



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Лабораторна робота №1

Реактивне програмування

Виконав студент групи ІТ-01: Чабан А.Є.		Перевірив:
		Полупан Ю. В.
		Дата:
		Оцінка:

Київ 2023

Зміст

Частина 1:	3
Завдання	3
Хід виконання	3
Опис основних структурних блоків Angular-додатку «HelloApp»: модулі, компоненти, шаблони.	4
Опис основних структурних блоків Angular-додатку «Shopping list»: модулі, компоненти, шаблони.	6
Опис файлу package.json. Призначення, основні параметри	8
Опис файлу tsconfig.json. Призначення, основні параметри	9
Опис файлу angular.json. Призначення, основні параметри	9
Розгортання Angular-додатку «Shopping list» на платформі FireBase	11
Частина 2:	12
Завдання	12
Хід виконання	13
Інтерполяція в Angular	13
Прив'язка властивостей елементів HTML	13
Прив'язка до атрибуту	13
Прив'язка до події	14
Двостороння прив'язка	15
Прив'язка до класів CSS	16
Прив'язка стилів	16
Розгортання Angular-додатку «Binding1» на платформі FireBase	17

Частина 1:

Завдання

- 1) створити при допомозі текстового редактора простий Angular-додаток "HelloApp";
- 2) при допомозі текстового редактора створити простий додаток «Shopping list»;
- 3) зробити звіт по роботі;
- 4) розгорнути Angular-додаток «Shopping list» на платформі FireBase.

Хід виконання

1) Встановлення Node.js і Angular CLI:

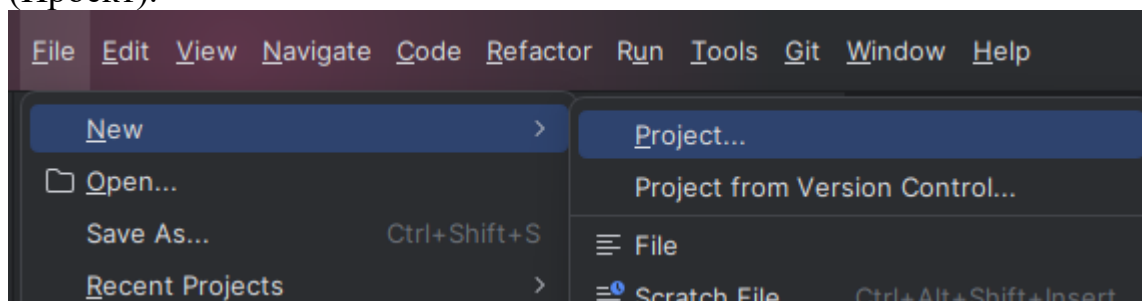
Перш за все, завантажимо та встановимо Node.js з офіційного веб-сайту Node.js.

Далі відкриємо командний рядок (термінал) і виконаємо наступну команду для встановлення Angular CLI глобально:

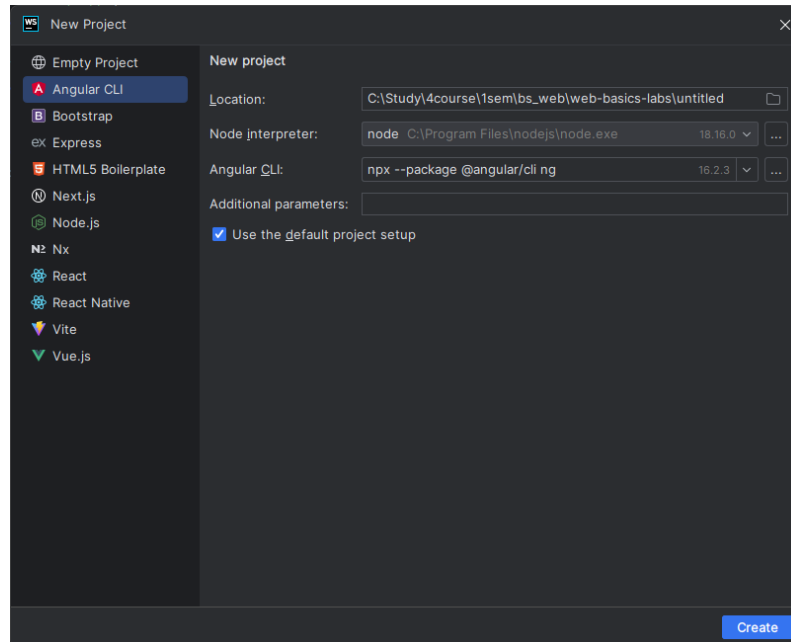
```
npm install -g @angular/cli
```

Створення нового Angular додатку (використаємо WebStorm):

Відкриємо WebStorm і оберемо "File" (Файл) -> "New" (Новий) -> "Project" (Проект).



У вікні "New Project" (Новий проект) оберемо "Angular CLI" як тип проекту та натискаємо "Next" (Далі).



Angular додаток має наступну структуру:

Опис основних структурних блоків Angular-додатку «HelloApp»: модулі, компоненти, шаблони.

app.module.ts:

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

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

Цей файл є головним модулем додатку. Він імпортує та налаштовує всі необхідні модулі та компоненти для вашого додатку.

imports: Список модулів, які ваш додаток використовуватиме. Ми використовуємо BrowserModule та FormsModule.

declarations: Список всіх компонентів, які належать до цього модулю. У нас є один компонент, AppComponent.

bootstrap: Вказує, який компонент повинен бути кореневим для додатку: AppComponent.

app.component.ts:

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

@Component({
  selector: 'my-app',
  template: `<label>Введіть назву:</label>

  <input [(ngModel)]="name" placeholder="name">
  <h1>Ласкаво просимо {{name}}!</h1>`
})
export class AppComponent {
  name = ' ';
}
```

Цей файл містить опис головного компонента, AppComponent.

selector: Ця властивість визначає, як буде використовуватись цей компонент у шаблонах інших компонентів.

template: HTML-розмітка компонента. В цьому випадку, він містить рядок HTML для введення тексту та відображення повідомлення зі змінною name.

main.ts:

```
import {platformBrowserDynamic} from '@angular/platform-browser-dynamic';
import {AppModule} from './app/app.module';

const platform = platformBrowserDynamic();
platform.bootstrapModule(AppModule);
```

Цей файл є головним файлом для запуску додатку. Він імпортує platformBrowserDynamic та головний модуль AppModule, а потім викликає bootstrapModule, щоб запустити додаток.

polyfills.ts:

```
import 'zone.js/dist/zone';
```

Цей файл містить поліфіли, які додають підтримку для деяких функцій JavaScript і браузерних функцій для старих браузерів або платформ. У нашому випадку, він імпортує zone.js, який використовується Angular для управління зонами і подіями.

Опис основних структурних блоків Angular-додатку «Shopping list»: модулі, компоненти, шаблони.

Додаток Shopping List має схожу структуру, основною відмінністю є файл `app.component.ts`:

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

class Item {
  purchase: string;
  done: boolean;
  price: number;

  constructor(purchase: string, price: number) {

    this.purchase = purchase;
    this.price = price;
    this.done = false;
  }
}

@Component({
  selector: 'my-app',
  template: `
    <div class="page-header">
      <h1> Shopping list </h1>
    </div>
    <div class="panel">
      <div class="form-inline">
        <div class="form-group">
          <div class="col-md-8">
            <input class="form-control" [(ngModel)]="text" placeholder="Назва"/>
          </div>
        </div>
        <div class="form-group">
          <div class="col-md-6">
            <input type="number" class="form-control" [(ngModel)]="price"
              placeholder="Ціна"/>
          </div>
        </div>
        <div class="form-group">
          <div class="col-md-offset-2 col-md-8">
            <button class="btn btn-default" (click)="addItem(text,
price)">Додати</button>
          </div>
        </div>
      </div>
      <table class="table table-striped">
        <thead>
          <tr>
            <th>Предмет</th>
            <th>Ціна</th>
            <th>Куплено</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let item of items">
            <td>{{item.purchase}}</td>
            <td>{{item.price}}</td>
```

```

        <td><input type="checkbox" [(ngModel)]="item.done"/></td>
    </tr>
</tbody>
</table>
</div>`
))
export class AppComponent {
    text: string = "";
    price: number = 0;

    items: Item[] =
    [
        {purchase: "Хліб", done: false, price: 15.9},
        {purchase: "Вершкове масло", done: false, price: 60},
        {purchase: "Картопля", done: true, price: 22.6},
        {purchase: "Сир", done: false, price: 310}
    ];

    addItem(text: string, price: number): void {
        if (text == null || text.trim() == "" || price == null)
            return;
        this.items.push(new Item(text, price));
    }
}

```

Клас Item:

Це внутрішній клас, який описує структуру об'єкта Item, який представляє елементи покупок.

purchase: Рядок, що представляє назву покупки.

done: Логічний тип, що вказує, чи була купка вже зроблена.

price: Число, що представляє ціну покупки.

Конструктор ініціалізує ці властивості.

Компонент AppComponent:

Це основний компонент додатку, який описує його логіку та відображення.

selector: Вказує, як можна використовувати цей компонент у шаблонах інших компонентів.

Змінні та масив items:

text: Рядок для зберігання назви покупки, яку користувач вводить у текстовому полі.

price: Число для зберігання ціни покупки, яку користувач вводить у числовому полі.

items: Масив об'єктів типу Item, який представляє список покупок. Початкові дані вже визначені.

Метод addItem:

Цей метод викликається, коли користувач натискає кнопку "Додати".

Він приймає два параметри: text (назва покупки) і price (ціна покупки).

Перевіряється, чи введені дані не є порожніми, і якщо так, то створюється новий

об'єкт `Item` і додається до масиву `items`.

Опис файлу `package.json`. Призначення, основні параметри

`package.json`:

```
{
  "name": "angular-lab1",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^16.2.0",
    "@angular/common": "^16.2.0",
    "@angular/compiler": "^16.2.0",
    "@angular/core": "^16.2.0",
    "@angular/forms": "^16.2.0",
    "@angular/platform-browser": "^16.2.0",
    "@angular/platform-browser-dynamic": "^16.2.0",
    "@angular/router": "^16.2.0",
    "rxjs": "~7.8.0",
    "tslib": "^2.3.0",
    "zone.js": "~0.13.0"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "^16.2.1",
    "@angular/cli": "~16.2.1",
    "@angular/compiler-cli": "^16.2.0",
    "@types/jasmine": "~4.3.0",
    "jasmine-core": "~4.6.0",
    "karma": "~6.4.0",
    "karma-chrome-launcher": "~3.2.0",
    "karma-coverage": "~2.2.0",
    "karma-jasmine": "~5.1.0",
    "karma-jasmine-html-reporter": "~2.1.0",
    "typescript": "~5.1.3"
  }
}
```

Цей файл містить інформацію про додаток та його залежності.

`"scripts"`: Містить команди, які можуть використовуватись для запуску різних операцій, таких як запуск сервера розробки (`"start"`), збірка додатку (`"build"`), тощо.

`"dependencies"` та `"devDependencies"`: Містять залежності додатку, які встановлюються за допомогою `npm`. Наприклад, `"@angular/core"` є основною залежністю Angular.

Опис файлу tsconfig.json. Призначення, основні параметри tsconfig.json:

```
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "module": "esnext",
    "moduleResolution": "node",
    "target": "es2022",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
      "es2022",
      "dom"
    ]
  },
  "files": [
    "src/main.ts",
    "src/polyfills.ts"
  ],
  "include": [
    "src/**/*.d.ts"
  ]
}
```

Цей файл містить конфігурацію компілятора TypeScript для додатку. compilerOptions: Налаштування компілятора TypeScript, такі як версія ECMAScript ("target"), додавання підтримки декораторів ("experimentalDecorators"), тощо.

Опис файлу angular.json. Призначення, основні параметри angular.json:

```
{
  "version": 1,
  "projects": {
    "angularLab1": {
      "projectType": "application",
      "root": "",
      "sourceRoot": "src",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/angularLab1",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.json",
            "aot": true
          }
        },
        "serve": {
```

```
    "builder": "@angular-devkit/build-angular:dev-server",
    "options": {
      "browserTarget": "angularLab1:build"
    }
  },
  "cli": {
    "analytics": "0e2b16c1-8c63-4591-80a5-39b1ec362a8a"
  }
}
```

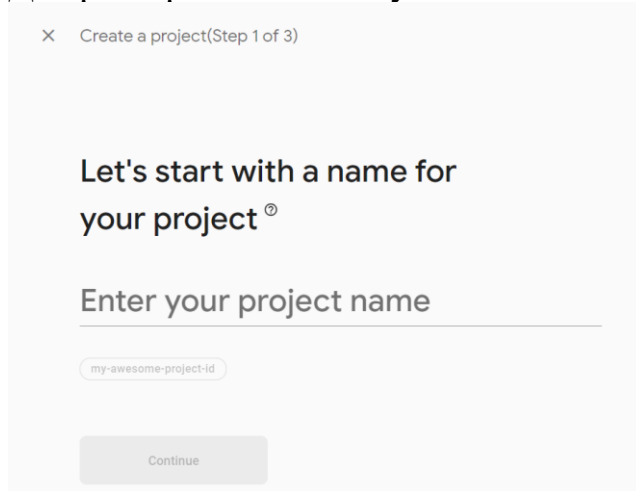
Цей файл містить конфігурацію Angular проекту, таку як налаштування для збірки, розгортання та інше.

"projects": Описує ваші проекти Angular. Ваш додаток має проект з ім'ям "angularLab1".

"architect": Налаштування для різних операцій, таких як збірка ("build") та запуск сервера розробки ("serve").

Розгортання Angular-додатку «Shopping list» на платформі FireBase

Для розгортання додатку на Firebase необхідно створити новий проект



Назвемо його ChabanIT01Laba1-1.

Далі створимо компонент проекту, в нашому випадку це Web Application



Далі за допомогою терміналу потрібно виконати декілька команд:

firebase login – для логіну

firebase init – для ініціалізації firebase

При ініціалізації необхідно вибрати наступні пункти:

```
Are you ready to proceed? Yes
Which Firebase features do you want to set up for this directory? Press Space to select features, then Enter to confirm your choices. (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)
  ( ) Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision default instance
  ( ) Firestore: Configure security rules and indexes files for Firestore
  ( ) Functions: Configure a Cloud Functions directory and its files
  x ( ) Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
  ( ) Hosting: Set up GitHub Action deploys
  ( ) Storage: Configure a security rules file for Cloud Storage
  ( ) Emulators: Set up local emulators for Firebase products
(Move up and down to reveal more choices)
```

Далі потрібно вказати папку з уже збілдженим проектом (ng build)

=== Hosting Setup

Your public directory is the folder (relative to your project directory) that will contain Hosting assets to be uploaded with `firebase deploy`. If you have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? `dist/purchaseapp`

І слідувати подальшим інструкціям, обрати наш поточний проект ChabanIT01Laba1-1.

Далі необхідно виконати команду `firebase deploy`

```
PS C:\Study\4course\1sem\reactive_programming\lab1\purchaseapp> firebase deploy

=== Deploying to 'chabanit01laba1-1'...

i  deploying hosting
i  hosting[chabanit01laba1-1]: beginning deploy...
i  hosting[chabanit01laba1-1]: found 5 files in dist/purchaseapp
+  hosting[chabanit01laba1-1]: file upload complete
i  hosting[chabanit01laba1-1]: finalizing version...
+  hosting[chabanit01laba1-1]: version finalized
i  hosting[chabanit01laba1-1]: releasing new version...
+  hosting[chabanit01laba1-1]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/chabanit01laba1-1/overview
Hosting URL: https://chabanit01laba1-1.web.app
PS C:\Study\4course\1sem\reactive_programming\lab1\purchaseapp> 
```

Додаток доступний за посиланням: [Покупки \(chabanit01laba1-1.web.app\)](https://chabanit01laba1-1.web.app)

Частина 2:

Завдання

Створити два Angular-додатки під назвою Binding1 та Binding2, як показано в частині 1.

- 1) Для Angular-додатку Binding1 виконати вправи 1-5;
- 2) Для Angular-додатку Binding2 виконати вправи 6-7;
- 3) Зробити звіт по роботі (по Angular-додатках Binding1 та Binding2);
- 4) Angular-додаток Binding1 розвернути на платформі FireBase.

Хід виконання

Інтерполяція в Angular:

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

@Component({
  selector: 'my-app',
  template: `<p>Ім'я: {{name}}</p>
<p>Вік: {{age}}</p>`
})
export class AppComponent {
  name = "Tom";
  age = 25;
}
```

Інтерполяція в Angular дозволяє вставляти значення з змінних компоненту безпосередньо в шаблон HTML. В даному коді прикладом інтерполяції є вставка значення змінних `name` та `age` в текст параграфів.

Якщо в процесі роботи програми властивості `name` і `age` в компоненті змінять своє значення, то також зміниться значення в розмітці html, яка прив'язана до цих властивостей.

Прив'язка властивостей елементів HTML

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

@Component({
  selector: 'my-app',
  template: ` <p>Ім'я: {{name}}</p>
<p>Возраст: {{age}}</p>
<input type="text" [value]="name"/>
<input type="text" [value]="age"/>`
})
export class AppComponent {
  name = "Tom";
  age = 25;
}
```

Прив'язка властивостей HTML-елементів дозволяє динамічно змінювати їхні властивості. У цьому коді, значення змінних `name` та `age` прив'язані до властивостей `value` двох текстових полів `<input>`.

Прив'язка до атрибуту:

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

@Component({
  selector: 'my-app',
  template: `
    <p>Ім'я: {{name}}</p>
    <p>Вік: {{age}}</p>
    <input type="text" [value]="name"/><br/>
    <input type="text" [value]="age"/>
    <p [textContent]="name"></p>
    <table border="1">
      <tr>
        <td [attr.colspan]="colspan">One-Two</td>

```

```

        </tr>
        <tr>
            <td>Three</td>
            <td>Four</td>
        </tr>
        <tr>
            <td>Five</td>
            <td>Six</td>
        </tr>
    </table>`
    })
    export class AppComponent {
        name = 'Tom';
        age = 25;
        colspan = 2;
    }

```

Прив'язка до атрибуту дозволяє змінювати атрибути HTML-елементів на основі значень змінних. У цьому коді, значення змінної `colspan` прив'язане до атрибуту `colspan` елемента `<td>`.

Прив'язка до події:

```

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

@Component({
    selector: 'my-app',
    template: `<p>Ім'я: {{name}}</p>
    <p>Вік: {{age}}</p>
    <input type="text" [value]="name"/><br/>
    <input type="text" [value]="age"/>
    <p [textContent]="name"></p>
    <table border="1">
        <tr>
            <td [attr.colspan]="colspan">One-Two</td>
        </tr>
        <tr>
            <td>Three</td>
            <td>Four</td>
        </tr>
        <tr>
            <td>Five</td>
            <td>Six</td>
        </tr>
    </table>
    <p>Кількість кліків {{count}}</p>
    <button (click)="increase()">Click</button>`
})
export class AppComponent {
    name = 'Tom';
    age = 25;
    colspan = 2;
    count: number = 0;

    increase(): void {
        this.count++;
    }
}

```

Прив'язка до подій дозволяє реагувати на події, які виникають на сторінці. У цьому коді, подія click на кнопці викликає метод increase(), який збільшує значення змінної count.

Двостороння прив'язка:

```
import { Component } from '@angular/core';
@Component({
  selector: 'my-app',
  template: `<p>Ім'я: {{name}}</p>
<p>Вік: {{age}}</p>
<input type="text" [value]="name"/><br/>
<input type="text" [value]="age"/>
<p [textContent]="name"></p>
<table border="1">
  <tr>
    <td [attr.colspan]="colspan">One-Two</td>
  </tr>
  <tr>
    <td>Three</td>
    <td>Four</td>
  </tr>
  <tr>
    <td>Five</td>
    <td>Six</td>
  </tr>
</table>
<p>Кількість кліків {{count}}</p>
<button (click)="increase()">Click</button>
<p>Кількість кліків {{count_2}}</p>
<button (click)="increase_2($event)">Click</button>
<p>Привіт {{name}}</p>
<input type="text" [(ngModel)]="name"/> <br><br>
<input type="text" [(ngModel)]="name"/>`
})
export class AppComponent {
  name = 'Tom';
  age = 25;
  colspan = 2;
  count: number = 0;
  count_2: number = 0;
  increase() : void {
    this.count++;
  }
  increase_2($event : any) : void {
    this.count_2++;
    console.log($event);
  }
}
```

Двостороння прив'язка дозволяє не тільки відображати значення змінних у шаблоні, але й оновлювати їх значення на основі введення користувача. У цьому коді, двостороння прив'язка здійснюється за допомогою [(ngModel)], яка зв'язує значення поля вводу зі змінною name.

Прив'язка до класів CSS:

```
import { Component } from '@angular/core';
@Component({
  selector: 'my-app',
  template: `<div [class.isredbox]="isRed"></div>
<div [class.isredbox]="!isRed"></div>
<input type="checkbox" [(ngModel)]="isRed" />`,
  styles: [`
    div {
      width: 50px;
      height: 50px;
      border: 1px solid #ccc
    }

    .isredbox {
      background-color: red;
    }
  `]
})
export class AppComponent {
  isRed = false;
}
```

У шаблоні ми використовуємо прив'язку до класів CSS за допомогою `[class.isredbox]="isRed"`. Якщо `isRed` дорівнює `true`, то клас `isredbox` буде доданий до елемента `<div>`, і елемент матиме червоний фон. Якщо `isRed` дорівнює `false`, то клас `isredbox` буде видалений, і елемент матиме стандартний фон.

Також, ми використовуємо `<input type="checkbox">` з двосторонньою прив'язкою `[(ngModel)]="isRed"`, яка дозволяє нам змінювати значення `isRed` шляхом вибору або зняття позначки з чекбоксу.

Прив'язка стилів:

```
import { Component } from '@angular/core';
@Component({
  selector: 'my-app',
  template: `
    <div [class.isredbox]="isRed"></div>
    <div [class.isredbox]="!isRed"></div>
    <input type="checkbox" [(ngModel)]="isRed"/>
    <div [class]="blue"></div> <br><br>
    <div [style.backgroundColor]="isyellow? 'yellow' : 'blue'"></div>
    <div [style.background-color]="!isyellow ? 'yellow' : 'blue'"></div>
    <input type="checkbox" [(ngModel)]="isyellow"/>
  `,
  styles: [`
    div {width:50px; height:50px; border:1px solid #ccc}
    .isredbox{background-color:red;}
    .isbluebox{background-color:blue;}
  `]
})
export class AppComponent {
  isRed = false;
  isyellow=false;
}
```

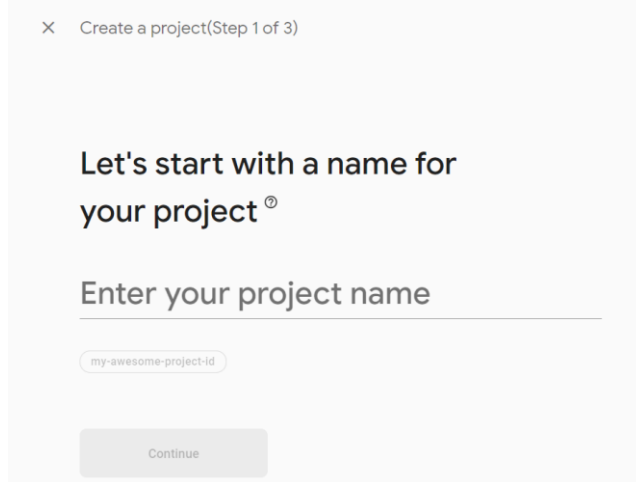


```
blue = "isbluebox"
}
```

Ми використовуємо прив'язку стилів за допомогою `[style.backgroundColor]` та `[style.background-color]`. Залежно від значення `isyellow`, фоновий колір елементів буде встановлюватися на жовтий або синій колір.

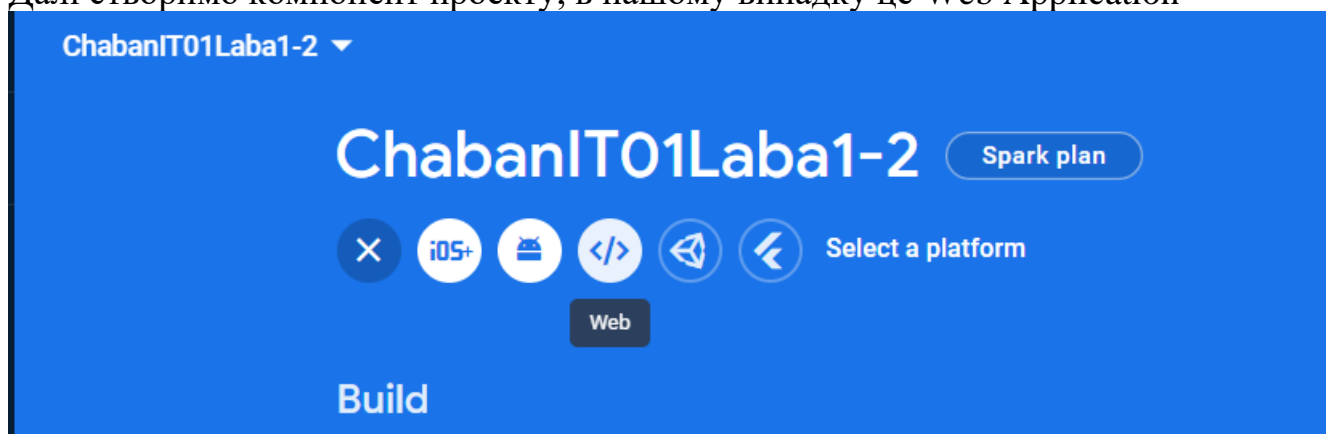
Розгортання Angular-додатку «Binding1» на платформі FireBase

Для розгортання додатку на Firebase необхідно створити новий проект

A screenshot of the 'Create a project' dialog in the Firebase console. The dialog has a title bar with a close button and the text 'Create a project (Step 1 of 3)'. The main content area says 'Let's start with a name for your project' followed by a registered trademark symbol. Below this is a text input field labeled 'Enter your project name' with the placeholder text 'my-awesome-project-id'. At the bottom is a 'Continue' button.

Назвемо його ChabanIT01Laba1-2.

Далі створимо компонент проекту, в нашому випадку це Web Application



Далі за допомогою терміналу потрібно виконати декілька команд:

`firebase login` – для логіну

`firebase init` – для ініціалізації firebase

При ініціалізації необхідно вибрати наступні пункти:

```
> Are you ready to proceed? Yes
> Which Firebase features do you want to set up for this directory? Press Space to select features, then Enter to confirm your choices. (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)
  ( ) Realtime Database: Configure a security rules file for Realtime Database and (optionally) provision default instance
  ( ) Firestore: Configure security rules and indexes files for Firestore
  ( ) Functions: Configure a Cloud Functions directory and its files
  x ( ) Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
  ( ) Hosting: Set up GitHub Action deploys
  ( ) Storage: Configure a security rules file for Cloud Storage
  ( ) Emulators: Set up local emulators for Firebase products
(Move up and down to reveal more choices)
```

Далі потрібно вказати папку з уже збілдженим проектом (ng build)

=== Hosting Setup

Your public directory is the folder (relative to your project directory) that will contain Hosting assets to be uploaded with `firebase deploy`. If you have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? dist/binding1

І слідувати подальшим інструкціям, обрати наш поточний проект ChabanIT01Laba1-2.

Далі необхідно виконати команду `firebase deploy`

```
+ hosting[chabanit01laba1-2-7c63e]: file upload complete
i hosting[chabanit01laba1-2-7c63e]: finalizing version...
+ hosting[chabanit01laba1-2-7c63e]: version finalized
i hosting[chabanit01laba1-2-7c63e]: releasing new version...
+ hosting[chabanit01laba1-2-7c63e]: release complete

+ Deploy complete!
```

Project Console: <https://console.firebase.google.com/project/chabanit01laba1-2-7c63e/overview>

Hosting URL: <https://chabanit01laba1-2-7c63e.web.app>

PS C:\Study\4course\1sem\reactive_programming\lab1\binding1>

Додаток доступний за посиланням: [Binding1 \(chabanit01laba1-2-7c63e.web.app\)](https://chabanit01laba1-2-7c63e.web.app)