Лекція 4. Angular: основні поняття. Односторінкові та багатосторінкові додатки. Історія створення фреймворку. Інструментарій Angular-розробника. Створення Angular-додатку. Angular.json, package.json

Що таке Angular. Початок роботи з фреймворком¶

Angular представляє фреймворк від компанії Google для створення клієнтських програм. Насамперед він орієнтований на розробку SPA-рішень (Single Page Application), тобто односторінкових додатків. У цьому плані Angular є спадкоємцем іншого фреймворку AngularJS. У той же час, Angular це не нова версія AngularJS, а принципово новий фреймворк.

Angular надає таку функціональність як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому, шаблони, маршрутизація і так далі.

Однією з ключових особливостей Angular є те, що він використовує мову програмування TypeScript.

Але Angular не обмежений мовою TypeScript. За бажанням можна писати програми на Angular за допомогою таких мов як Dart або JavaScript. Однак TypeScript є основною мовою для Angular.

Офіційний репозиторій фреймворку на гітхабі: https://github.com/angular/angular. Там можна знайти самі вихідні файли, а також деяку додаткову інформацію.

Як платформа, Angular включає:

- Заснований на компонентах фреймворк для створення масштабованих веб-застосунків.
- Набір добре інтегрованих бібліотек, що охоплюють широкий спектр функцій: маршрутизація, керування формами, клієнт-серверна взаємодія тощо.
- Набір інструментів розробника, які допомогають розробляти, збирати, тестувати та оновлювати код.

Коли створюється програма за допомогою Angular, то використовуються переваги платформи, яка може масштабуватися від проекту, який розроблює одна людина, до програм корпоративного рівня. Angular розроблено, щоб максимально спростити оновлення, тому можна використовувати останні розробки з мінімумом зусиль. При цьому, що екосистема Angular складається з величезної спільноти, що включає більш ніж 1.7 мільйона розробників, авторів бібліотек і творців контенту.

Односторінковий додаток проти багатосторінкового додатку

Отже, всі web-додатки поділяються на односторінкові (SPA) та багатосторінкові (MPA). SPA, Single Page Application, або «додаток однієї сторінки» – це тип веб-застосунків, в яких завантаження необхідного коду відбувається на одну сторінку. Це дозволяє заощадити час на повторне завантаження тих самих елементів.

Багатосторінковий додаток - це традиційний веб-додаток, в якому з сервера запитується нова сторінка для відображення щоразу, коли відбувається обмін даними з сервером. Обсяг контенту, який несуть такі додатки, величезний, тому вони, як правило, багаторівневі, зі значною кількістю посилань і складними інтерфейсами користувача.

Програма SPA - це буквально одна сторінка, яка постійно взаємодіє з користувачем, динамічно переписуючи поточну сторінку, а не завантажуючи нові сторінки з сервера. Приклади односторінкових програм: Trello, Facebook, Gmail, Twitter - ось кілька прикладів SPA.

Особливість SPA архітектури полягає в тому, що всі елементи, необхідні для роботи софту знаходяться на одній сторінці. Вони завантажуються під час ініціалізації. Також цей вид програм завантажує додаткові модулі після запиту від користувача. Будь-яка активність користувача фіксується для зручності навігації. Це дозволяє скопіювати посилання та відкрити софт на тому ж етапі взаємодії на іншій вкладці, браузері або пристрої.

При завантаженні нових модулів у SPA контент на них оновлюється лише частково, тому що немає необхідності повторно завантажувати постійні елементи. Це збільшує швидкість відповіді і скорочує обсяг даних, що передається між браузером і сервером.

Даний вид софту за способом взаємодії з користувачем найбільше нагадує роботу десктопних додатків, але з серверу.

Переваги Single Page Applications

- **Доступність**. Можна отримати негайний доступ до функціоналу з будь-якого типу пристрою без проблем із сумісністю, обсягом пам'яті, потужностями або часом на установку.
- **Універсальність**. Використовувати програму можна практично з будь-якого пристрою, якщо на ньому є доступ до інтернету. Якщо при розробці інтерфейсу враховувалися різні роздільні здатності екрану, то використовувати SPA однаково зручно і з ПК, і зі смартфона.
- **Можливість задіяти великі обсяги даних**. Розмір програми та даних, що використовуються, не обмежений пам'яттю пристрою.

- **Швидкість**. Одна сторінка з усім необхідним не тільки заощаджує час на повторне завантаження даних, а й підвищує продуктивність роботи.
- **Можливості розробки**. Розробникам доступні фреймворки, які спрощують створення архітектури проекту та надають чимало готових елементів для роботи.

Недоліки SPA

- **Необхідність інтернет-з'єднання**. Без доступу до мережі використовувати цей софт неможливо. Але навіть десктопне ПЗ використовує у роботі зовнішні бази даних, то доступ до Інтернету необхідний у будь-якому випадку.
- **Проблеми з SEO**. Особливості SPA ускладнюють або унеможливлюють процес індексації пошуковими системами всіх модулів програми. Це може спричинити труднощі з оптимізацією і з поширенням.
- **Не працює у користувачів з відключеною підтримкою JS**. Багато хто відключає відображення JS-елементів у себе в браузерах, а Single Page Application використовує їх у роботі, тому може не працювати.

Перерахуємо найпопулярніші фреймворки односторінкових додатків, які можуть обробляти великі архітектури, необхідні для повнофункціональних веб-додатків.

- Angular.js
- React.js
- Vue.js
- Backbone.js

Історія створення

Перероблений AngularJS отримав назву «Angular 2», але це призвело до плутанини серед розробників. Щоб уточнити, команда оголосила, що для кожного фреймворку слід використовувати окремі назви: для "AngularJS" - з посиланням на версії 1.X; для "Angular" без "JS" - версії 2 і вище.

Версія 2

Про Angular 2.0 було анонсовано на конференції ng-Europe 22–23 жовтня 2014 року. Кардинальні зміни у версії 2.0 викликали значні суперечки серед розробників. 30 квітня 2015 року розробники Angular оголосили, що Angular 2 перейшов з Alpha на Developer Preview. Angular 2 перейшов на бета-версію в грудні 2015 року, і перший реліз був опублікований у травні 2016 року. RxJS включили в Angular 2. Остаточна версія була випущена 14 вересня 2016 року.

Версія 4

13 грудня 2016 року було анонсовано Angular 4, пропустивши 3, щоб уникнути плутанини через невідповідність версії пакета маршрутизатора, яка вже була розповсюджена як v3.3.0. Остаточна версія була випущена 23 березня 2017 року. Angular 4 є зворотно сумісний з Angular 2.

Angular версія 4.3 є другорядним випуском, що означає, що він не містить критичних змін і є заміною для 4.xx

Особливості у версії 4.3

- Представлено HttpClient, меншу, простішу у використанні та потужнішу бібліотеку для створення HTTP-запитів.
- Нові події життєвого циклу маршрутизатора для Guards і Resolvers.
- Умовно вимкнуто анімацію.

Версія 5

Angular 5 був випущений 1 листопада 2017 року. Ключові вдосконалення в Angular 5 включають підтримку для прогресивних веб-програм, оптимізатор збірки та вдосконалення, пов’язані з Material Design.

Версія 6

Angular 6 було випущено 4 травня 2018 року. Це важливий випуск, зосереджений не на базовому фреймворку, а більше на ланцюжку інструментів і полегшенні швидкого оновлення Angular у майбутньому. Версія включає нові команди, наприклад: ng update, ng add. Також нові елементи, такі як, Angular Elements, Angular Material + CDK Components, Angular Material Starter Components, CLI Workspaces, підтримка бібліотек, покращення продуктивності анімації та RxJS v6.

Версія 7

Angular 7 було випущено 18 жовтня 2018 року. Оновлення торкнулись продуктивності програми, Angular Material & CDK, віртуального прокручування, а також оновлення залежностей щодо Typescript 3.1, RxJS 6.3, Node 10 (все ще підтримує Node 8).

Версія 8

Angular 8 був випущений 28 травня 2019 року. Він містить диференційне завантаження для всього коду програми, динамічний імпорт для відкладених маршрутів, Web workers, підтримку TypeScript 3.4 і Angular Ivy. Angular Ivy включає:

- Згенерований код, який легше читати та налагоджувати під час виконання
- Швидший час відновлення
- Покращений розмір корисного навантаження
- Покращена перевірка типу шаблону

- Зворотна сумісність

Версія 9

Angular 9 було випущено 6 лютого 2020 року. Версія 9 переводить усі програми на використання компілятора Ivy і середовища виконання за замовчуванням. Angular оновлено для роботи з TypeScript 3.6 і 3.7. Окрім сотень виправлень помилок, компілятор і середовище виконання Ivy пропонують численні переваги:

- Швидше тестування
- Краще налагодження
- Покращено зв'язування класів і стилів CSS
- Покращена перевірка типу
- Покращено відстежування помилок
- Покращено час збірки, увімкнувши AOT за замовчуванням
- Покращена Інтернаціоналізація

Версія 10

Angular 10 було випущено 24 червня 2020 року.

- Новий засіб вибору діапазону дат
- Додаткові суворіші налаштування
- Нова конфігурація браузера за замовчуванням
- Скасування та видалення

Версія 11

Angular 11 було випущено 11 листопада 2020 року.

Версія 12

Angular 12 було випущено 12 травня 2021 року.

- Застаріла підтримка IE11

Версія 13

Angular 13 було випущено 4 листопада 2021 року

Версія 14

Angular 14 було випущено 2 червня 2022 року. Деякі нові функції включають типізовані форми, автономні компоненти та нові примітиви в Angular CDK (набір для розробки компонентів). Окремі компоненти працюють у Angular, і тепер вони повністю працюють у HttpClient, Angular Elements, маршрутизаторі тощо.

Версія 15

Angular 15 вийшов 16 листопада 2022 року.

Версія 16

Angular 16 було випущено 3 травня 2023 року. Серед нових функцій — рендеринг на стороні сервера, підтримка автономного проекту тощо.

Графік випуску та підтримки Angular

Майбутні випуски

Починаючи з версії 9, команда Angular перевела всі нові програми на використання компілятора та середовища виконання Ivy. Команда працюватиме над Ivy, щоб покращити розмір вихідних пакетів і швидкість розробки.

Очікується, що кожна версія буде зворотно сумісною з попередньою версією. Команда розробників Angular пообіцяла робити оновлення двічі на рік.

Angular и RxJS

Angular з версії 2.0 тісно співпрацює з бібліотекою RxJS.

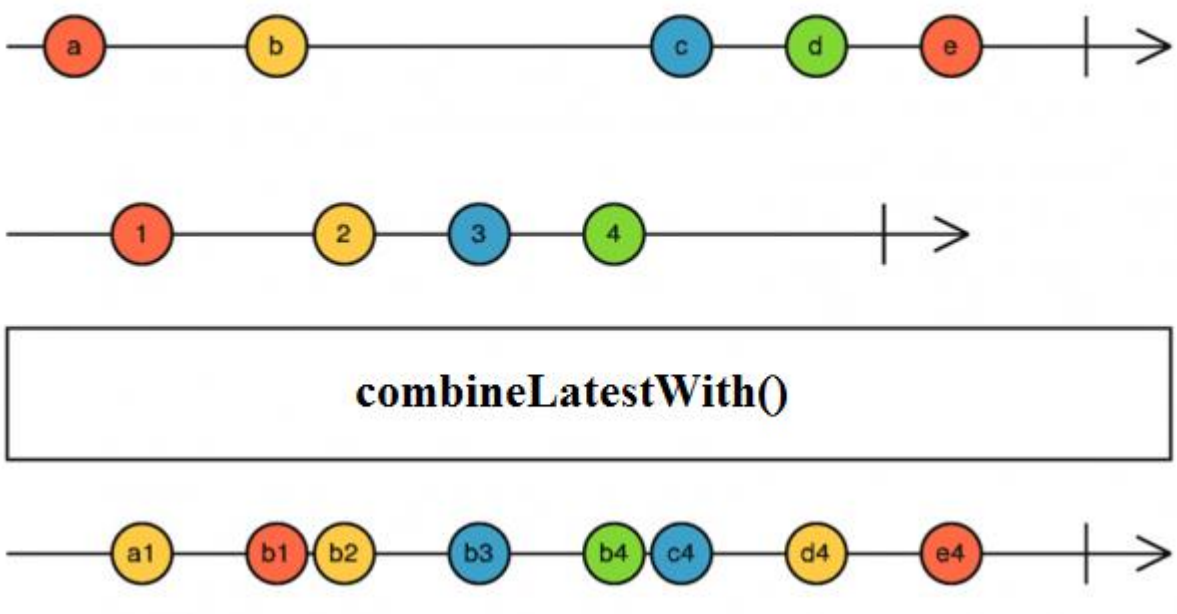
RxJS (Reactive Extensions for JavaScript) — це бібліотека для реактивного програмування з використанням значень, що спостерігаються, що полегшує створення асинхронного коду або коду, заснованого на зворотних викликах.

RxJS надає реалізацію типу Observable, яка необхідна доти, доки цей тип не стане частиною мови і поки браузери не підтримуватимуть його. Бібліотека також надає службові функції для створення та роботи зі значеннями, що спостерігаються. Ці утиліти можуть бути використані для:

- Перетворення існуючого коду для асинхронних операцій на спостережувані значення;
- Ітерації значень у потоці;
- Зіставлення значень з різними типами;

- Фільтрації потоків
- Компонування кількох потоків

Наприклад, при допомозі оператора combineLatestWith() можна об’єднати декілька потоків в один. Після того, як кожен потік зробить хоча б один еміт, ми отримуємо останні значення від кожного у вигляді масиву. Далі, після будь-якого еміту з об'єднуваних потоків він буде віддавати нові значення. При вивченні операторів буде у нагоді проект rxmarbles.com — його автор (André Staltz). Він створив велику кількість динамічних погодинних діаграм, по яким можно прослідкувати, як саме впливає порядок даних на підсумковий потік. Наприклад, оператор combineLatestWith() працює наступним чином:



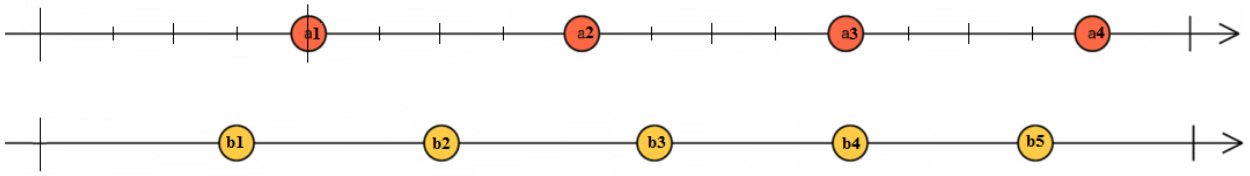
Діаграма Marbles для оператора combineLatestWith()

Приклад роботи оператора combineLatestWith()

e:\Project(RxJS)\Project6\index.js

```
const a1$ = rxjs.Observable.create((observer) => {
  let i = 1;
  // Емітим значення раз в 1000мс
  setInterval(() => {
    observer.next('a: ' + i++);
  }, 1000);
 });
```

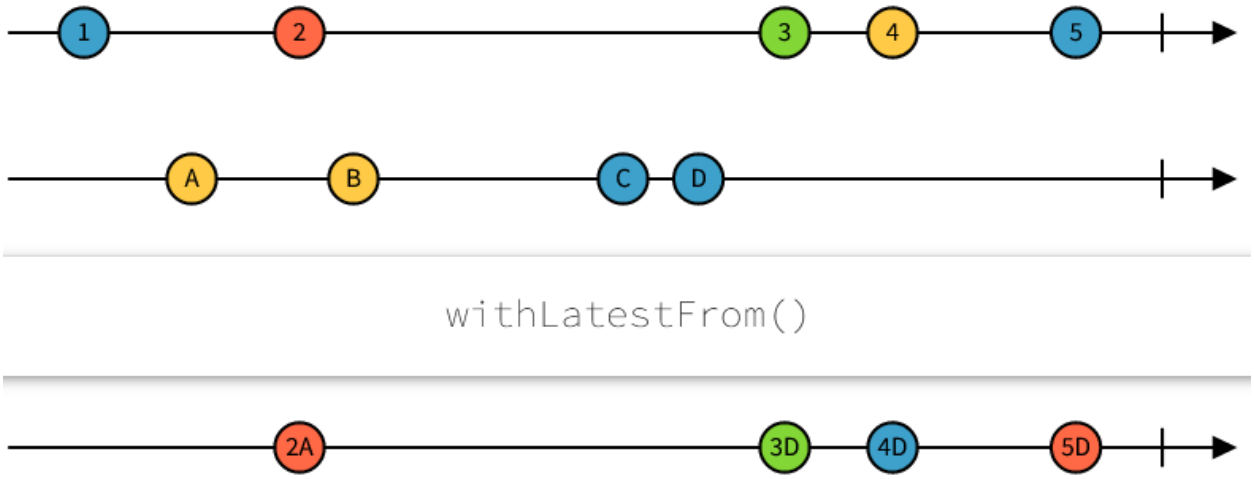
```
const b1$ = rxjs.Observable.create((observer) => {
  let i = 1;
  // Емітим значення раз в 750мс
  setInterval(() => {
    observer.next('b: ' + i++);
  }, 750);
});

a1
.pipe(
combineLatestWith(b1),
take(8))
.subscribe(val => console.log(val));
```



Результат:

```
▸ Array [ "a: 1", "b: 1" ]
▸ Array [ "a: 1", "b: 2" ]
▸ Array [ "a: 2", "b: 2" ]
▸ Array [ "a: 2", "b: 3" ]
▸ Array [ "a: 3", "b: 3" ]
▸ Array [ "a: 3", "b: 4" ]
▸ Array [ "a: 3", "b: 5" ]
▸ Array [ "a: 4", "b: 5" ]
```

Актуальні на даний час оператори https://rxjs-dev.firebaseapp.com/api/operators



Діаграма Marbles для оператора withLatestFrom()

Перш ніж розпочати вивчення платформи Angular, спочатку варто познайомитися з Angular CLI. Angular CLI - це швидкий, простий та рекомендований спосіб розробки Angular-додатків. Angular CLI полегшує виконання низки завдань. Ось деякі з них:

ng build	**Компілює Angular-додаток у вихідний каталог.**
ng serve	Збирає та запускає вашу програму, перезбираючи її при зміні файлів.
ng generate	Генерує або змінює файли на основі схематиків
ng test	Запуск модульних тестів для заданого проекту.
ng e2e	Збирає і запускає Angular-додаток, запускаючи потім наскрізні тести.

Необхідні умови

Щоб встановити Angular у вашій локальній системі, вам знадобиться таке:

- **Node.js** Angular вимагає поточну, останню LTS або підтримувану LTS версію Node.js. Для отримання додаткової інформації про встановлення Node.js дивіться nodejs.org. Якщо ви не знаєте, яка версія Node.js встановлена у вашій системі, запустіть node –v у терміналі.
- **пакетний менеджер npm** Angular, Angular CLI, Angular програми залежать від npm пакетів, які забезпечують багато функцій. Для завантаження та встановлення npm пакетів вам потрібен пакетний менеджер npm. npm встановлюється разом з Node.js за замовчуванням. Щоб дізнатися яка версія npm встановлена, запустіть npm –v у терміналі.
- **Angular CLI-це інтерфейс командного рядка для роботи з Angular.** Глобально у системі Angular CLI встановлюється при допомозі команди @angular/cli:

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

@angular/cli - це інтерфейс командного рядка для роботи з Angular, його використання значно полегшує процес розробки. Наприклад, однією командою можна згенерувати повністю налаштовану працюючу програму або створити нову сутність (модуль, сервіс, компонент і т. д.) в існуючому додатку.

Звичайно, можна створювати Angular-додатки самостійно в ручну, але це набагато складніше і ні до чого, коли є готовий інструмент. Тому при створенні і роботі з Angular-додатки рекомендується використовувати інтерфейс командного рядка. На цьому етапі встановлення Angular може вважатися завершеним.

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

•Генерація файлової структури нового проекту

•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 в браузері сторінки index.html

•**ng generate component my-component**-генерація нового компонента

•. . . . . .

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 проектів, компонентів, сервісів;

•Visual Studio Code (https://code.visualstudio.com/) і деякі плагіні;

•Браузер.

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!
```

TypeScript11/index.html

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>
  <input id="tip" value="5" class="require">
  <input id="Id_kalib" type="button" value="Калібрувати">
  <table>
  <tr id="id_test"></tr>
  </table>
</body>
  <script>
  var el = document.getElementById('tip').value;
  alert(typeof el);              // string
  console.log(typeof el);   // string
```

```
el = document.getElementById("Id_kalib");
alert(typeof el);            //object
console.log(typeof el);  // object
el = document.getElementById("id_test");
alert(typeof el);          //object
console.log(typeof el);  //object
</script>
```

Створення Angular-додатку “myToDo”

Для створення нового проекту в робочій директорії (наприклад, Desktop) виконайте команду ng new, щоб створити новий додаток під назвою todo:

```
ng new myToDo --routing=false --style=css
```

У поточній директорії команда ng new створить необхідний для роботи Angular-додаток. Додаткові прапори,--routing і--style, визначають, як обробляти навігацію та стилі у додатку. На створення проекту знадобиться час.

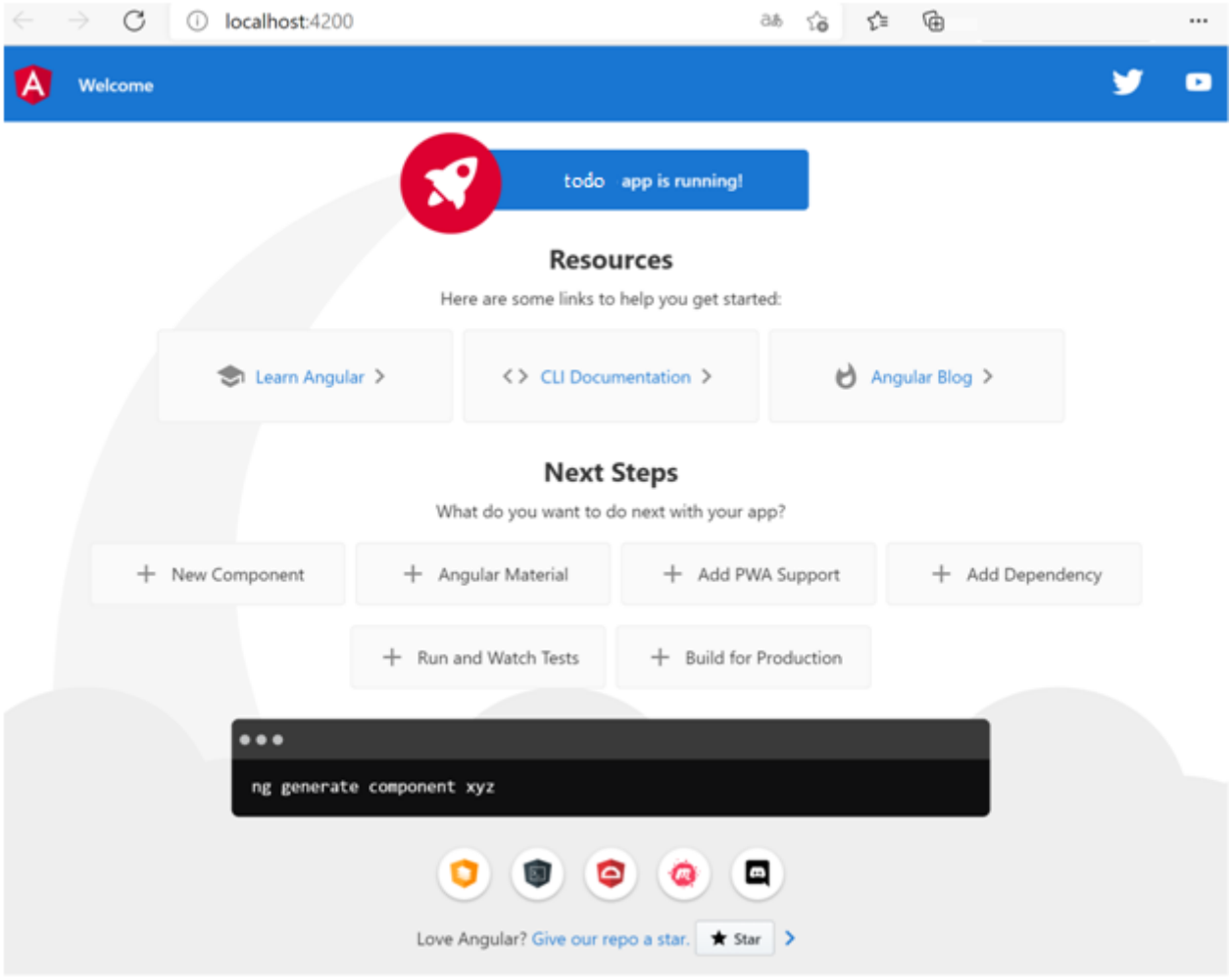
Далі перейдіть у ваш новий проект за допомогою cd:

```
cd myToDo
```

Запустіть вашу програму myToDo, виконавши команду ng serve:

```
ng serve
```

У браузері перейдіть на http://localhost:4200/, і ви побачите ваш новий додаток. Якщо змінити будь-який вихідний файл, програма автоматично перезавантажиться.



Так виглядає додаток Angular, створений командою ng new todo

Поки виконується ng serve, запускати інші команди можна у новій вкладці чи вікні терміналу. Якщо ви захочете зупинити роботу програми, натисніть Ctrl+C у терміналі, де вона була запущена.

Запуск команди ng new створює базову програму в папці myToDo.

Структура папки myToDo:

- e2e – директорія з інтеграційними тестами (end-to-end тести);
- node_modules – встановлені npm-модулі;
- src – вихідні файли;
- angular.json – опис конфігурації;
- package.json - метаінформація та список необхідних npm-модулів;
- README.md - опис;
- tsconfig.json – загальна конфігурація typescript;
- tslint.json - налаштування tslint (лінтер - інструмент для перевірки та виправлення помилок в файлах TypeScript, зараз вже відбувається перехід на eslint).

Структура каталогу src:

- app - модулі, компоненти, сервіси, директиви тощо;
- assets – статичний контент (зображення, аудіо);
- environments - конфігурації для конкретного середовища запуску;
- favicon.ico – іконка, що відображається у верхній частині вкладки браузера;
- 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;
- karma.conf.js – конфігураційний файл для запуску юніт-тестів з використанням Karma, запуск тестів можна виконати командою ng test;
- main.ts – точка входу вашого додатку. 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 – список модулів, що підключаються для підтримки кросбраузерності;
- styles.css - опис діючих глобально стилів;
- test.ts - відповідає за пошук та завантаження тестів при їх запуску;
- tsconfig.app.json - налаштування typescript;
- tsconfig.spec.json – налаштування typescript при запуску unit-тестів.

angular.json

Наступний крок після встановлення Angular та вивчення структури "скелета" програми - налаштування. angular.json - головний конфігураційний файл так званого Angular Workspace (директорія myToDo), згенерованого з використанням @angular/cli і поєднує в собі безліч файлів (саме додаток та створені бібліотеки для нього).

Angular_App/myToDo (додаток за замовчуванням, створений Angular CLI)

Основні властивості:

- version- версія Angular Workspace;
- newProjectRoot - вказує, де розташовуватимуться внутрішні додатки та бібліотеки, за замовчуванням projects;
- projects- опис конфігурацій для кожного з елементів Angular Workspace. За замовчуванням разом із основним генерується проект із інтеграційними тестами.

Опції:

- root- визначає директорію з файлами, включаючи конфігураційні, всього проекту;
- sourceRoot- визначає директорію з вихідними файлами;

- projectType- тип проекту, можливо тільки application або library;
- prefix- префікс, який використовуватиметься під час іменування компонентів і директив;
- schematics- дозволяє задавати відмінну від за замовчуванням конфігурацію сутностей Angular, наприклад, можна для всіх компонентів, що створюються, перевизначити роботу механізму Change Detection;
- architect- використовується для налаштування запуску або збирання.
- defaultProject- ім'я проекту, який використовуватиметься при використанні команд Angular CLI (за умовчанням основний).

Розглянемо докладно в angular.json параметр build властивості architect. Тут нас цікавлять options та configurations.

В options вказуються такі опції:

- outputPath- шлях, де буде знаходитись "зібраний" проект;
- index- шлях до index.html;
- main- шлях до main.ts;
- polyfills- шлях до polyfills.ts;
- tsConfig- шлях до tsconfig.app.json;
- assets- масив із зазначенням шляхів до статичного контенту, можуть бути папки чи окремі файли;
- styles- масив із зазначенням шляхів до стилів, причому стилі поширюються на всі додатки;
- scripts- масив із зазначенням шляхів до JavaScript-файлів, зазвичай тут підключаються сторонні бібліотеки, що не є модулями Angular, наприклад, jQuery.

Рекомендується завжди підключати сторонні скрипти та стилі не в index.html, а в angular.json.

У configurations вказуються налаштування, пов'язані з середовищем оточення (environment, далі СО) роботи програми.

Кожна властивість об'єкта configurations - назва СО з об'єктом налаштувань середовища як значення.

ng build -c production

СО вказується при запуску (збірці) як параметр --configuration (короткий запис -c).

В об'єкті налаштувань СО безліч параметрів, призначення більшості яких зрозуміло з їхньої назви. Зупинимося на fileReplacements.

У директорії environments за замовчуванням є два файли: environment.ts і environment.prod.ts. В них вказуються параметри, які залежать від СО, наприклад, адреса

сервера, з якого будуть запитуватися дані. За замовчуванням використовується environment.ts.

Об'єкт, що використовується як значення fileReplacements, дозволяє перевизначити джерело для зазначеного СО.

Можете відкрити angular.json та знайти відповідний код.

{

"replace": "src/environments/environment.ts",

"with": "src/environments/environment.prod.ts"

}

Package.json

Цей файл встановлює пакети та залежності, які будуть використовуватися проектом. У секції dependencies в основному визначаються пакети angular, які необхідні додатку для роботи. У секції devDependencies прописані лише пакети, які використовуватимуться для розробки. Зокрема, це пакети для роботи з мовою typescript, а також пакети, необхідні для компіляції програми за допомогою інфраструктури Angular CLI.

У секції "scripts" описані команди. Зокрема, команда ng serve запускає веб-сервер для тестування програми та саму програму. А команда ng build компілює програму.

Давайте розглянемо кожен ключ у вихідному файлі package.json:

- name: ім'я проекту, має бути написано малими літерами та підходити для URL. Ім'я може мати префікс області видимості (наприклад, @angular/angular-cli). Якщо проект приватний, ім'я проекту вибирати необов'язково, але якщо проект буде опубліковано, ім'я потрібно обов'язково, і воно має бути унікальним у репозиторії npm.
- version: номер версії, який має бути зрозумілим для node-semver. Це також необов'язково для приватних проектів, але є обов'язковим і дуже важливим для публічних.
- description: опис проекту. Це опціональне поле. Воно корисне і допомагає швидко знайти проект у репозиторії.
- main: вхідний файл проекту.
- scripts: ключ scripts очікує об'єкт з іменами скриптів як ключі та командами як значення. Він потрібний для вказівки сценаріїв, які можна запускати безпосередньо з командного рядка і які можуть виконувати всілякі завдання (запуск проекту на локальному сервері, збірка для продакшену або проведення тестів). Поле scripts – це те місце, де можна вручну внести більшість змін у типовий файл package.json.
- keywords: це масив, що допомагає знайти модуль у репозиторії npm.

- author: це поле приймає об'єкт із ключами name, email та url. Це дозволяє людям легко зв'язатися із автором проекту.
- license: приймає назву ліцензії з використанням ідентифікатора SPDX. За замовчуванням застосовується ліцензія ISC. Ще один популярний вибір - це MIT. Ви також можете використовувати UNLICENSED для приватних проектів та проектів із закритим кодом.

Управління залежностями

Основна сила npm – це можливість легко керувати залежностями проекту. Тому цілком природно, що файл package.json зосереджений здебільшого на перерахунку залежностей. Існують звичайні залежності, а також devDependencies, peerDependencies, optionalDependencies і bundledDependencies. Пройдемося по них детальніше:

- dependencies: звичайні залежності проекту. Саме тут, швидше за все, перебуватиме основна частина ваших залежностей. Додати такі залежності до свого проекту можна за допомогою $npm install my-dependency.
- devDependencies: залежності, які потрібні лише при розробці проекту. Наприклад, бібліотеки тестування та транспайлери найчастіше слід додавати як devDependencies. Додати devDependency можна за допомогою прапора npm –save-dev із командою install.
- optionalDependencies: залежності, які npm слід розглядати як опціональні. Якщо такі залежності не встановлені, npm не буде скаржитися або видавати помилку. Додати опціональні залежності можна за допомогою save-optional у команді install.
- bundledDependencies: очікує масив імен пакетів, які будуть пов'язані з проектом. Використовуйте прапорець –save-bundle з командою install, щоб залежність була додана до списку bundledDependencies.
- peerDependencies: однорангові залежності потрібні для визначення модулів, від яких залежить ваш проект. За відсутності однорангових залежностей npm видасть попередження.