

Лабораторна робота № 4

Поняття SPA та знайомство з Angular.

Web-застосунки дозволяють інтернет-користувачам отримати доступ до функціонала наданого вами сервісу або інструменту, використовуючи тільки браузер. Це значно економить час, так як програму не потрібно завантажувати і встановлювати, а отже - нею скористається набагато більше число користувачів.

Всі web-застосунки діляться на односторінкові (SPA) і багатосторінкові (MPA). SPA-застосунок, Single Page Application, або «застосунок однієї сторінки» - це тип web-застосунків, в яких завантаження необхідного коду відбувається на одну сторінку, що дозволяє заощадити час на повторне завантаження одних і тих же елементів. Взаємодія з користувачем організована як правило через динамічно підкачуємі HTML, CSS, JavaScript, зазвичай за допомогою AJAX. SPA нагадують нативні (native) застосунки, з тією лише різницею, що виконуються в рамках браузера, а не у власному процесі операційної системи

Angular - це фреймворк від компанії Google для створення клієнтських застосунків, та націлений перш за все на розробку SPA-рішень. В цьому плані Angular є спадкоємцем іншого фреймворка 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 та багато іншого. Однією з ключових особливостей Angular є те, що він використовує в якості мови програмування TypeScript. Використовувати мову TypeScript не обов'язково. При бажанні можемо писати програми на Angular за допомогою таких мов як Dart або JavaScript. Однак TypeScript все таки є основною мовою для Angular.

TypeScript представляє мову програмування на основі JavaScript, TypeScript зародився в компанії Microsoft, його творцем вважається програміст Андерс Хейлсберг, так само відомий як творець мови C#. Спробуємо розібратися, навіщо потрібна ще одна мова програмування для клієнтської сторони в середовищі Web, якщо можна використати традиційний JavaScript, яким володіє багато розробників і підтримка якого в співтоваристві програмістів досить висока.

По-перше TypeScript - це строго типізована і компільована мова, чим, можливо, буде ближче до програмістам Java, C#. Строга типізація зменшує кількість потенційних помилок, які могли б виникнути при розробці на JavaScript.

По-друге, TypeScript реалізує багато концепції, які властиві об'єктно-орієнтованим мовам, наприклад, наслідування, поліморфізм, інкапсуляція, модифікатори доступу і так далі.

По-третє, потенціал TypeScript дозволяє швидше і простіше писати великі складні комплексні програми, відповідно їх легше підтримувати, розвивати, масштабувати і тестувати, ніж на стандартному JavaScript.

У той же час TypeScript є надмножиною над JavaScript, а це значить, що будь-яка програма на JS є програмою на TypeScript. В TS можна використовувати всі ті конструкції, які застосовуються в JS - ті ж оператори, умовні та циклічні конструкції. Більш того код TS компілюється в javascript. В кінцевому рахунку, TS - це всього лише інструмент, який покликаний полегшити розробку застосунків, а згенерований компілятором TypeScript код JS підтримується переважною більшістю браузерів.

Для того, щоб почати роботу із Angular, необхідно попередньо виконати деякі налаштування середовища розробки. Для цього встановимо node.js та менеджер пакетів npm.

Node або Node.js - програмна платформа, заснована на движку V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованої мови в мову загального призначення. Node.js додає

можливість JavaScript взаємодіяти з пристроями вводу-виводу, підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи їх виклики з JavaScript-коду. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні застосунки. В основі Node.js лежить подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням. Для установки node.js перейдемо на офіційний сайт <https://nodejs.org/uk/> (рис. 1), завантажимо дистрибутив та виконаємо встановлення, користуючись підказками інсталятора.

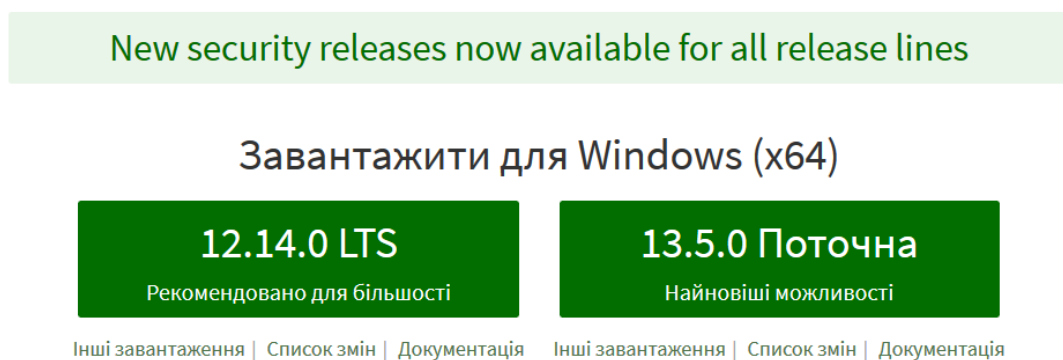


Рис. 1. Сторінка для завантаження node.js.

Щоб використовувати інструменти (або пакети) в Node.js нам потрібна можливість встановлювати і управляти ними. Для цього створено npm, пакетний менеджер Node.js. Він встановлює потрібні вам пакети і надає зручний інтерфейс для роботи з ними.

Для перевірки наявності node.js та npm запустимо командний рядок та виконаємо команди `node --version` та `npm --version` (рис. 2).

```
C:\>node --version
v10.15.3

C:\>npm --version
6.4.1
```

Рис. 2. Перевірка версії node та npm.

Для встановлення Angular та розгортання нового проекту завантажимо сторінку із інструкціями установки на офіційному сайті <https://angular.io/guide/setup-local> (рис. 3).

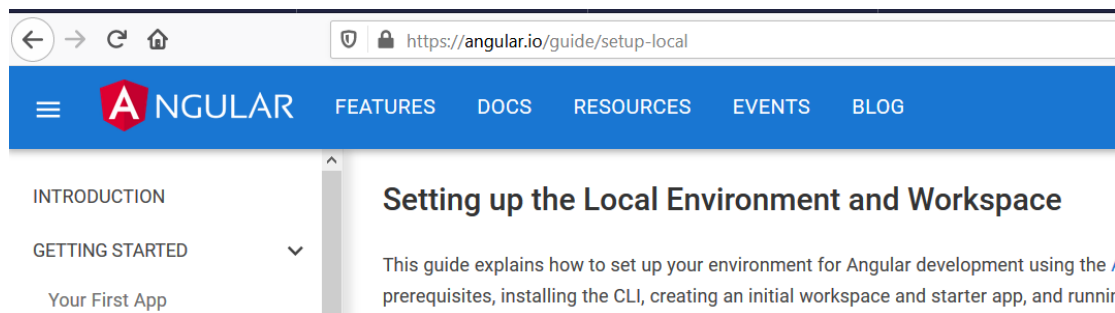


Рис. 3. Офіційний сайт проекту Angular.

Почнімо із встановлення утиліти командного рядку Angular CLI за допомогою менеджера пакетів npm. Для цього виконаємо команду `npm install -g @angular/cli` (рис. 4).

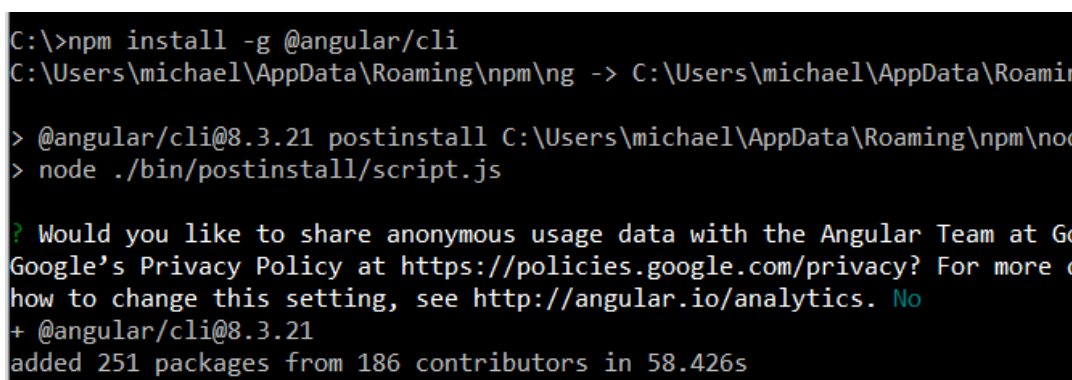


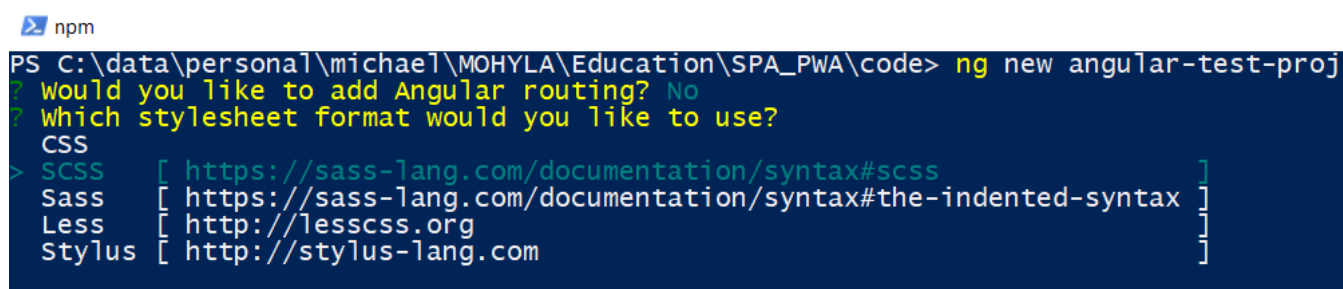
Рис. 4. Встановлення Angular CLI.

Для перевірки версії Angular CLI у командному рядку можемо виконати `ng --version` (рис. 5).



Рис. 5. Перевірка версії Angular CLI.

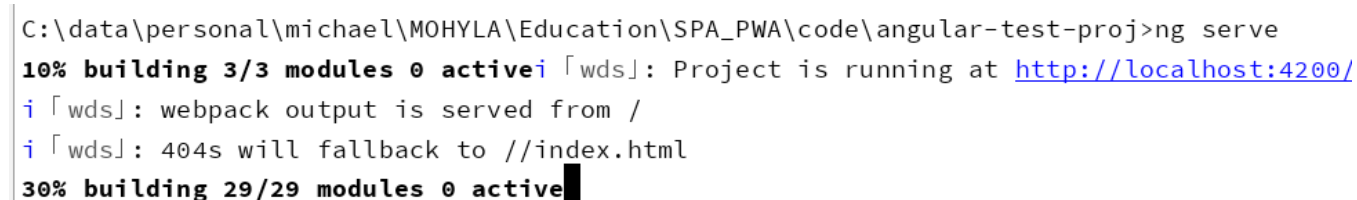
Після встановлення Angular CLI можна нарешті приступати до створення нового проекту Angular. Для цього у директорії, де плануємо створити проект, виконуємо команду `ng new project-name` (рис. 6). Відмовляємось від додавання роутінгу та обираємо SCSS у якості формату стилів.



```
PS C:\data\personal\michael\MOHYLA\Education\SPA_PWA\code> ng new angular-test-proj
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use?
  CSS
> SCSS [ https://sass-lang.com/documentation/syntax#scss ]
  Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less [ http://lesscss.org ]
  Stylus [ http://stylus-lang.com ]
```

Рис. 6. Створення нового проекту Angular.

Після завершення установки відкриваємо директорію `angular-test-proj` у редакторі коду. У командному рядку виконаємо команду `ng serve` (рис. 7).



```
C:\data\personal\michael\MOHYLA\Education\SPA_PWA\code\angular-test-proj>ng serve
10% building 3/3 modules 0 active i 「wds」: Project is running at http://localhost:4200/
i 「wds」: webpack output is served from /
i 「wds」: 404s will fallback to //index.html
30% building 29/29 modules 0 active
```

Рис. 7. Виконання проекту.

Після завершення build-у переходимо у браузері за посиланням `http://localhost:4200/` та переглядаємо зовнішній вигляд сторінки за замовченням після створення проекту (рис. 8).

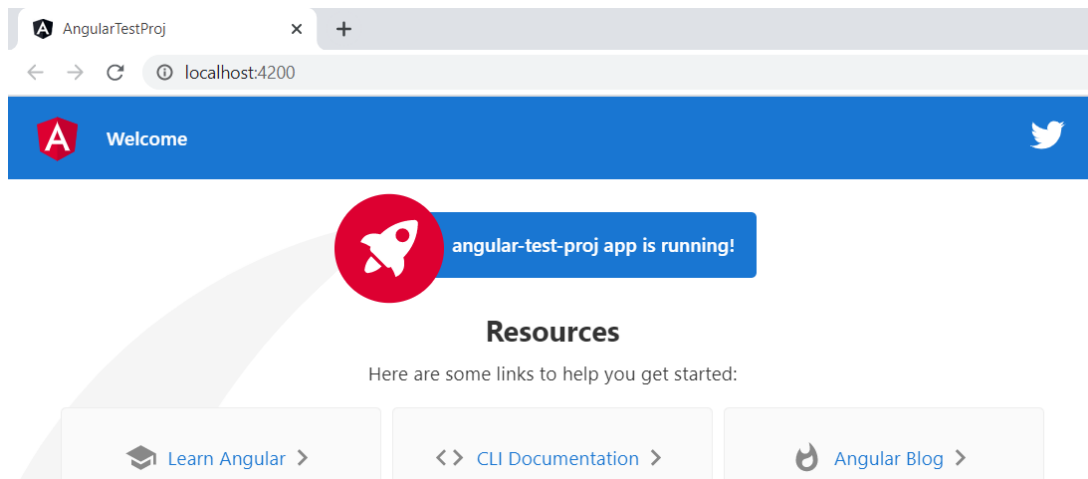


Рис. 8. Зовнішній вигляд застосунку.

Відкривши файл `angular.json` можемо побачити, що відправною точкою для завантаження сторінки застосунку є файл `src/index.html`, а точкою входу при виконанні коду - `src/main.ts` (рис. 9).

```

16      "architect": {
17        "build": {
18          "builder": "@angular-devkit/build-angular:browser",
19          "options": {
20            "outputPath": "dist/angular-test-proj",
21            "index": "src/index.html",
22            "main": "src/main.ts",
23            "polyfills": "src/polyfills.ts",

```

Рис. 9. Фрагмент файлу `angular.json`.

Переглянемо вміст файлу `src/index.html` (рис. 10).

```

1  <!doctype html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <title>AngularTestProj</title>
6    <base href="/">
7    <meta name="viewport" content="width=device-width, initial-scale=1">
8    <link rel="icon" type="image/x-icon" href="favicon.ico">
9  </head>
10 <body>
11   <app-root></app-root>
12 </body>
13 </html>

```

Рис. 10. Файл `src/index.html`.

Як бачимо, маємо стандартну html5 розмітку, за виключенням тегу app-root. Для того, щоб зрозуміти, що він означає і звідки береться вміст сторінки, наведеної на рис. 8, продовжимо інспектувати файли проекту та переглянемо src/main.ts (рис. 11).

```
1 import { enableProdMode } from '@angular/core';
2 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4 import { AppModule } from './app/app.module';
5 import { environment } from './environments/environment';
6
7 if (environment.production) {
8   enableProdMode();
9 }
10
11 platformBrowserDynamic().bootstrapModule(AppModule)
12   .catch( onrejected: err => console.error(err));
```

Рис. 11. Файл src/main.ts.

Тут нас передусім цікавить рядок bootstrapModule(AppModule), який вказує, що стартовим модулем буде AppModule. Також із import бачимо, що він розташований у ./app/app.module. Отже відкриємо та переглянемо файл src/app/app.module.ts (рис. 12).

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10   imports: [
11     BrowserModule
12   ],
13   providers: [],
14   bootstrap: [AppComponent]
15 })
16 export class AppModule { }
```

Рис. 12. Файл app.module.ts.

Поки що тут нас цікавить масив declarations, у якому міститься один елемент – компонент AppComponent, що підключений із файлу ./app.component. Відкриваємо файл app.component.ts (рис. 13).

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.scss']
7  })
8  export class AppComponent {
9    title = 'angular-test-proj';
10 }
```

Рис. 13. Файл app.component.ts.

Бачимо стандартний клас-компонент Angular. У декораторі @Component вказано ім'я тегу, що може бути використано в html-кодi, у нашому випадку (саме такий ми бачили у index.html на рис. 10) та ім'я файлу-шаблону, html-код якого буде виводитись при вказанні нашого тегу (<app-root></app-root>). Переглянемо нарешті вміст файлу app.component.html. Ми не наводимо його вміст, оскільки він досить великий і в цьому немає великого сенсу. Замість цього ви видалимо з нього усе і замінимо його наступним кодом (рис. 14).

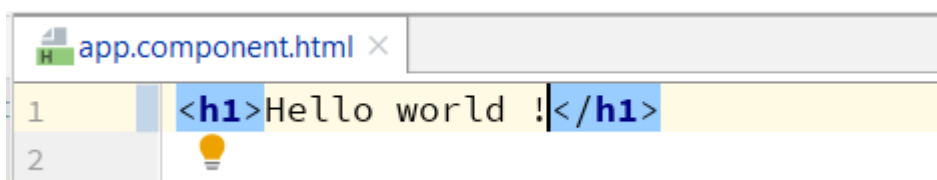
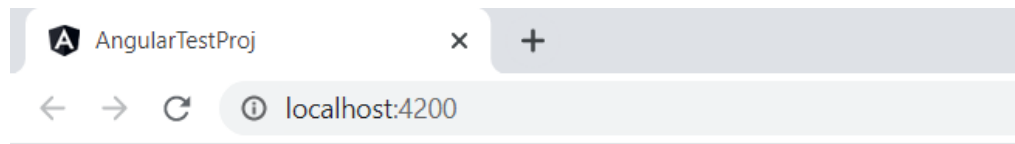


Рис. 14. Виконання змін у app.component.html.

Перейдемо в браузер та переглянемо результат (рис. 15). Як бачимо, зовнішній вигляд сторінки змінився згідно до виконаних нами змін у файлі app.component.html.



Hello world !

Рис. 15. Зовнішній вигляд сторінки після виконання змін.