Курс «Використання сучасних Web-фреймворків»

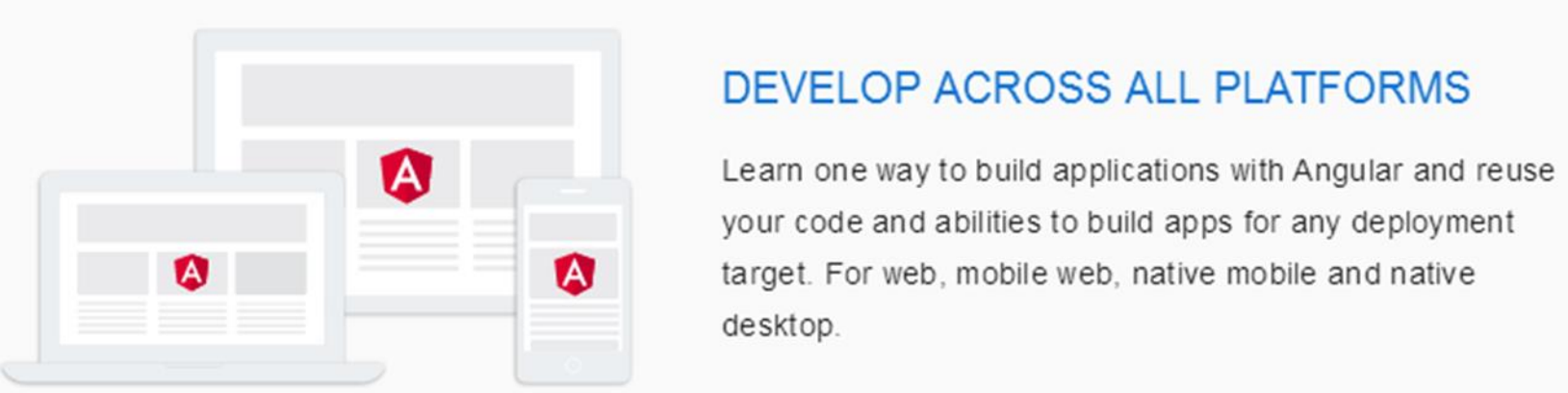
Тема «JavaScript фреймворк Angular»

План лекції

1. Чому Angular? Місце Angular серед інших web-фреймворків.
2. Основні ресурси для вивчення Angular.
3. Що необхідно для створення робочого середовища розробника.
4. Angular CLI.
5. TypeScript і Angular.
6. Швидкий старт. Створення першого проєкту.
7. Знайомство з архітектурою Angular. Як Angular працює.
8. Анатомія компонента Angular.
9. Дочірні компоненти. Взаємодія між компонентами.

Що таке Angular

- Angular — написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій.
- Поточна версія Angular v11.0.3 (квітень 2021)
- Angular – покращена версія Angular.JS (по суті заново написана)
- Сайт https://angular.io

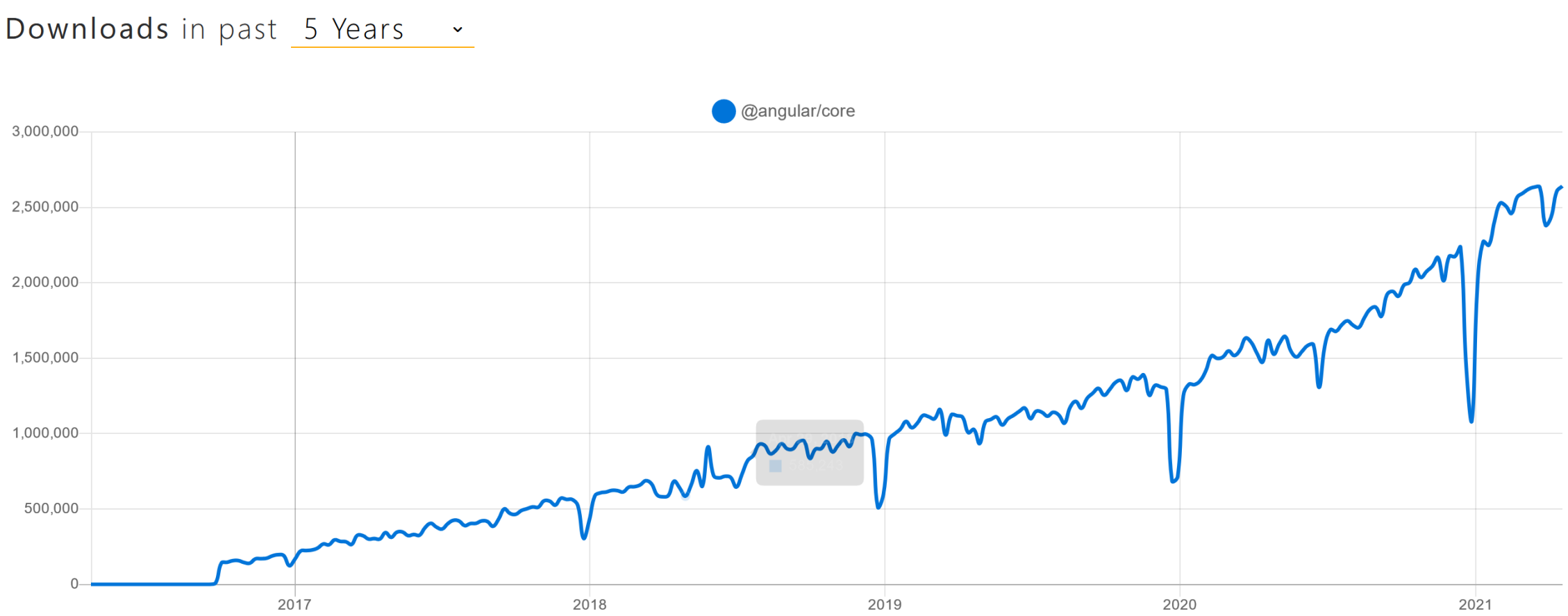


Графік випуску та підтримки Angular

Основні особливості Angular

- Має Angular CLI (command-line interface tool)
- Містить механізм створення користувацьких тегів і атрибутів HTML за допомогою концепції директив, яка дозволяє розширювати набір тегів HTML відповідно до потреб додатка.
- Будує архітектуру додатка у вигляді дерева web-компонентів
- Використовує мову TypeScript — надмножину ECMAScript 6 (ES6)
- Активно використовує прив'язку даних
- Містить модуль впровадження залежностей (dependency injection)
- Підтримує модульність
- Надає механізм для налаштування маршрутизації

Статистика користування Angular



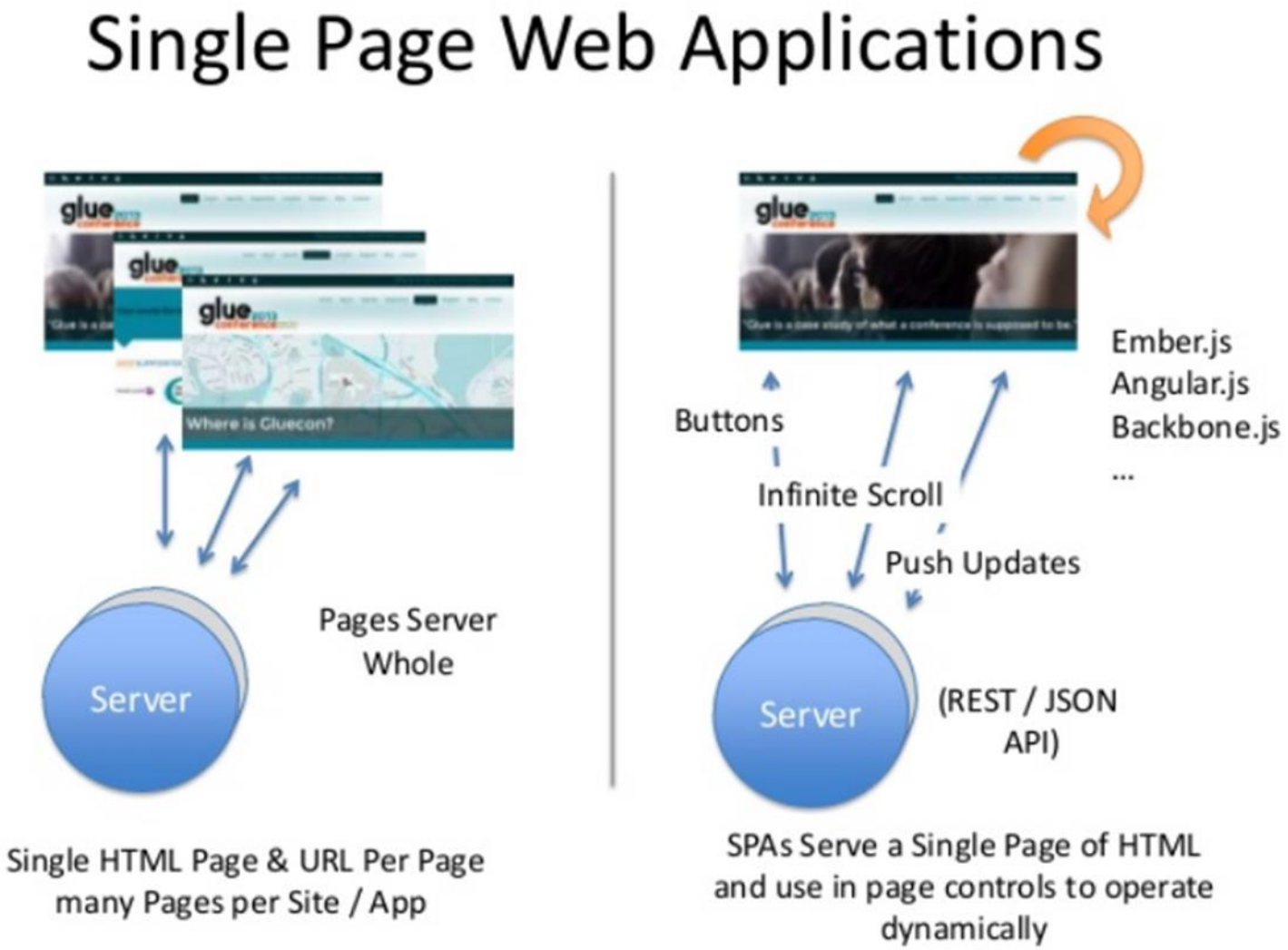
Корисні посилання

- Офіційний сайт Angular - https://angular.io
- 2020 Developer Servey - https://insights.stackoverflow.com/survey/2020
- Руководство по Angular 11 - https://metanit.com/web/angular2/

Література

1. Дворецький М.Л., Дворецька С.В. **Розробка односторінкових вебзастосунків та адаптивних інтерфейсів**. Навчальний посібник – Миколаїв, Чорноморський університет імені Петра Могили, 2020 – 69 с.
2. Bampakos A., Deeleman P. **Learning Angular**. Packt Publishing, 2020
3. Chivukula S.R., Iskandar A. **Web Development with Angular and Bootstrap**, Packt Publishing, 2019
4. Freeman A. **Pro Angular 9**: Build Powerful and Dynamic Web Apps, Apress, 2020
5. Файн Я., Моисеев А. **Angular и TypeScript**. Сайтостроение для профессионалов. — СПб.: Питер, 2018. — 464 с.
6. Фримен А. **Angular для профессионалов**. — СПб.: Питер, 2018. — 800 с.
7. Doguhan Uluca **Angular for Enterprise-Ready Web Applications**. — Packt Publishing, 2020
8. Valerio De Sanctis **ASP.NET Core 5 and Angular**: Full-stack web development with .NET 5 and Angular 11. — Packt Publishing, 2021
9. Oswald Campesato **Angular and Machine Learning**. — Mercury Learning and Information, 2020

Що таке SPA


Single Page Web Applications
Where is Gluecon?
Server
Pages Server
Whole
Single HTML Page & URL Per Page
many Pages per Site / App
Buttons
Infinite Scroll
Push Updates
Ember.js
Angular.js
Backbone.js
...
Server
(REST / JSON
API)
SPAs Serve a Single Page of HTML
and use in page controls to operate
dynamically

Інструментарій Angular-розробника

- **npm** – для завантаження утиліт і залежностей
- **Node.js** (http://nodejs.org/) – в якості середовища виконання і для запуску web-сервера
- **Angular CLI** – для генерації проєктів, компонентів, сервісів
- **Visual Studio Code** (https://code.visualstudio.com/) і деякі плагіни
- Браузер

Angular CLI

Встановлення Angular CLI:

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

Інтерфейс CLI має ряд функцій, які допомагають у розробці Angular додатків (https://cli.angular.io). Ось **основні функції**:

- Генерація файлової структури нового проекту (scaffolding)
- Генерація нових частин програми (components, services, routes, pipe)
- Керування всім інструментарієм створення
- Обслуговування сервера розробки localhost
- Підтримка тестування

CLI commands

- Angular CLI - це інструмент інтерфейсу командного рядка, який автоматизує конкретні завдання під час розробки.
- Як випливає з назви, він використовує командний рядок, щоб викликати виконуваний файл **ng** та запускати команди, використовуючи такий синтаксис:

ng command [options]

Основні команди:

- ✓ **ng help**
- ✓ **ng new my-first-project** - Створення нового проєкту
- ✓ **ng serve –o** - Побудова Angular додатка і завантаження на web сервер
- ✓ **ng generate component my-component** - генерація нового компонента
- ✓ . . . . . .

Примітка (Книга Learning Angular - Packt Publishing, 2020 p.67)

Налаштування терміналу в VS Code для Windows 10

Important Note

In Windows, the integrated terminal of VS Code uses **PowerShell** by default, which may prevent you from running Angular CLI commands due to security reasons. To change it, click on the dropdown that reads **1: powershell**, choose the **Select Default Shell** option, and select **Command Prompt**.

Modern JavaScript and Angular

Angular використовує багато функцій, які є досить недавніми для веб-платформи. Більшість з них стали частиною специфікації JavaScript ES2015 (також відомої як ES6):

- Classes
- Decorators
- Modules
- Template literals

. . . . .

Listing 1.1 Modern JavaScript Syntax

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

@Component({
  selector: 'my-component',
  template: `
<div>
  <h4>{{title}}</h4>
</div>
`
})
export class MyComponent {
  constructor() {
    this.title = 'My Component';
  }
}
```

Imports the Component object from another module

Uses a decorator to add metadata to the MyComponent object

Uses a template literal string to write inline HTML

Exports the MyComponent object, which was defined as a class

TypeScript and Angular

- Сам Angular написаний на мові TypeScript, яка є надмножиною JavaScript, що вводить перевірку типів.

Приклад (JavaScript).
```
let bill = 20;
let tip = document.getElementById('tip').value; // Contains '5'
console.log(bill + tip); // 205
```

Приклад (TypeScript).
```
let bill: number = 20;
let tip: number = document.getElementById('tip').value; // ‘5’, error!
var total: number = bill + tip; // error!
```

(www.typescriptlang.org/docs/tutorial.html)

Швидкий старт

Крок 1. Налаштування середовища розробки

- Інсталюйте Node.js (https://nodejs.org/en/download/)
- Далі інсталюйте Angular CLI (command line interface):
  - Запустіть консоль (cmd.exe) або VS Code і перейдіть до каталогу проекту
  - В командному рядку наберіть: npm install -g @angular/cli

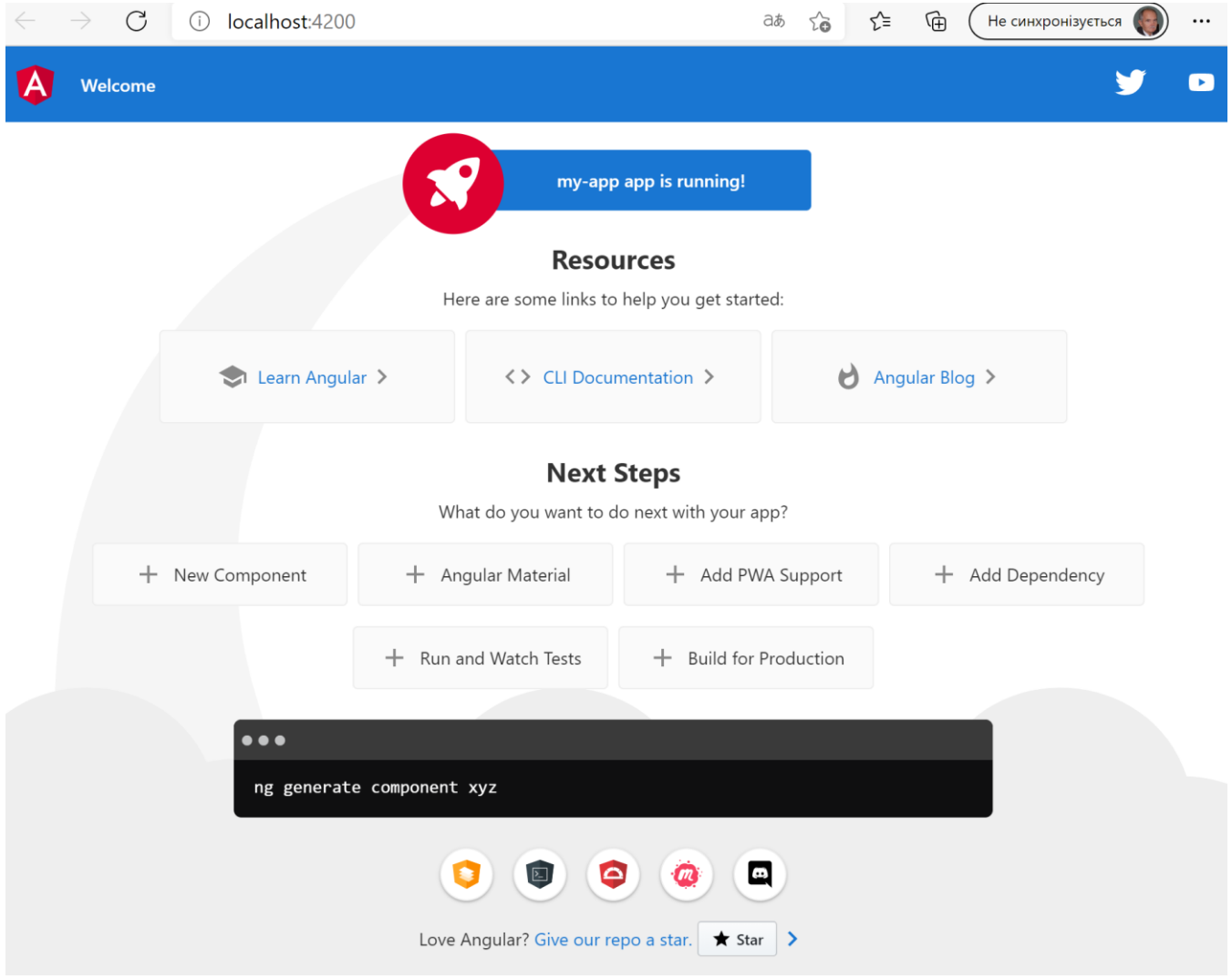
Крок 2. Створення нового проєкту

- ng new my-app

Крок 3. Запуск сервера

- cd my-app
- ng serve --open

- У відкритій вкладці http://localhost:4200/ за замовчуванням ви побачите стандартний макет Angular додатка :



- **Крок 4. Редагування нашого першого Angular компонента**

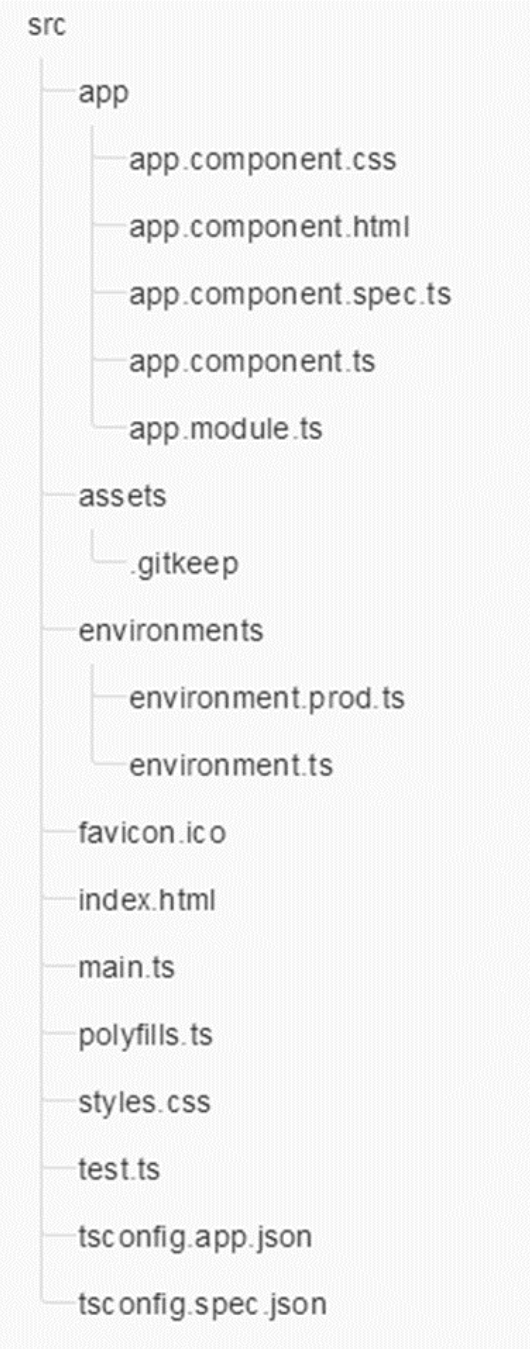
  CLI створив для вас перший компонент.
  Це кореневий компонент *app-root*.
  Ви можете знайти його в ./src/app/app.component.ts
  Відкрийте файл компонента і змініть властивість *title*:

```
export class AppComponent {
        title = ‘ My First Angular App! ';
}
```

  Браузер автоматично перезавантажується зі зміненим заголовком.

Огляд файлів проекту

- **app/app.component.{ts,html,css, spec.ts}** – визначає AppComponent разом з html-шаблоном, стилями і юніт-тестами. Це кореневий компонент, для якого в міру розвитку додатка з'явиться дерево вкладених компонентів.
- **app/app.module.ts** – визначає AppModule. Кореневий модуль, який повідомляє Angular, як буде зібрано додаток. Зараз в ньому оголошений тільки AppComponent. Згодом ви будете оголошувати в ньому інші компоненти.



- **assets/** * - директорія, в якій ви розміщуєте зображення і все інше
- **environments/** * - ця директорія містить файли цілей збірки (dev або prod режими)
- **favicon.ico** - ви можете додати свою власну іконку, яка буде відображатися на вкладці в браузері
- **index.html** - основна HTML-сторінка, яка відображається, коли хтось відвідує ваш сайт. У більшості випадків вам ніколи не знадобиться її редагувати. Angular CLI автоматично додає всі згенеровані js і css-файли.
- **main.ts** - точка входу вашого застосунку. Зараз, за замовчуванням, ваш додаток компілюється в постачанні з JIT-компілятором. Даний файл завантажує кореневий модуль програми (AppModule) і запускає його в браузері.

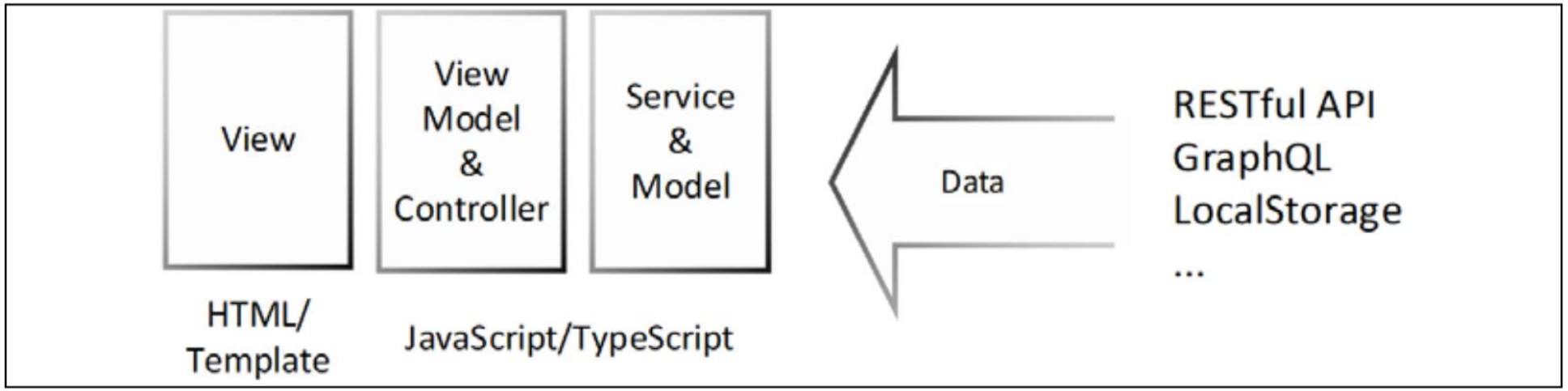
- **polyfills.ts** - різні браузери мають різні рівні підтримки тих чи інших веб-стандартів. Поліфілли допомагають нормалізувати ці відмінності.
- **styles.css** - тут зберігаються глобальні стилі. Більшу частину часу ви будете працювати з локальними стилями своїх компонентів.
- **test.ts** - це точка входу всіх ваших юніт-тестів.
- **tsconfig. {app | spec} .json** - конфігурація компілятора TypeScript описується в файлі tsconfig.app.json, для юніт-тестів же використовується конфігурація tsconfig.spec.json
- **src/** є тільки одним з елементів кореневої директорії проекту. Інші файли допомагають збирати, тестувати, підтримувати, документувати і розгортати додаток.

- **e2e/** - всередині директорії e2e/ розташовуються e2e (end-to-end) тести.
- **node_modules/** - Node.js створює дану директорію, в якій зберігає всі сторонні модулі, перераховані в package.json
- **angular.json** - конфігураційний файл Angular.
- **.editorconfig** - Просте налаштування для вашого редактора
- **karma.conf.js** - конфігураційний файл для запуску юніт-тестів з використанням Karma, запуск тестів можна виконати командою ng test.
- **package.json** - конфігураційний файл npm, в ньому перераховуються сторонні модулі (пакети) розробників, які використовує ваш проект. Тут ви також можете прописати і свої власні скрипти.
- **tsconfig.json** - конфігурація компілятора TypeScript для вашої IDE

```
|-- README.md
|-- e2e
|-- karma.conf.js
|-- node_modules
|-- package-lock.json
|-- package.json
|-- protractor.conf.js
|-- src
|-- tsconfig.json
`-- tslint.json
```

Basic Angular architecture (Angular for Enterprise-Ready Web Applications. — Packt Publishing, 2020, p.17)

Angular слідує за шаблоном MV *, який є гібридом шаблонів MVC та MVVM. Раніше ми розглядали шаблон MVC. На високому рівні архітектура обох зразків є відносно схожою, як показано на діаграмі, яка наведена нижче:

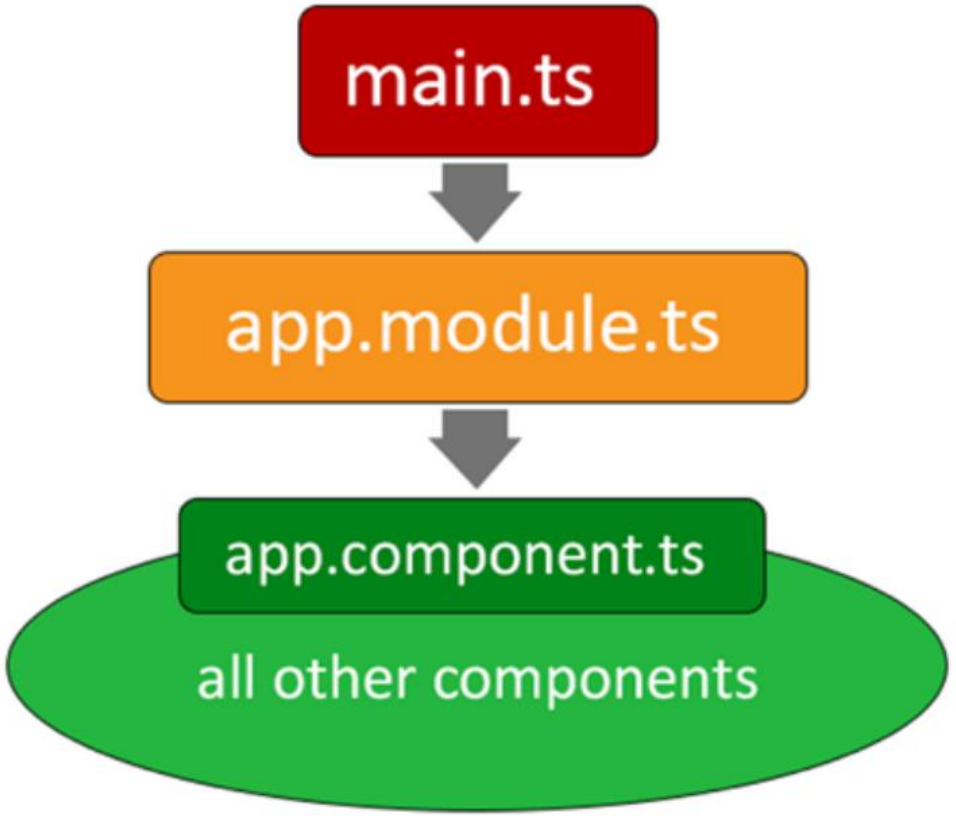


- Нова концепція тут - ViewModel, яка представляє код “клею”, який пов’язує ваше View з вашою моделлю або послугою.
- У Angular цей “клей” відомий як зв’язування (binding).
- Тоді як MVC-фреймворки, такі як Backbone або React, повинні викликати метод render для обробки своїх HTML-шаблонів, в Angular цей процес є простим і прозорим для розробника.
- Прив'язка - це те, що відрізняє програму MVC від програми MVVM.

Як Angular виконує базовий додаток

- Angular’у потрібен принаймні один компонент та один модуль.
- Додаток Angular вимагає root module - умовно званого AppModule, який повідомляє Angular, як збирати програму.
- Кореневий модуль також містить список усіх доступних компонентів.
- Далі наведена схема стандартного циклу Angular Initialization Cycle, яка допоможе нам краще уявити, як це працює:

Angular Initialization Cycle



Модулі

- Модуль Angular є контейнер для групи пов'язаних компонентів, сервісів, директив і т. д.
- Модуль можна вважати бібліотекою компонентів і сервісів, що реалізує певну функціональність, наприклад, модуль відправки товару або модуль формування рахунків.
- Всі елементи невеликого додатки можуть знаходитися в одному модулі (кореневому), а більші додатки - мати більше одного модуля.
- Всі додатки повинні мати хоча б один кореневий модуль, який ініціалізується під час запуску програми.
- З точки зору синтаксису модуль - це клас, анотований декоратором NgModule, який може містити інші ресурси.

App module

Listing 2.2 App module (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 { }
```

Providers are any services used in the app.

Imports Angular dependencies needed

Imports the App component

Uses the NgModule annotation to define a module by passing an object

Declarations are to list any components and directives used in the app.

Imports are other modules that are used in the app.

Bootstrap declares which component to use as the first to bootstrap the application.

Exports an empty class, which gets annotated with configuration from NgModule

App component

Listing 2.1 Generated App component (src/app/app.component.ts)

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app works!';
}
```

Import the component annotation

Define the component and its properties

Create the component controller, with a single property

- Основною одиницею програми Angular є компонент.
- Компонент - це комбінація класу JavaScript, написаного на TypeScript, та Angular шаблону, написаного на HTML, CSS та TypeScript.
- Клас і шаблон поєднуються через bindings, щоб вони могли спілкуватися між собою, як показано на схемі, яка наведена нижче:

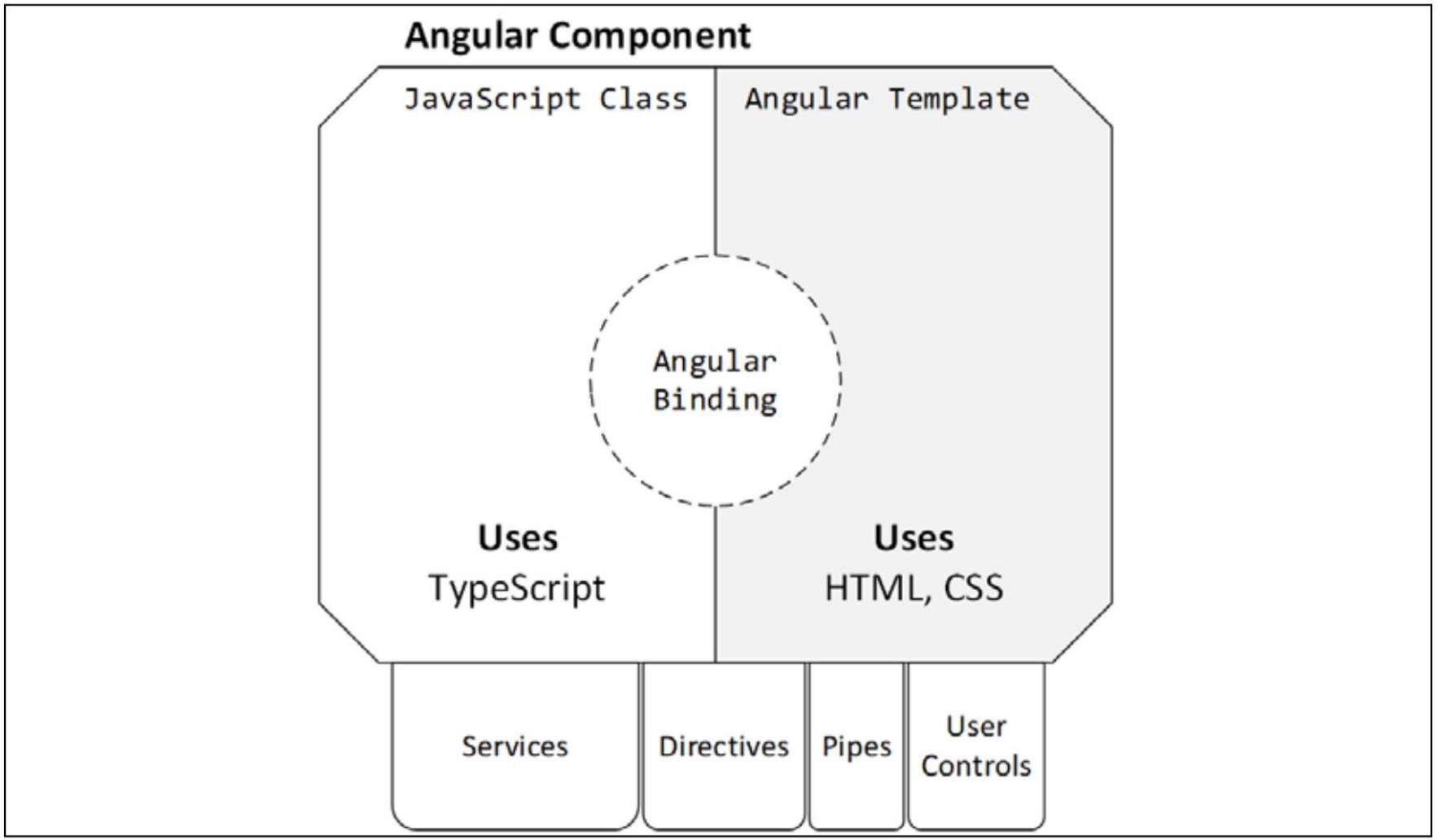
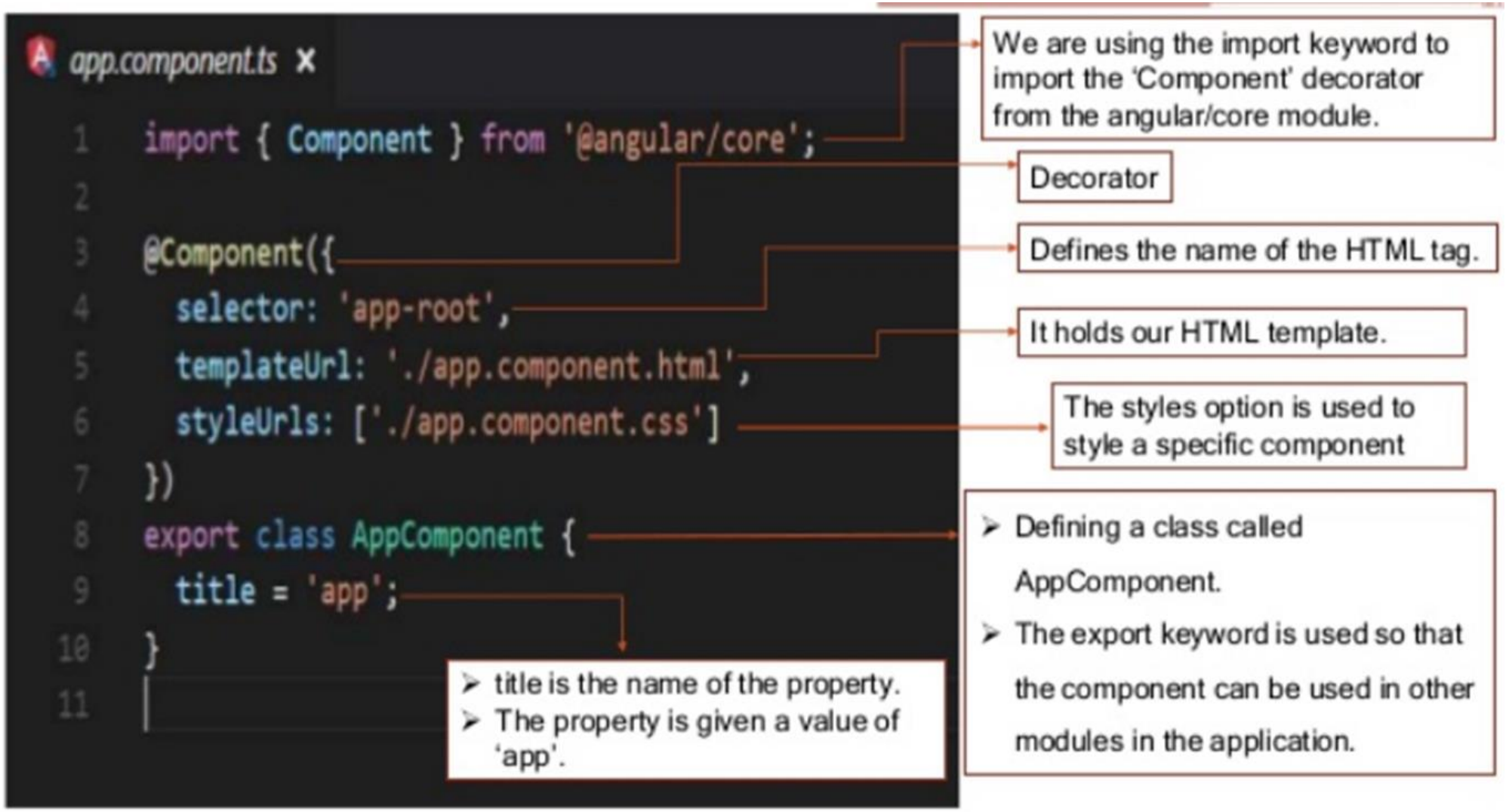


Рис. Анатомія компонента

Приклад компонента



Завантаження додатка (Bootstrapping the app)

Listing 2.3 The main file that is called on launch (src/app/main.ts)

Imports dependencies

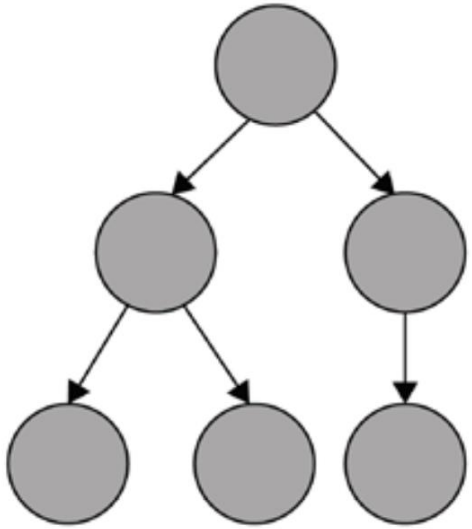
```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { environment } from './environments/environment';
import { AppModule } from './app/';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule);
```

If production is enabled, turn off Angular developer mode.

Bootstraps the App module

Створення дочірнього компонента

- Компоненти є основними будівельними блоками програми Angular. Вони контролюють різні частини веб-сторінки, які називаються переглядами (views), наприклад, список продуктів або реєстраційну форму.
- Додаток Angular складається з дерева компонентів, які можуть взаємодіяти між собою:



- Однією з найбільш часто використовуваних команд Angular CLI є команда generate , яку ми використовуємо для створення певних Angular артефактів.

ng generate <type> <name>

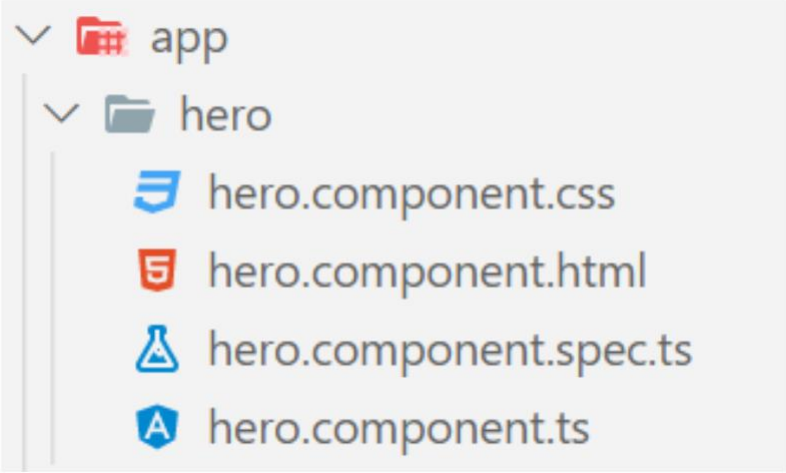
- Щоб створити компонент, перейдіть до кореневої папки проекту Angular CLI та запустіть у командному рядку наступне:

ng generate component hero

- Якщо ви використовуєте VS Code, розгляньте можливість використання другого вікна терміналу для запуску команд Angular CLI. Виберіть **Terminal | New Terminal** з головного меню, щоб відкрити його.
- Створення компонента Angular - це двоетапний процес. Він включає створення необхідних файлів компонента та його реєстрацію за допомогою Angular module.

Файли нового компонента

- Коли ми запускаємо команду **ng generate component hero**, Angular CLI створює папку **hero** всередині папки **app** та генерує такі файли:



Реєстрація модуля (Файл **app.module.ts**)

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

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

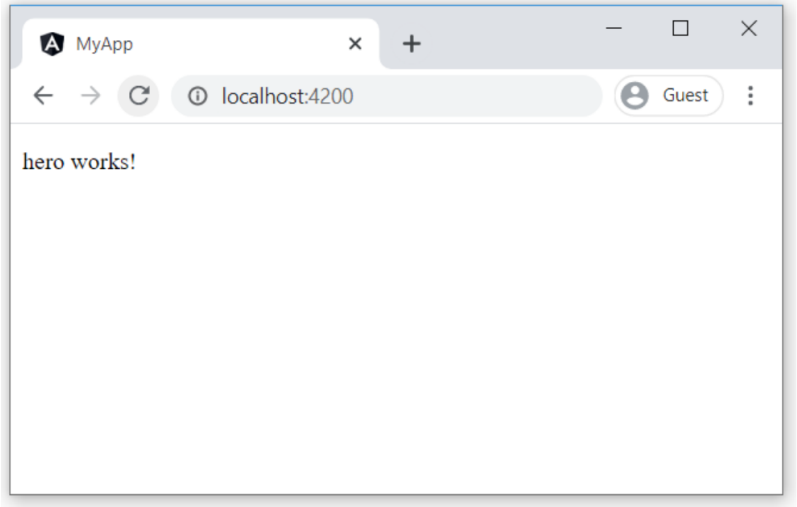
Взаємодія з іншими компонентами

- Компоненти Angular мають public API, що дозволяє їм взаємодіяти з іншими компонентами.
- Цей API охоплює **input properties**, які ми використовуємо для передачі компоненту даних.
- Він також надає **output properties**, до яких ми можемо прив’язати прослуховувачі подій (**event listeners**), тим самим отримуючи своєчасну інформацію про зміни стану компонента.
- Давайте розглянемо на простих прикладах, як Angular вирішує проблеми введення даних в компонент та отримання даних із компонентів.

Передача даних за допомогою прив'язки вводу

Раніше ми створили новий компонент **hero**, тепер розглянемо як він працює.

1. Перейдіть до папки **app**, яка знаходиться всередині папки **src**.
2. Відкрийте шаблон основного компонента програми, **app.component.html**.
3. Замініть вміст шаблону на селектор компонента hero, **<app-hero> </app-hero>**.
4. Якщо ми запустимо додаток, ми побачимо на екрані шаблон нашого нового компонента:



- Шаблони, що відображають лише статичну інформацію, рідко зустрічаються у програмі Angular.
- Давайте зробимо наш компонент **hero** більш інтерактивним, показуючи ім’я фактичного героя, який працює.

5. Ім'я буде динамічно передаватися з AppComponent. Спочатку ми визначаємо властивість у класі компонента hero за допомогою декоратора @Input, а потім імені властивості:

   @Input() name: string;

- Декоратор @Input - це спеціалізований декоратор TypeScript, створений командою Angular, який використовується, коли ми хочемо передати дані з компонента в інший компонент.

6. Спочатку нам потрібно імпортувати декоратор з пакета @angular/core, щоб використовувати його:

   import { Input } from '@angular/core';

7. После того, як ми визначили вхідну властивість name, ми використовуємо прив'язку властивості name до шаблону компонента hero:

   ** {{name}} hero works! **

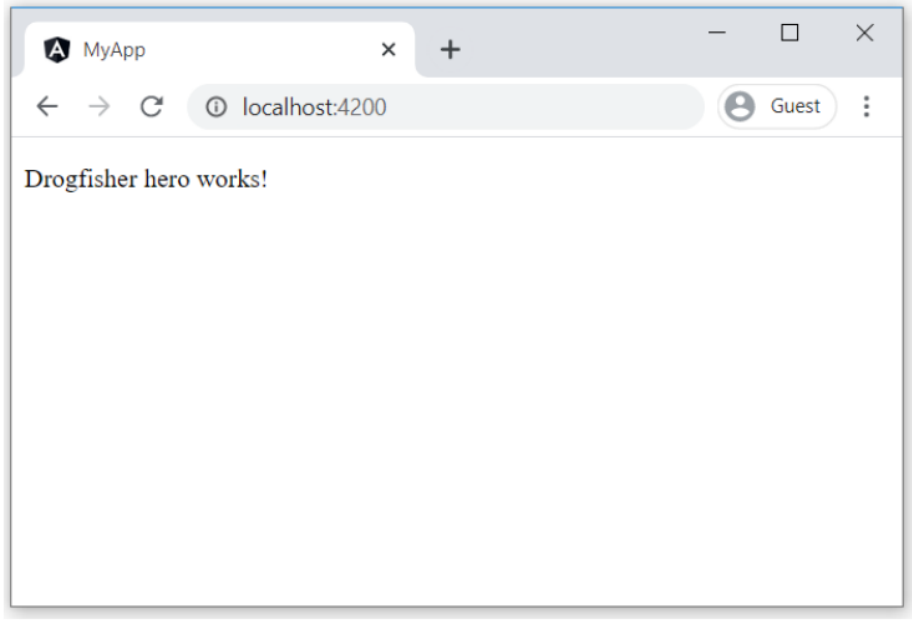
8. Більшу частину роботи ми вже виконали; тепер нам потрібно передати значення властивості name з AppComponent.

   <app-hero [name]="hero"></app-hero>

9. Змінна hero, яку ми використовуємо у вхідному прив'язуванні, відповідає властивості в класі AppComponent:

```
export class AppComponent {
  title = 'my-app';
  hero = 'Drogfisher';
}
```

Після збереження змін ми побачимо:



Прослуховування подій за допомогою output binding

- Ми дізналися, що введення прив'язки використовується, коли ми хочемо передавати дані між компонентами.
- Цей метод застосовний у сценаріях, коли ми маємо два компоненти, один виконує роль батьківського компонента, а інший - дочірнього.
- Що робити, якщо ми хочемо спілкуватися навпаки, від дочірнього компонента до батьківського?
- Як ми повідомляємо батьківський компонент про конкретні дії, які відбуваються з дочірнім компонентом?
- Розглянемо сценарій, коли шаблон компонента героя містить елемент кнопки Like, який при натисканні повинен повідомляти AppComponent про дії користувача.

1. Спочатку ми визначаємо вихідну властивість у класі компонента hero:

   @Output() liked = new EventEmitter();

- Властивість **liked**, - це EventEmitter, позначений декоратором @Output. Декоратор @Output - це спеціалізований декоратор TypeScript, який використовується, коли ми хочемо запускати події від компонента до іншого компонента.

2. Спочатку нам потрібно імпортувати їх обох із пакета @angular/core, щоб використовувати їх:

   import { Output, EventEmitter } from '@angular/core';

3. Наша кнопка повинна викликати метод emit властивості **liked**, щоб запустити EventEmitter:

   <button (click)="liked.emit()">Like</button>

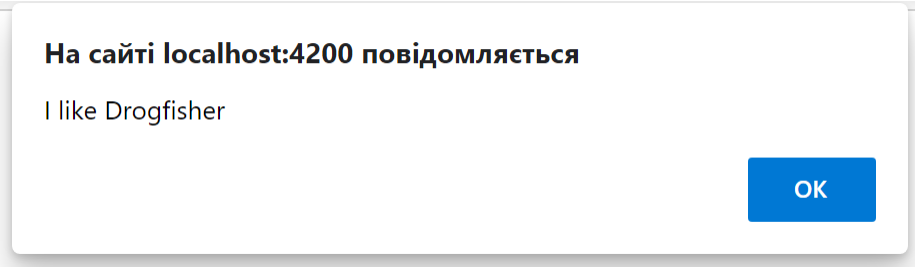
4. Нам потрібно зробити прив’язку в AppComponent, щоб ці два компоненти могли взаємодіяти між собою. Ми використовуємо прив'язку подій, щоб прив'язати метод onLike від AppComponent до вихідної властивості **liked** компонента **hero**. Цей підхід називається вихідним прив'язуванням:

   <app-hero [name]="hero" **(liked)="onLike()** “ ></app-hero>

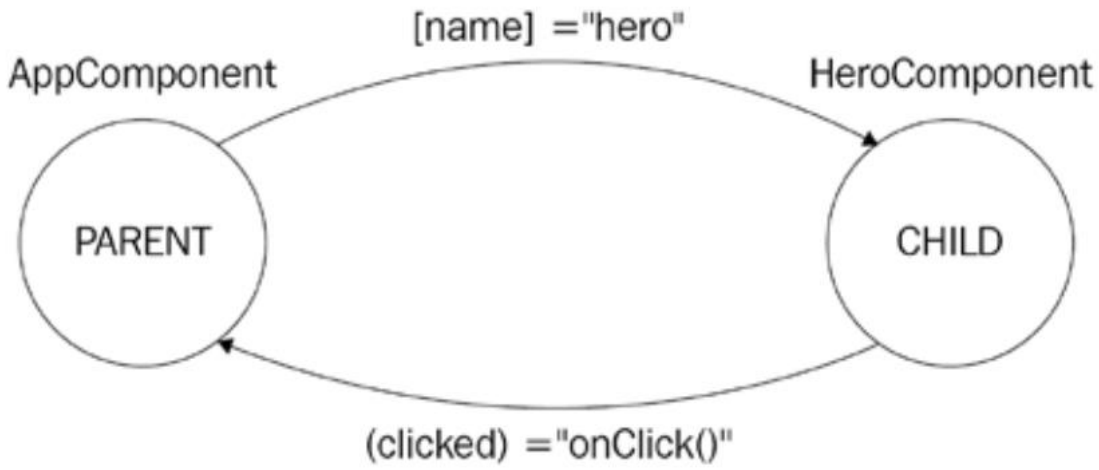
5. Коли користувач натискає кнопку Like у компоненті **hero**, AppComponent викликає метод onLine:

```
onLike() {
   window.alert(`I like ${this.hero}`);
}
```

- Після збереження змін ми побачимо: (при натисканні кнопки Like)



- Тут ви можете побачити огляд механізму зв'язку компонентів, який ми вже обговорювали:



- Клас EventEmitter також може бути використаний для передачі довільних даних через метод emit, про що ми дізнаємось у наступному підрозділі.

Передача даних через події користувача (custom events)

- Припустимо, що програма повинна зберігати стан кнопки Like кожного разу, коли її натискали, щоб герой міг подобатись або не подобатись користувачу.
- Оголосимо тип даних, які будуть передані в AppComponent:

```
@Output() liked = new EventEmitter<boolean>();
```

- Прив'язка події click шаблону компонента hero викликає метод emit, що передає логічне значення наступним чином:

```
<button (click)="liked.emit(true)">Like</button>
```

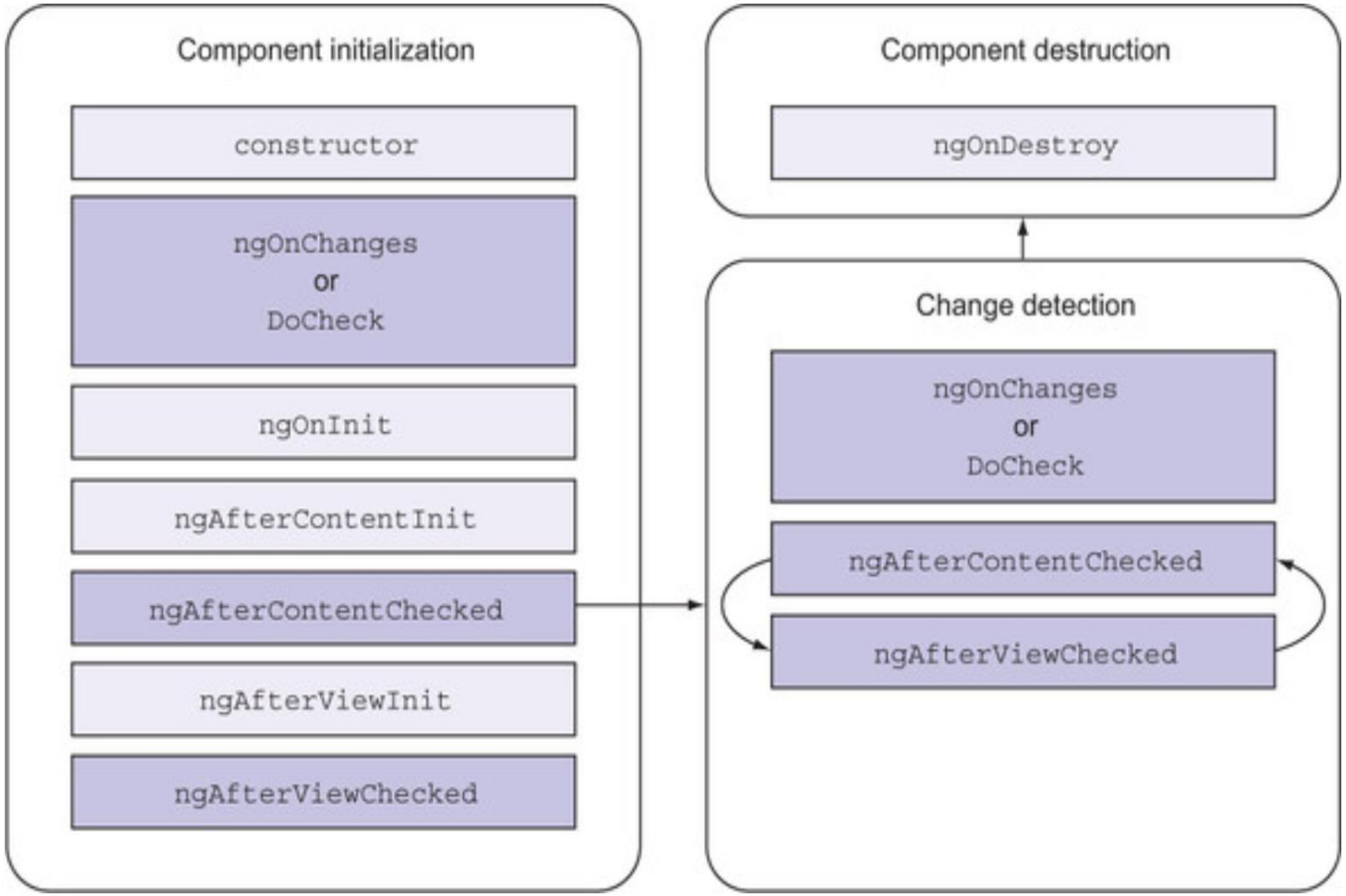
- Далі дані будуть доступні в AppComponent через об’єкт $event.

```
<app-hero [name]="hero" (liked)="onLike($event)"></app-hero>
```

Для коректної роботи застосунку ще треба змінити метод onLike (завдання для самостійної роботи)

Життєвий цикл компонента (component lifecycle)

- За час життєвого циклу з компонентом Angular трапляються різні події.
- Після створення елемента механізм визначення змін починає за ним спостерігати.
- Компонент ініціалізується, додається в модель DOM і промальовується, щоб користувач міг побачити його.
- Потім стан елемента (значення його властивостей) може змінитися, що викличе перерисовку інтерфейсу; і нарешті, компонент знищується.
- Події життєвого циклу - це зачепи, прив’язки (hooks), які дозволяють нам підглядати за певними етапами життєвого циклу компонента та застосовувати власну логіку коли це потрібно.



- На рисунку показані прив'язки життєвого циклу (функції зворотного виклику), де ви додаєте призначений для користувача код, якщо потрібно. Функції зворотного виклику, показані на світло-сірому фоні, будуть викликані всього один раз, а функції на темно-сірому фоні - кілька разів.

Покращення компонентів за допомогою каналів (pipes) та директив (directives)

- У попередньому підрозділі ми побудували кілька компонентів, які відображали дані на екрані за допомогою властивостей введення та виведення.
- Зараз ми покращимо наші компоненти, використовуючи директиви та канали.
- Канали (фільтри) - класи, метою яких є перетворення даних перед їх отриманням компонентом.
- Директиви дозволяють нам виконувати більш амбітні функції, такі як маніпулювання DOM або зміна зовнішнього вигляду та поведінки елементів HTML.

Вступ до директив

- Директиви Angular - це атрибути HTML, які розширюють поведінку або вигляд стандартного елемента HTML.
- Коли ми застосовуємо директиву до елемента HTML або навіть компонента Angular, ми можемо додати до нього власну поведінку або змінити його вигляд.

Існує три типи директив:

- Компоненти (**Components**) - це директиви з асоційованим шаблоном.
- Структурні директиви (**Structural directives**) додають або видаляють елементи з DOM.
- Директиви атрибутів (**Attribute directives**) змінюють зовнішній вигляд або визначають власну поведінку елемента DOM.

Angular надає нам набір вбудованих директив, які ми можемо використовувати в наших компонентах та охоплюють більшість випадків використання.

Перетворення елементів за допомогою директив

Фреймворк Angular включає набір готових структурних директив, які ми можемо почати використовувати відразу в наших програмах:

- **ngIf** додає або видаляє частину дерева DOM на основі виразу.
- **ngFor** переглядає список елементів і прив'язує кожен елемент до шаблону.
- **ngSwitch** перемикається між шаблонами в межах певного набору і відображає кожен із них залежно від умови.

Ми опишемо кожну із них у наступних підрозділах.

Відображення даних умовно

- Директива ngIf додає або видаляє елемент HTML у DOM на основі оцінки виразу.
- Якщо вираз має значення true, елемент вставляється в DOM. В іншому випадку елемент видаляється з DOM.
- Ми могли б вдосконалити наш компонент *hero* з попереднього підрозділу, використовуючи цю директиву:

```
<p *ngIf="name === 'Boothstomper'">{{name}} hero works!</p>
```

- Коли властивість name класу компонента має значення Boothstomper, елемент абзацу відображається на екрані. В іншому випадку він повністю видаляється.

Перебір даних (Iterating through data)

- Директива **ngFor** дозволяє нам перебирати колекцію елементів і рендерити шаблон для кожного з них.
- Ми можемо розглядати ngFor як цикл for для шаблонів HTML.
- Припустимо, що ми маємо масив об’єктів Hero, які ми хочемо відобразити. Ми можемо зробити це за допомогою ngFor:

```
<ul>
  <li *ngFor="let hero of heroes">
    {{hero.name}}
  </li>
</ul>
```