Views.

Modules

A **Module** is a piece of software that is grouped by responsibility.

It encapsulates a defined set of functionality and provides a well-defined interface for using this module.

Source: Michał Michałowski

`Angular` apps are **modular** and Angular has its own modularity system called `NgModules`.

NgModules are containers for a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities.

Alliance
For Digital
Employability

NgModule Walkthrough:

`/src/app/app.module.ts`

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';

@NgModule({
    declarations: [ AppComponent ],
    imports: [ BrowserModule ],
    providers: [],
    bootstrap: [ AppComponent ]
})
export class AppModule { }
```

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';

@NgModule({
    declarations: [ AppComponent ],
    imports: [ BrowserModule ],
    providers: [],
    bootstrap: [ AppComponent ]
})
export class AppModule { }
```

BrowserModule provides services that are essential to launch and run a browser app.

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { AppComponent } from './app.component';
4
5  @NgModule({
6      declarations: [ AppComponent ],
7      imports: [ BrowserModule ],
8      providers: [],
9      bootstrap: [ AppComponent ]
10 })
11 export class AppModule { }
```

The NgModule Decorator *(used on line #5)* is imported from the Angular Core library.

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { AppComponent } from './app.component';
4
5  @NgModule({
6      declarations: [ AppComponent ],
7      imports: [ BrowserModule ],
8      providers: [],
9      bootstrap: [ AppComponent ]
10 })
11 export class AppModule { }
```

We import the AppComponent components to our Module.

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { AppComponent } from './app.component';
4
5  @NgModule({
6      declarations: [ AppComponent ],
7      imports: [ BrowserModule ],
8      providers: [],
9      bootstrap: [ AppComponent ]
10 })
11 export class AppModule { }
```

We use the @NgModule decorator *(imported on line #2)* and the supplied metadata *(the object argument)* to mark the AppModule class as an NgModule.

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { AppComponent } from './app.component';
4
5  @NgModule({
6      declarations: [ AppComponent ],
7      imports: [ BrowserModule ],
8      providers: [],
9      bootstrap: [ AppComponent ]
10 })
11 export class AppModule { }
```

We are ready to export our AppModule to the world.

Alliance
For Digital
Employability

Components

Help us break down an Application's Interface into **individual parts**.

Components can be **re-used** and allow us to work efficiently, especially with complex applications, large codebases and large developer teams.

Advantages

- Encapsulation
- Templating
- Reusability
- Composability
- Extensibility
- Efficiency

Reference

Alliance For Digital Employability

Why Components at all?

Encapsulation: A component should be completely separate from the main application. This increases reusability, testability and reliability because the component is only responsible for its internals and shouldn't be concerned with the state of the application, only its own. Both the component author and user can upgrade the component without fear of affecting the rest of the application.

Composability: It should be possible to create more complex components or even entire applications from a collection of components. The decreases the need for "global" logic providing a better defined architecture and less chance of bugs because each individual part of the application (or composite component) is better defined.

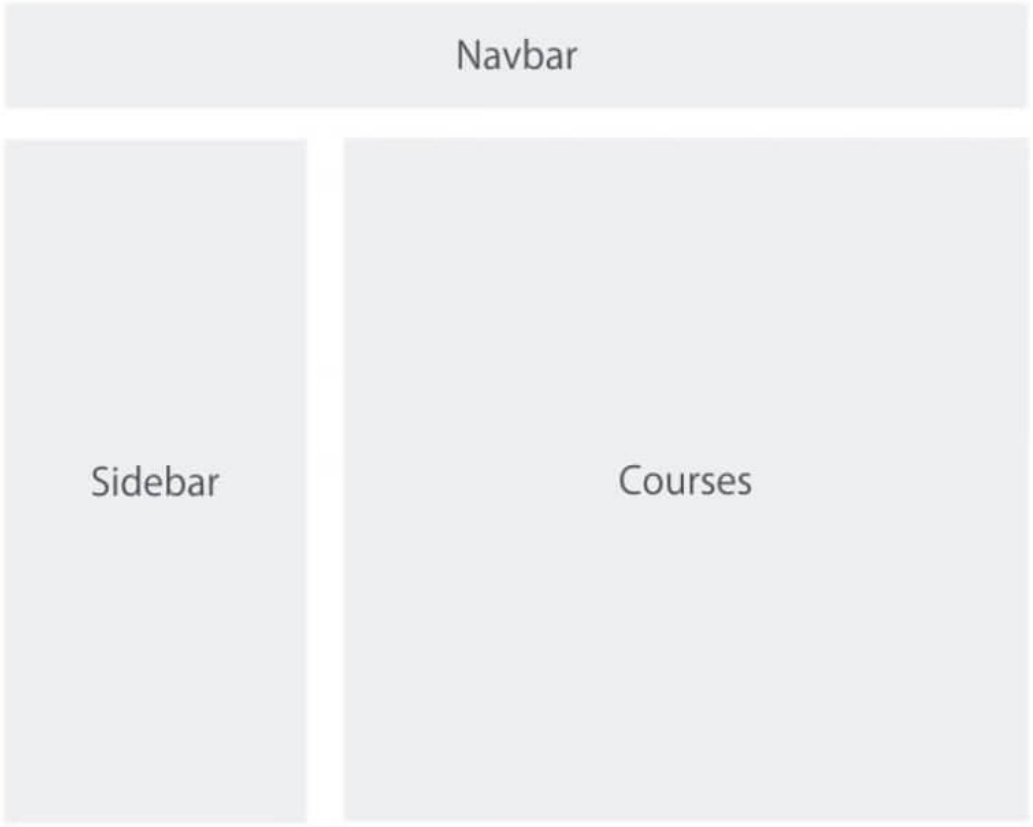
Extensibility: Components should be able to extend each other and in the case of web components other DOM elements. This means that a component author doesn't need to reinvent the wheel and can easily reuse functionality.

Reusability: The big one. With all of the above a component should be easily reusable with minimal dependencies and a clearly defined API.

Reference | By Leon Revill

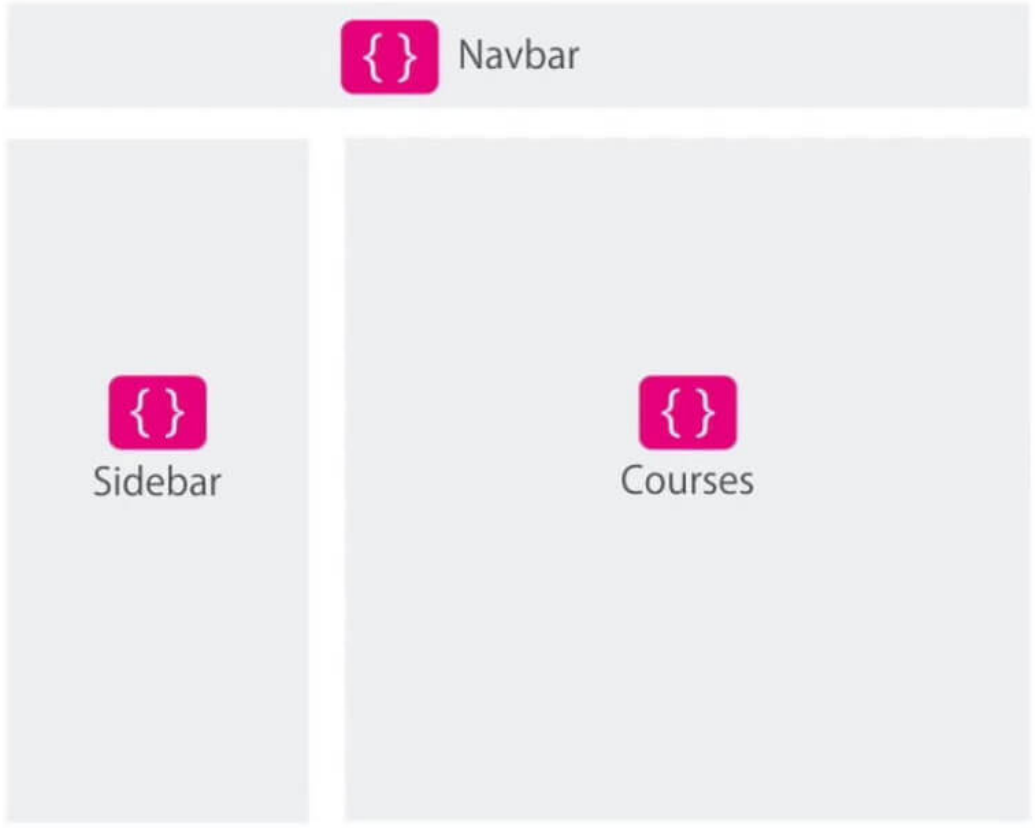
Breaking down an Application into Components

Online Learning Platform

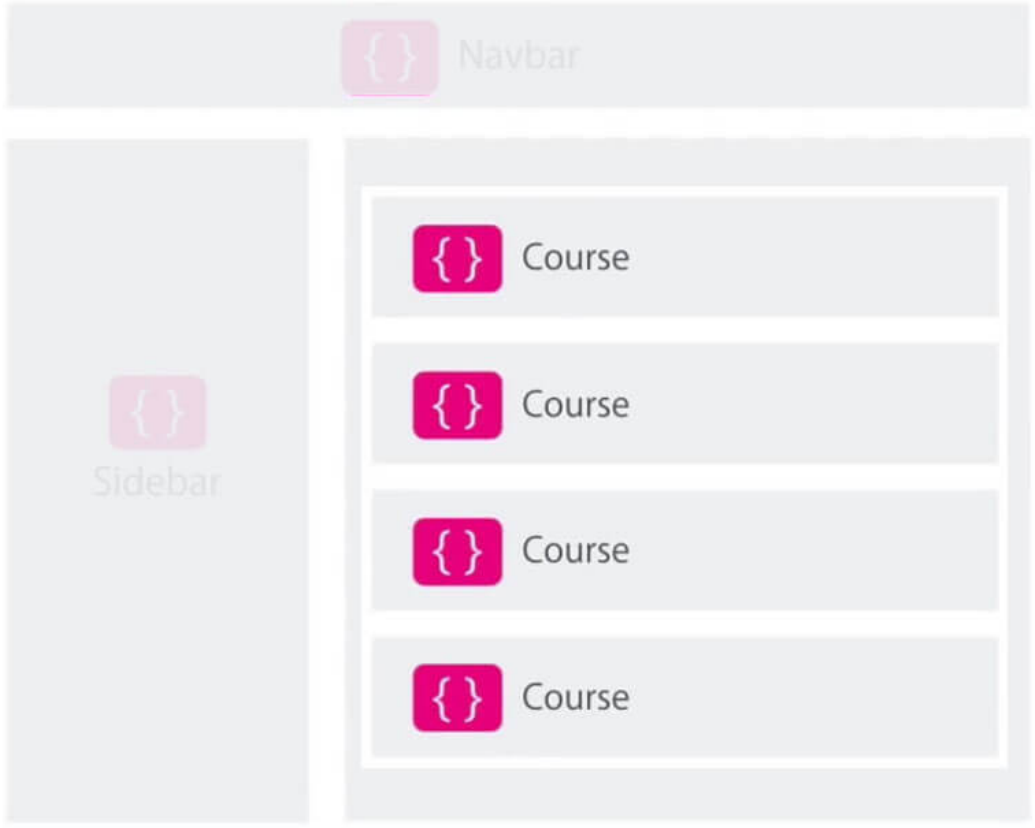


Online Learning Platform

Online Learning Platform



Online Learning Platform



Online Learning Platform

Alliance
For Digital
Employability

Resources

- Angular (Official) Documentation
- Glossary of Terms
- **Build your first Angular app Interactive Screencasts** (3 hours)