

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»**



**Лабораторна робота № 1**

**з дисципліни «Реактивне програмування»**

**на тему:**

**«Створення Angular-додатків «HelloApp» і «Shopping list».**

**Прив'язка даних в Angular.»**

Перевірила  
доцент  
Полупан Ю. В.

Виконав  
студент ФІОТ  
групи ІС-01  
Адамов Денис

## Лабораторна робота №1

**Тема:** Створення Angular-додатків «HelloApp» і «Shopping list» .

**Мета роботи:** навчитися встановлювати необхідне ПО для створення Angular додатку. Навчитися створювати шаблон в компоненті Angular

**Завдання:**

### Частина 1. : Створення Angular-додатків “HelloApp” і «Shopping list»

1. створити за допомогою текстового редактора простий Angular-додаток “HelloApp”;
2. за допомогою текстового редактора створити простий додаток «Shopping list»;
3. зробити звіт по роботі;
4. розгорнути Angular-додаток «Shopping list» на платформі FireBase.

### Частина 2: Прив'язка даних.

**Тема:** Навчитися працювати з прив'язкою даних.

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

I) Для Angular-додатку Binding1 виконати вправи 1-5;

II) Для Angular-додатку Binding2 виконати вправи 6-7;

III) Зробити звіт по роботі (по Angular-додатках Binding1 та Binding2);

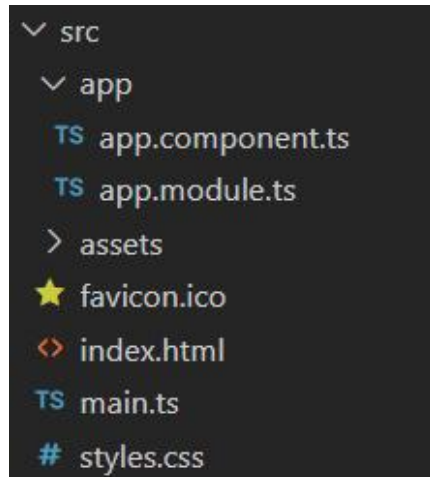
IV) Angular-додаток Binding1 розвернути на платформі FireBase.

**Вправи:**

- 1) Прив'язка DOM до значення компонента (одностороння);
- 2) Прив'язка властивості елемента DOM до значення компонента (одностороння);
- 3) Прив'язка методу компонента до події DOM;
- 4) Двостороння прив'язка (two-way binding);
- 5) Прив'язка до атрибуту елемента html;
- 6) Прив'язка до класу CSS;
- 7) Прив'язка до атрибуту елемента html.

## Частина 1: Створення Angular-додатків «HelloApp» і «Shopping list»

### HelloApp



Основна структура застосунку helloapp

Папка src містить усі вихідні файли.

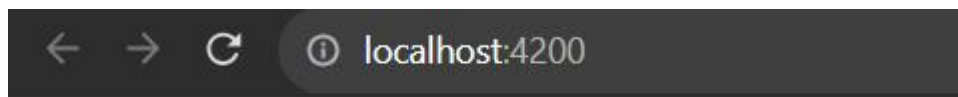
У папці app зберігаються компонент і модуль.

```
lab1 > helloapp > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'my-app',
5    template: `<label>Введіть назву:</label>
6      <input [(ngModel)]="name" placeholder="name" />
7      <h1>Ласкаво просимо {{ name }}!</h1>`,
8  })
9  export class AppComponent {
10    name = '';
11  }
```

app.component.ts

```
lab1 > helloapp > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5  @NgModule({
6    imports: [BrowserModule, FormsModule],
7    declarations: [AppComponent],
8    bootstrap: [AppComponent],
9  })
10 export class AppModule {}
```

app.module.ts

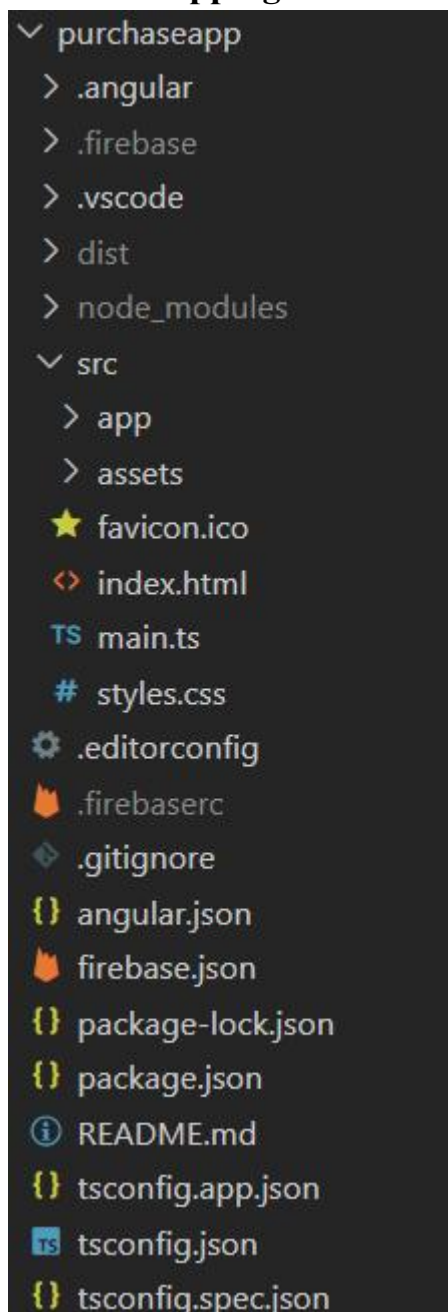


Введіть назву:

# Ласкаво просимо abuba!

Працюючий застосунок helloapp

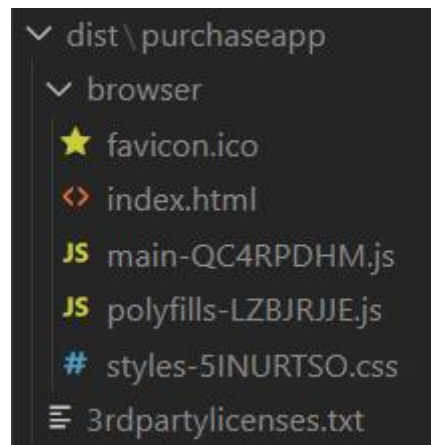
## ShoppingList



Структура застосунку purchaseapp

Папка src містить усі вихідні файли.

У папці app зберігаються компонент і модуль.  
Папка dist містить файли, які розміщуються на хостингу.



Структура папки dist

### 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, що використовуються в проекті. Зокрема, команда ng serve запускає простий веб-сервер для тестування програми та її запуску. А команда ng build компілює програму.

Файл було згенеровано автоматично запуском команди ng new, тому він містить деякі популярні залежності та команди, які в даній роботі не використовуються.

```
lab1 > purchaseapp > {} package.json > ...
1  {
2    "name": "purchaseapp",
3    "version": "0.0.0",
4    "scripts": {
5      "ng": "ng",
6      "start": "ng serve",
7      "build": "ng build",
8      "watch": "ng build --watch --configuration development",
9      "test": "ng test"
10  },
```

```
11  "private": true,  
12  "dependencies": {  
13    "@angular/animations": "^17.0.0",  
14    "@angular/common": "^17.0.0",  
15    "@angular/compiler": "^17.0.0",  
16    "@angular/core": "^17.0.0",  
17    "@angular/forms": "^17.0.0",  
18    "@angular/platform-browser": "^17.0.0",  
19    "@angular/platform-browser-dynamic": "^17.0.0",  
20    "@angular/router": "^17.0.0",  
21    "rxjs": "~7.8.0",  
22    "tslib": "^2.3.0",  
23    "zone.js": "~0.14.2"  
24  },  
25  "devDependencies": {  
26    "@angular-devkit/build-angular": "^17.0.0",  
27    "@angular/cli": "^17.0.0",  
28    "@angular/compiler-cli": "^17.0.0",  
29    "@types/jasmine": "~5.1.0",  
30    "jasmine-core": "~5.1.0",  
31    "karma": "~6.4.0",  
32    "karma-chrome-launcher": "~3.2.0",  
33    "karma-coverage": "~2.2.0",  
34    "karma-jasmine": "~5.1.0",  
35    "karma-jasmine-html-reporter": "~2.1.0",  
36    "typescript": "~5.2.2"  
37  }  
38 }
```

### tsconfig.json

Цей файл визначає параметри компілятора TypeScript. Опція `compilerOptions` встановлює параметри компіляції. `angularCompilerOptions` визначає параметри при ahead-of-time компіляції.



```

lab1 > purchaseapp > tsconfig.json > {} compilerOptions
1  /* To learn more about this file see: https://angular.io/config/tsconfig. */
2  {
3    "compileOnSave": false,
4    "compilerOptions": {
5      "outDir": "./dist/out-tsc",
6      "forceConsistentCasingInFileNames": true,
7      "strict": true,
8      "noImplicitOverride": true,
9      "noPropertyAccessFromIndexSignature": true,
10     "noImplicitReturns": true,
11     "noFallthroughCasesInSwitch": true,
12     "esModuleInterop": true,
13     "sourceMap": true,
14     "declaration": false,
15     "downlevelIteration": true,
16     "experimentalDecorators": true,
17     "moduleResolution": "node",
18     "importHelpers": true,
19     "target": "ES2022",
20     "module": "ES2022",
21     "useDefineForClassFields": false,
22     "lib": [
23       "ES2022",
24       "dom"
25     ]
26   },
27   "angularCompilerOptions": {
28     "enableI18nLegacyMessageIdFormat": false,
29     "strictInjectionParameters": true,
30     "strictInputAccessModifiers": true,
31     "strictTemplates": true
32   }
33 }

```

## angular.json

Цей файл визначає правила для angular CLI.

```

1  {
2    "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
3    "version": 1,
4    "newProjectRoot": "projects",
5    "projects": {
6      "purchaseapp": {
7        "projectType": "application",
8        "schematics": {},
9        "root": "",
10       "sourceRoot": "src",
11       "prefix": "app",
12       "architect": {
13         "build": {
14           "builder": "@angular-devkit/build-angular:application",

```

```
15  ✓      "options": {
16          "outputPath": "dist/purchaseapp",
17          "index": "src/index.html",
18          "browser": "src/main.ts",
19  ✓      "polyfills": [
20          "zone.js"
21      ],
22      "tsConfig": "tsconfig.app.json",
23  ✓      "assets": [
24          "src/favicon.ico",
25          "src/assets"
26      ],
27  ✓      "styles": [
28          "src/styles.css"
29      ],
30      "scripts": []
31  },
32  ✓      "configurations": {
33  ✓          "production": {
34  ✓              "budgets": [
35  ✓                  {
36                      "type": "initial",
37                      "maximumWarning": "500kb",
38                      "maximumError": "1mb"
39                  },
40  ✓                  {
41                      "type": "anyComponentStyle",
42                      "maximumWarning": "2kb",
43                      "maximumError": "4kb"
44                  }
45              ],
46              "outputHashing": "all"
47          },
48  ✓          "development": {
49              "optimization": false,
50              "extractLicenses": false,
51              "sourceMap": true
52          }
53      },
54      "defaultConfiguration": "production"
55  },
```




```
56     "serve": {
57       "builder": "@angular-devkit/build-angular:dev-server",
58       "configurations": {
59         "production": {
60           "buildTarget": "purchaseapp:build:production"
61         },
62         "development": {
63           "buildTarget": "purchaseapp:build:development"
64         }
65       },
66       "defaultConfiguration": "development"
67     },
68     "extract-i18n": {
69       "builder": "@angular-devkit/build-angular:extract-i18n",
70       "options": {
71         "buildTarget": "purchaseapp:build"
72       }
73     },
74     "test": {
75       "builder": "@angular-devkit/build-angular:karma",
76       "options": {
77         "polyfills": [
78           "zone.js",
79           "zone.js/testing"
80         ],
81         "tsConfig": "tsconfig.spec.json",
82         "assets": [
83           "src/favicon.ico",
84           "src/assets"
85         ],
86         "styles": [
87           "src/styles.css"
88         ],
89         "scripts": []
90       }
91     }
92   },
93   "cli": {
94     "analytics": false
95   }
96 }
97
98 }
```

## Розгорнутий на Firebase застосунок ShoppingList

<https://adamovis-01laba1-1-894e7.web.app/>

adamovis-01laba1-1-894e7.web.app

 Google Диск

# Shopping list

abuba

1000000

Додати


Предмет	Ціна	Куплено
Хліб	15.9	<input type="checkbox"/>
Вершкове масло 60		<input type="checkbox"/>
Картопля	22.6	<input checked="" type="checkbox"/>
Сир	310	<input type="checkbox"/>
abuba	1000000	<input checked="" type="checkbox"/>

## Частина 2: Прив'язка даних

### Інтерполяція

Прив'язка DOM до значення компонента (одностороння). У подвійних фігурних дужках вказується вираз, до якого йде прив'язка: {{вираз}}.

```
template: `
  <p>Ім'я: {{ name }}</p>
  <p>Вік: {{ age }}</p>`,
export class AppComponent {
  name = 'Tom';
  age = 25;
}
```



Ім'я: Tom

Вік: 25

### Прив'язка властивостей елементів HTML

Прив'язка властивості елемента DOM до значення компонента (одностороння).

```
<!-- Exercise 2 -->
<input type="text" [value]="name" />
<input type="text" [value]="age" />
<p [textContent]="name"></p>
export class AppComponent {
  name = 'Tom';
  age = 25;
}
```

Tom	25
-----	----

Tom

## Прив'язка до атрибуту

Зазвичай подібна прив'язка застосовується до атрибутів елементів `aria`, `svg` та `table`.

Наприклад, атрибут `colspan`, який поєднує стовпці таблиці, не має відповідної властивості.

І в цьому випадку ми можемо застосовувати прив'язку до атрибутів.

```
<!-- Exercise 3 -->
<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;
  // Exercise 3
  colspan = 2;
}
```

One-Two	
Three	Four
Five	Six

## Прив'язка до події

Прив'язка методу компонента до події DOM (генерація події DOM викликає метод на компоненті) (одностороння).

```
<!-- Exercise 4 -->
<p>Кількість кліків {{ count }}</p>
<button (click)="increase()">Click</button>
<p>Кількість кліків {{ count_2 }}</p>
<button (click)="increase_2($event)">Click</button>

export class AppComponent {
  name = 'Tom';
  age = 25;

  // Exercise 3
  colspan = 2;

  // Exercise 4
  count: number = 0;
  count_2: number = 0;

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

  increase_2($event: any): void {
    this.count_2++;
    console.log($event);
  }
}
```

Кількість кліків 0

Click

Кількість кліків 0

Click

Кількість кліків 2

Click

Кількість кліків 3

Click



## Двостороння прив'язка

Елемент DOM прив'язаний до значення на компоненті, зміни на одному кінці прив'язки відразу призводять до змін на іншому кінці. У цьому випадку застосовується модель. Як правило, двостороння прив'язка застосовується під час роботи з елементами введення, наприклад, елементами типу `input`

```
<!-- Exercise 5 -->
<p>Привіт {{ name }}</p>
<input type="text" [(ngModel)]="name" /> <br /><br />
<input type="text" [(ngModel)]="name" />` ,
```

```
lab1 > binding1 > src > app > TS app.module.ts > ...
```

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

```
template: `
```

```
<p>Ім'я: {{ name }}</p>
```

Привіт Abuba

## Прив'язка до класів CSS

Прив'язка до класу CSS має таку форму:

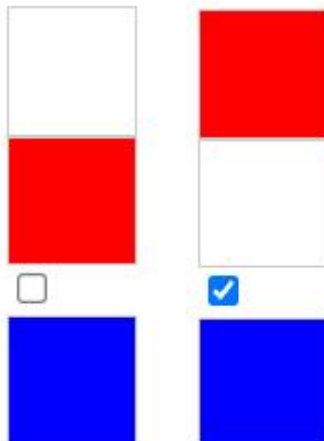
`[class.ім'я_класу]="true/false"`

Після префіксу `class` через точку вказується ім'я класу, яке хочемо додати чи видалити. Причому прив'язка йде до логічного значення. Якщо дорівнює `true`, то клас застосовується, якщо `false` - клас не застосовується.

```
template: `
  <!-- exercise 6 -->
  <div [class.isredbox]="isRed"></div>
  <div [class.isredbox]="!isRed"></div>
  <input type="checkbox" [(ngModel)]="isRed" />
  <div [class]="blue"></div>
```

```
styles: [
  `
    /*exercise 6*/
    div {
      width: 50px;
      height: 50px;
      border: 1px solid #ccc;
    }
    .isredbox {
      background-color: red;
    }
    .isbluebox {
      background-color: blue;
    }
  `,
],
```

```
export class AppComponent {
  // exercise 6
  ⚡title = 'binding2';
  isRed = false;
  blue = 'isbluebox';
}
```



## Прив'язка стилів

Прив'язка стилів має наступний синтаксис:

`[style.стильова_властивість]="вираз ? А : В"`

Після префіксу `style` через точку йде назва властивості стилю. Як значення передається деякий вираз: якщо воно повертає `true`, то стильовій властивості надається значення А; якщо воно повертає `false`, то стильовій властивості присвоюється значення В.

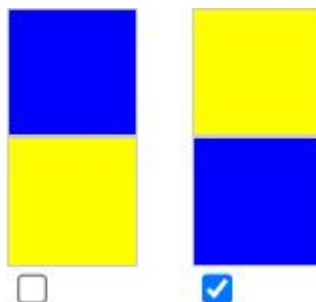
```
<!-- exercise 7 -->
<div [style.backgroundColor]="isyellow ? 'yellow' : 'blue'"></div>
<div [style.background-color]="!isyellow ? 'yellow' : 'blue'"></div>
<input type="checkbox" [(ngModel)]="isyellow" />
```

```
export class AppComponent {
  // exercise 6
  title = 'binding2';
  isRed = false;
  blue = 'isbluebox';

  // exercise 7
  isyellow = false;
}
```

```
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 {}
```



## Розгорнутий на Firebase застосунок Binding2

<https://adamovis-01laba1-2.web.app/>

