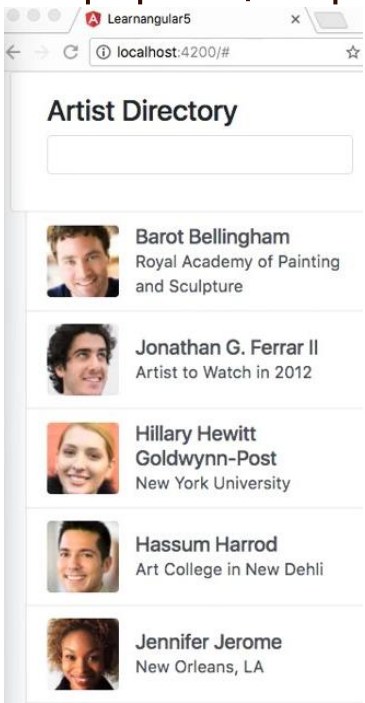
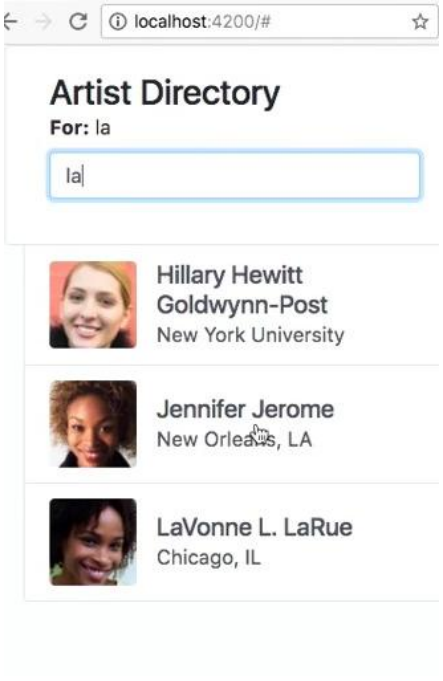
Курс "Програмування Інтернет"

Лабораторна робота №10 "Фреймворк Angular"

Завдання. За допомогою фреймворку Angular розробити сайт з інформацією про артистів. Текстова інформація задається в файлі формату JSON. Фото знаходяться в каталозі images. При наборі тексту в полі вводу здійснюється пошук і виводяться знайдені елементи. При виділенні елементу списку виводиться детальна інформація про артиста.





Хід роботи

1. Інсталюйте Node.js (https://nodejs.org/)
2. Відкрийте термінал (cmd.exe), перейдіть в каталог проектів (наприклад AngularProjects).
3. Інсталюйте Angular CLI
   - npm install –g @angular/cli
4. Інсталюйте Visual Studio Code (https://code.visualstudio.com/)
5. Згенеруйте систему директорій проекту
   - ng new learnangular5
6. Перейдіть в каталог проекту: cd learnangular5

7. Запустіть Visual Studio Code, відкрийте директорію проекту (learnangular5), перегляньте згенеровані файли.
8. Зробіть браузер Google Chrome браузером за замовчуванням.
9. Запустіть з консолі сервер розробника: ng serve –o . Ви побачите в браузері ваш сайт.

10. Додайте до проекту bootstrap та jQuery:
    - npm install jquery --save-dev
    - npm install bootstrap --save-dev
    - В папці node_modules з'являться папки bootstrap і jquery.
    - Внесіть зміни в файл .angular-cli.json :

```
"prefix": "app",
"styles": [
  "../node_modules/bootstrap/dist/css/bootstrap.css",
  "styles.css"
],
"scripts": [
  "../node_modules/jquery/dist/jquery.js",
  "../node_modules/bootstrap/dist/js/bootstrap.bundle.js"
],
```

11. Внесіть зміни в файл src/app/index.html :

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Learnangular5</title>

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <div class="container">
    <app-root></app-root>
  </div>
</body>
</html>
```

12. Запустіть сервер: ng serve -o . Ви побачите деякі зміни в форматуванні.

13. Поекспериментуйте з шаблоном компонента. Відкрийте файл app.component.ts . Внесіть зміни в шаблон (зверніть увагу на зворотні апострофи):

```
@Component({
  selector: 'app-root',
  template: `
    <h1>Artist Directory</h1>
    <p>Hello</p>
  `,
  styleUrls: ['./app.component.css']
})
```

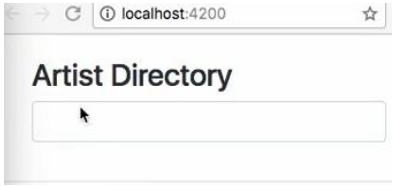
Ви побачите:

Візьміть фрагмент коду (file1.txt)

```
<div class="row justify-content-center">
  <div class="card mt-sm-3 col-md-8">
    <div class="card-body">
      <h3 class="mb-0">Artist Directory</h3>
      <div class="form-group">
        <div class="form-label"></div>
        <input class="form-control mt-2" type="text">
        </div><!-- form-group -->
      </div><!-- card-body -->
    </div><!-- card -->
</div><!-- row -->
```

```
@Component({
  selector: 'app-root',
  template: `
    <div class="row justify-content-center">
    <div class="card mt-sm-3 col-md-8">
      <div class="card-body">
        <h3 class="mb-0">Artist Directory</h3>
        <div class="form-group">
          <div class="form-label"></div>
          <input class="form-control mt-2"
          type="text">
        </div><!-- form-group -->
      </div><!-- card-body -->
    </div><!-- card -->
  </div><!-- row -->
  `,
```

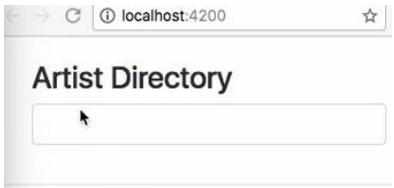
і вставте в якості шаблону.



В браузері ви побачите:

Поверніть файлу app.component.ts оригінальний вигляд, а фрагмент коду file1.txt перенесіть в файл app.component.html, видаливши його попередній вміст. Результат в браузері буде без змін.

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

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



14. Видаліть файл app.component.css (він пустий) і підкорегуйте файл app.component.ts

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  title = 'my app';
}
```

15. Видаліть також файл для тестів (app.component.spec.ts).

16. Внесіть невеличкі зміни в файл index.html

```
<body class="bg-light">
  <div class="container">
    <app-root></app-root>
  </div>
</body>
```

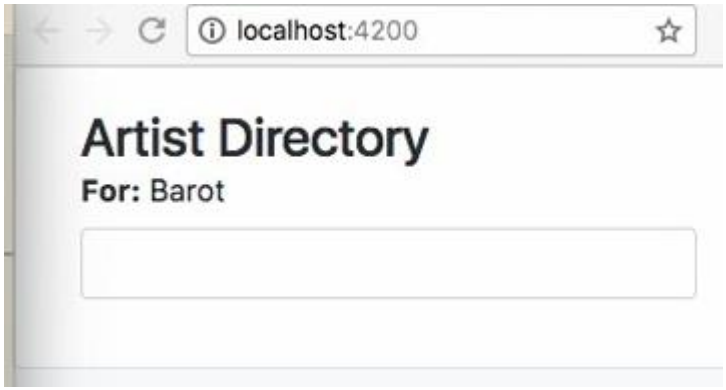
17. Поступово перейдемо до зв'язування даних. Введіть в компонент змінну query і ініціалізуйте її в конструкторі.

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  query: string;

  constructor() {
    this.query = 'Barot';
  }
}
```

18. Зв’яжіть змінну query з шаблоном (файл app.component.html)

```
<div class="row justify-content-center">
  <div class="card mt-sm-3 col-md-8">
    <div class="card-body">
      <h3 class="mb-0">Artist Directory</h3>
      <div class="form-group">
        <div
        class="form-label"><strong>For:</strong>
          {{ query }}
        </div>
        <input class="form-control mt-2"
        type="text">
      </div><!-- form-group -->
    </div><!-- card-body -->
  </div><!-- card -->
</div><!-- row -->
```

Результат в браузері буде таким:



19. Введіть в компонент дані про артистів. Для цього введіть в компонент змінну artists типу object і ініціалізуйте цю змінну в конструкторі

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  query: string;
  artists: object;

  constructor() {
    this.query = 'Barot';
    this.artists =
  }
}
```

Після знаку присвоєння треба вставити дані з файлу data.json, представленого на наступних двох слайдах.

Файл data.json


```
[
 {
  "name":"Barot Bellingham",
  "shortname":"Barot_Bellingham",
  "reknown":"Royal Academy of Painting and Sculpture",
  "bio":"Barot has just finished his final year at The Royal Academy of Painting and Sculpture, where he excelled in glass etching paintings and portraiture. Hailed as one of the most diverse artists of his generation, Barot is equally as skilled with watercolors as he is with oils, and is just as well-balanced in different subject areas. Barot's collection entitled \"The Un-Collection\" will adorn the walls of Gilbert Hall, depicting his range of skills and sensibilities - all of them, uniquely Barot, yet undeniably different"
 },
 {
  "name":"Jonathan G. Ferrar II",
  "shortname":"Jonathan_Ferrar",
  "reknown":"Artist to Watch in 2012",
  "bio":"The Artist to Watch in 2012 by the London Review, Johnathan has already sold one of the highest priced-commissions paid to an art student, ever on record. The piece, entitled Gratitude Resort, a work in oil and mixed media, was sold for $750,000 and Jonathan donated all the proceeds to Art for Peace, an organization that provides college art scholarships for creative children in developing nations"
 },
 {
  "name":"Hillary Hewitt Goldwynn-Post",
  "shortname":"Hillary_Goldwynn",
  "reknown":"New York University",
  "bio":"Hillary is a sophomore art sculpture student at New York University, and has already won all the major international prizes for new sculptors, including the Divinity Circle, the International Sculptor's Medal, and the Academy of Paris Award. Hillary's CAC exhibit features 25 abstract watercolor paintings that contain only water images including waves, deep sea, and river."
 },
 {
  "name":"Hassum Harrod",
  "shortname":"Hassum_Harrod",
  "reknown":"Art College in New Dehli",
  "bio":"The Art College in New Dehli has sponsored Hassum on scholarship for his entire undergraduate career at the university, seeing great promise in his contemporary paintings of landscapes - that use equal parts muted and vibrant tones, and are almost a contradiction in art. Hassum will be speaking on \"The use and absence of color in modern art\" during Thursday's agenda."
 },
 {
  "name":"Jennifer Jerome",
  "shortname":"Jennifer_Jerome",
  "reknown":"New Orleans, LA",
  "bio":"A native of New Orleans, much of Jennifer's work has centered around abstract images that depict flooding and rebuilding, having grown up as a teenager in the post-flood years. Despite the sadness of devastation and lives lost, Jennifer's work also depicts the hope and togetherness of a community that has persevered. Jennifer's exhibit will be discussed during Tuesday's Water in Art theme."
 },
```

```
{
  "name":"LaVonne L. LaRue",
  "shortname":"LaVonne_LaRue",
  "reknown":"Chicago, IL",
  "bio":"LaVonne's giant-sized paintings all around Chicago tell the story of love, nature, and conservation - themes that are central to her heart. LaVonne will share her love and skill of graffiti art on Monday's schedule, as she starts the painting of a 20-foot high wall in the Rousseau Room of Hotel Contempo in front of a standing-room only audience in Art in Unexpected Places."
 },
 {
  "name":"Constance Olivia Smith",
  "shortname":"Constance_Smith",
  "reknown":"Fullerton-Brighton-Norwell Award",
  "bio":"Constance received the Fullerton-Brighton-Norwell Award for Modern Art for her mixed-media image of a tree of life, with jewel-adorned branches depicting the arms of humanity, and precious gemstone-decorated leaves representing the spouting buds of togetherness. The daughter of a New York jeweler, Constance has been salvaging the discarded remnants of her father's jewelry-making since she was five years old, and won the New York State Fair grand prize at the age of 8 years old for a gem-adorned painting of the Manhattan Bridge."
 },
 {
  "name":"Riley Rudolph Rewington",
  "shortname":"Riley_Rewington",
  "reknown":"Roux Academy of Art, Media, and Design",
  "bio":"A first-year student at the Roux Academy of Art, Media, and Design, Riley is already changing the face of modern art at the university. Riley's exquisite abstract pieces have no intention of ever being understood, but instead beg the viewer to dream, create, pretend, and envision with their mind's eye. Riley will be speaking on the \"Art of Abstract\" during Thursday's schedule"
 },
 {
  "name":"Xhou Ta",
  "shortname":"Xhou_Ta",
  "reknown":"China International Art University",
  "bio":"A senior at the China International Art University, Xhou has become well-known for his miniature sculptures, often the size of a rice granule, that are displayed by rear projection of microscope images on canvas. Xhou will discuss the art and science behind his incredibly detailed works of art."
}
]
```

Після вставки конструктор буде виглядати так:

```
constructor() {
  this.query = 'Barot';
  this.artists = [
    {
      "name":"Barot Bellingham",
      "shortname":"Barot_Bellingham",
      "reknown":"Royal Academy of Painting and
      Sculpture",
      "bio":"Barot has just finished his final
      year at The Royal Academy of Painting and
      Sculpture, where he excelled in glass
      etching paintings and portraiture. Hailed
      as one of the most diverse artists of his
      generation, Barot is equally as skilled
```

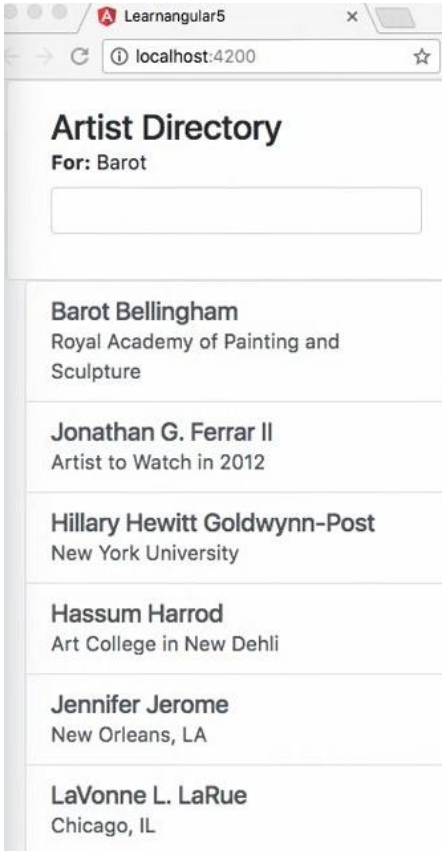
20. Відредагуйте файл app.component.html (треба відобразити список артистів):
    - Першим рядком вставте

    - Останнім рядком вставте
      <!-- position -->
    - Скопіюйте вміст файлу list-container.html (на наступному слайді) і вставте перед останнім рядком файлу app.component.html .
    - В браузері ви побачите список артистів.

Файл list-container.html

```
<div class="position-absolute container" style="left: 0; z-index:10">
  <div class="row justify-content-center">
    <div class="col-sm-10 col-md-8 col-lg-6 col-xl-5">
      <div class="list-group">
        <a href="#" class="list-group-item list-group-item-action flex-column align-
    items-start“
            *ngFor=“let artist of artists”>
          <div class="media">
            <div class="media-body align-self-center">
              <h5 class="m-0">{{ artist.name }}</h5>
               {{ artist.reknown }}
            </div><!-- media body -->
          </div><!-- media -->
        </a><!-- link -->
      </div><!-- list-group -->
    </div><!-- col -->
  </div><!-- row -->
</div><!-- container -->
```

В браузері ви побачите



В шаблоні була використана директива *ngFor для перебору списку артистів і зв'язування з властивостями artist.name та artist.reknown

21. Змініть стиль відображення компонента в самому компоненті (файл app.component.ts)

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styles: [
    `
      .list-group-item:first-child {
        border-top-left-radius: 0;
        border-top-right-radius: 0;
        border-top: 0;
      }
    `
  ]
})
```

Відображення в браузері трошки зміниться.

22. Наш компонент повинен реагувати на події. Нехай при кліку на конкретному артисті в списку, ім'я артиста буде виводитись в змінну query (і відображатися після For:).

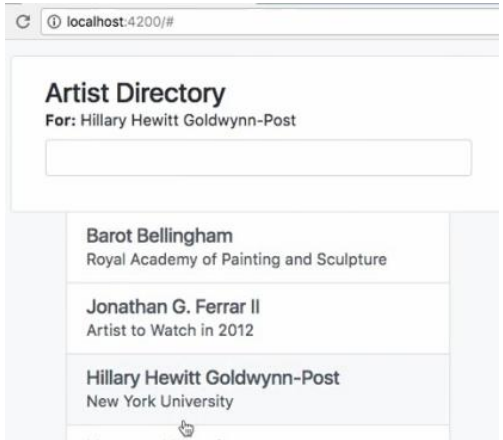
Щоб це реалізувати введіть в компонент метод showArtist

```
export class AppComponent {
  query: string;
  artists: object;

  showArtist(item) {
    this.query = item.name;
  }

  constructor() {
    this.query = 'Barot';
```

а в шаблон компонента (файл app.component.html) введіть подію (click)

```
align-items-start"
  *ngFor="let artist of artists"
  (click)="showArtist(artist)">
    <div class="media">
```

Після цього в браузері клік на артисті в списку призведе виведення імені артиста в мітці над полем введення:



23. Відмовтесь від початкової ініціалізації змінної query рядком ‘Barot’. В конструкторі напишіть this.query = ‘’;

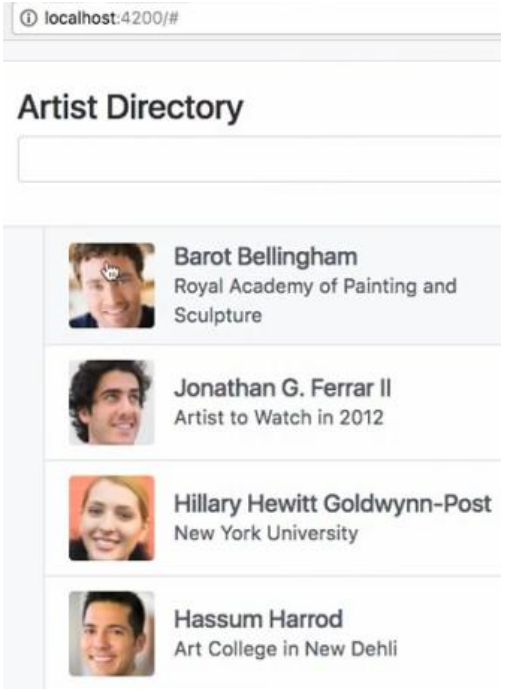
    А в шаблон компонента (файл app.component.html) введіть директиву *ngIf

```
<div class="form-label"
*ngIf="query"><strong>For:</strong>
  {{ query }}
</div>
```

    В результаті в браузері зникне початкове виведення значення змінної query, а при кліку з’явиться ім’я артиста.

24. Тепер перейдемо до розміщення фото поряд з інформацією про артиста в списку. Перемістіть в папку assets папку images з каталогу цієї лабораторної роботи (для кожного артиста там є два зображення, наприклад Barot_Bellingham.jpg і Barot_Bellingham_tn.jpg). Далі перейдіть в шаблон відображення компоненту (файл app.component.html) і наберіть необхідний код:

```
*ngFor="let artist of artists"
(click)="showArtist(artist)">
  <div class="media">
    <img class="mr-3 rounded align-self-center img-fluid"
      style="width:70px;" src="{{ './assets/images/' + artist.shortname +
      '_tn.jpg'}}" alt="{{ 'Photo of ' + artist.name }}">
    <div class="media-body align-self-center">
      <h5 class="m-0">{{ artist.name }}</h5>
      {{ artist.reknown }}
    </div><!-- media body -->
```



В браузері повинні з’явитись відповідні фото.

Але ми можемо мати потенційну проблему, підключивши фото таким традиційним способом. Якщо ви подивитесь на файл index.html, то помітите, що багато скриптових тегів можуть бути підключені в кінці файлу після тегу body. Це може призвести до того, що інколи ваш додаток буде завантажуватись до того, як зображення будуть прочитані з файлу даних. Особливо коли зображення будуть великі.

Тому трохи безпечніше буде використати тут властивості Angular (properties). Властивості поміщають в квадратні дужки. Ви можете замінити довільний атрибут тегу HTML на властивість, прив'язану до вашого компонента.

Поверніться в шаблон вашого компонента (файл app.component.html) і замість використання виразу src="…", застосуйте синтаксис властивостей. Атрибут src замініть на властивість, обгорнувши src в квадратні дужки.

```
<img class="mr-3 rounded align-self-center img-fluid"
  style="width:70px;" [src]="'./assets/images/' + artist.shortname + '_tn.jpg'"
  alt="{{ 'Photo of ' + artist.name }}">
```

Результат в браузері буде таким же, як на попередньому слайді.

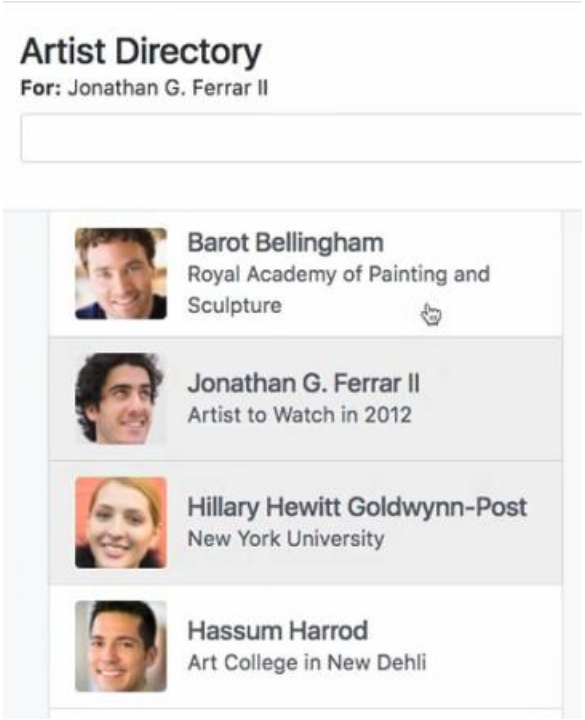
25. Треба ще додати наступну функціональність. Нехай при кліку на елементі списку змінюється колір фону. Це можна реалізувати, ввівши нову властивість style.backgroundColor:

```
*ngFor="let artist of artists"
(click)="showArtist(artist)"
[style.backgroundColor]="artist.highlight ? '#EEE' : '#FFF'">
  <div class="media">
```

і скорегувавши метод showArtist:

```
showArtist(item) {
  this.query = item.name;
  item.highlight = !item.highlight;
}
```

Результат в браузері буде виглядати так:



26. Ви вже знаєте, як пов'язати подію з компонентом, а також як використати властивості для модифікації HTML-елементів.

Angular може забезпечити об'єднання цих двох можливостей одночасно. Ця техніка відома під назвою Two-way Data Binding. Вона особливо корисна для елементів форми і існує два різні типи нотацій для різних типів полів. Перша форма – [(ngModel)] - дозволяє слідкувати за полями вводу (input field). Друга форма – [(ngControl)] - призначена для полів інших типів. Розглянемо, як це працює.

Давайте переробимо в шаблоні наше звичайне поле вводу згідно нотації ngModel і пов'яжемо його з нашою змінною query (файл app.component.html).

```
<input class="form-control mt-2" type="text"
  [(ngModel)]="query">
```

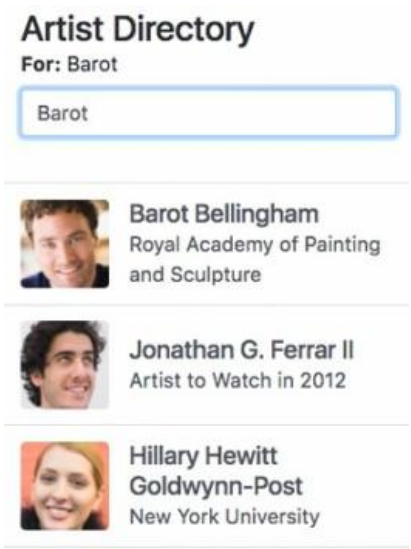
Ми також повинні додати ще один імпорт в файлі app.module.ts

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

і додати новий модуль в секції imports

```
imports: [
  BrowserModule, FormsModule
],
```

Після цього все буде працювати.
Таким чином ми продемонстрували, що можемо відслідковувати зміни в полі вводу і відповідно модифікувати значення змінної query.
Це дуже корисно, і це те, що ми реально потребуємо, працюючи з елементами форми.



27. Для того, щоб завантажити зовнішні дані ми можемо використати так звані методи життєвого циклу (lifecycle methods). Вони дозволяють виконати певну задачу в певний момент виконання додатку. Ці методи в Angular мають назву onInit. Вони виконуються після побудови компонента і потребують додаткового імпорту. Також є потреба працювати з HTTP verbs (наприклад, get) для отримання інформації, тому ми додамо і цю можливість. Для того, щоб виконати це, нам потрібно підписатися (subscribe) на так звані observable, які є новим паттерном в JavaScript. Вони надають можливість чекати появи деякої події і викликати callback-функцію при її появі.
В нашому випадку подією є отримання даних з зовнішнього документу (файлу). Тепер подивимось як це працює.

Поверніться в наш компонент (файл app.component.ts) і вилучіть (Ctrl-X) з конструктора присвоєння даних об'єкту artists, як ми це робили в пункті 19 (після вилучення правої частини присвоєння допишіть =[];)

```
this.artists = [];
```

Будемо ці дані завантажувати з файлу. Клікніть правою кнопкою по каталогу assets, виберіть New File, дайте ім'я data.json, відкрийте файл і скопіюйте в нього вилучені дані.

Відкрийте файл app.module.ts і допишіть 4-м рядком

```
import { HttpClientModule } from '@angular/common/http';
```

Додайте також імпортований модуль в секцію imports

```
imports: [
  BrowserModule, FormsModule, HttpClientModule
```

Поверніться в файл компонента (app.component.ts) і допишіть 2-м рядком

```
import { HttpClient } from '@angular/common/http';
```

Змініть також заголовок класу компонента

```
export class AppComponent implements OnInit {
```

Змініть також перший рядок цього файлу

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

Змініть також заголовок конструктора

```
constructor( private http: HttpClient ) {
```

А після тіла конструктора додайте метод ngOnInit

```
30
31    ngOnInit(): void {
32      this.http.get<Object>('../assets/data.json').subscribe(
        data => {
33        this.artists = data;
34      })
35    }
36
```

Якраз у цьому методі відбувається завантаження даних з файлу, підписка на подію (кінець отримання даних) і задається анонімна функція зворотного виклику, яка передає ці дані в об'єкт artists.

Якщо все зроблено правильно, то інформація про артистів вже буде зчитуватись з зовнішнього файлу.

28. При розробці регулярних застосунків на Angular краще створювати більше ніж один компонент для того, щоб розділити роботу додатку на частини. Це зменшує складність і спрощує оновлення застосунку. Отже, давайте розглянемо як це працює.

    Додаткові компоненти можна створювати вручну, але краще скористатись для цього засобами Angular CLI, який автоматично прописує всі необхідні зв'язки.

Для організації зв'язку між компонентами ми будемо використовувати властивості. Ми також повинні додати Inputs/imports для наших sub-компонентів.

Отже, в вашому терміналі подайте команду створення sub-компонента artist-items :

ng generate component artist-items

В результаті в каталозі app нашого додатку з'явиться каталог artist-items з чотирма файлами. Видаліть два з них – файл стилів (artist-items.component.css) і файл тестів (artist-items.component.spec.ts).

Відредагуйте файл artist-items.component.ts видаливши рядок

```
styleUrls: ['./artist-items.component.css']
```

Уважно вивчіть цей файл, зверніть увагу на автоматично згенерований імпорт на початку файла, на селектор app-artist-items для впровадження нашого sub-компонента в HTML файл. Автоматично згенерований також сам клас нашого sub-компонента з конструктором та методом ngOnInit.

Можете також перевірити як оновився файл app.module.ts. Наш sub-компонент автоматично включено в секції declarations та imports модуля.

Перейдіть в шаблон нашого основного компонента (файл app.component.html). Вставте тег нашого sub-компонента до секції media

```
<app-artist-items></app-artist-items>
  <div class="media">
```

Далі вилучіть (cut, Ctrl-X) всю секцію media (з 26-го по 33-й рядки) в буфер і вставте в файл шаблону нашого sub-компонента (файл artist-items.component.html) взамін його попереднього вмісту.

Тепер кожний компонент має свою відповідальність, але компоненти ще не зовсім точно зв'язані між собою. Додамо цей зв'язок вручну.

Перейдіть в файл artist-items.component.ts і додайте в sub-компонент секцію inputs, яка показує передані з основного компонента дані. Передати треба змінну artist. Цю змінну можна потім використати в HTML файлі.

```
@Component({
  selector: 'app-artist-items',
  templateUrl: './artist-items.component.html',
  inputs: ['artist']
})
```

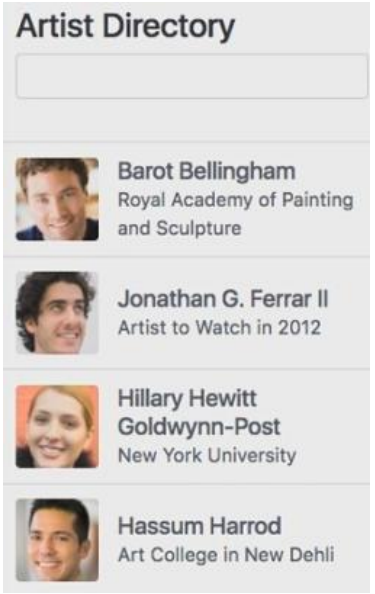
В скопійованій раніше секції media ця змінна artist вже використовується.

Перейдіть в файл app.component.html і скорегуйте рядок з тегом нашого sub-компонента, вставлений раніше.

```
<app-artist-items [artist]="artist" ></app-artist-items>
```

Тут зліва від знаку присвоєння стоїть властивість artist (яка фігурує в секції inputs) , а праворуч стоїть змінна циклу artist . Впевніться, що ви це розумієте.

Перейдіть в браузер, і впевніться, що все працює так, як і раніше. Тільки наш додаток вже складається з двох компонентів, які взаємодіють. Така декомпозиція в подальшому буде сприяти оновленню і тестуванню застосунку.



29. Щоб допомогти здійснити пошук нам знадобляться канали (pipes). Angular має декілька вбудованих каналів, але їх функціональність невелика, вони корисні, наприклад, для роботи з датами або перетворенням тексту. Тому для чогось більш суттєвого вам треба створювати свої власні канали.

    Ви можете згенерувати власний pipe за допомогою Angular CLI, що дає корисний шаблон для ваших перетворень.

Згенерувавши канал за допомогою CLI, ви отримаєте доступ до функції transform(pipeData, pipeModifier). Першим параметром тут передаються оригінальні дані, які ви хочете відфільтрувати. Другим передається параметр фільтрації.

Тепер давайте розглянемо як це працює.

В нашому застосунку ми хочемо набирати текст в полі вводу, а додаток повинен здійснювати пошук відповідно введеним даним і виводити знайдені елементи. Отже, ми повинні модифікувати початкове виведення списку артистів.

Поверніться до вашого терміналу і подайте команду, що створює канал: ng generate pipe search-artists

Ви побачите, що створено два файли, а файл app.module.ts оновлено. Видаліть файл тестів.

Вивчіть згенерований файл search-artists.pipe.ts. Тут імпортуються необхідні типи даних і створюється клас SearchArtistsPipe, що реалізує PipeTransform, дається заготовка методу transform.

Першим параметром цього метода в нашому випадку буде весь список артистів, а другим параметром – текст, набраний в полі вводу.

Сформуйте метод transform в наступному вигляді:

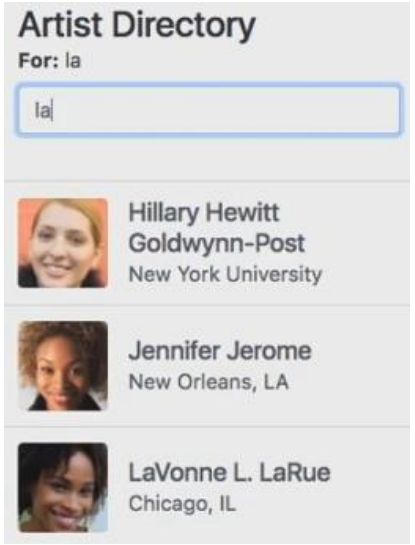
```
transform(pipeData, pipeModifier): any {
  return pipeData.filter(eachItem => {
    return (
      eachItem['name'].toLowerCase().includes(pipeModifier.toLowerCase()) ||
      eachItem['reknown'].toLowerCase().includes(pipeModifier.toLowerCase())
    )
  });
}
```

Пошук відбувається по полям name і reknown об'єктів JSON файлу.

Тепер внесіть зміни в файл шаблону app.component.html де перебираються елементи списку. Тут треба застосувати наш фільтр:

```
*ngFor="let artist of (artists |
searchArtists: query)"
```

Тепер при наборі тексту в полі вводу пошук буде працювати:



30. Нам залишилось виконати останнє завдання: виведення детальної інформації про артиста при кліку по елементі списку.

Поверніться до вашого терміналу і подайте команду для створення ще одного компонента:
ng generate component artist-details
В результаті в каталозі app нашого додатку з'явиться каталог artist-details з чотирма файлами. Видаліть два з них – файл стилів (artist-detais.component.css) і файл тестів (artist-details.component.spec.ts).
Відредагуйте файл artist-details.component.ts видаливши рядок

```
styleUrls: ['./artist-details.component.css']
```

Відкрийте файл artist-details.component.html і замініть його вміст на вміст файлу з наступного слайду.

А тепер ми повинні забезпечити взаємодію нашого основного компонента і sub-компонента. Перейдіть в файл artist-details.component.ts і скопіюйте в буфер ім'я селектора (app-artist-details) далі перейдіть в файл шаблона app.component.html і вставте тег селектора перед останнім рядком:

```
30    </div><!-- container -->
31
32    <app-artist-details  *ngIf="currentArtist"
33      [artist]="currentArtist"></app-artist-details>
34
35  </div><!-- position -->
```

Файл artist-details.component.html

```
<div class="container mt-3">
  <div class="row justify-content-center">
    <div class="col-sm-4 col-lg-3">
      <img class="img-fluid rounded"
   src="assets/images/{{artist.shortname}}.jpg" alt="{{artist.name}} photo">
    </div>
    <div class="col-sm-8 col-lg-4">
      <h2 class="mt-3 mt-sm-0 mb-0">{{artist.name}}</h2>
      <h4 class="mt-0">{{artist.reknown}}</h4>
      <p>{{artist.bio}}</p>
    </div>
  </div>
</div> <!--artistdetails-->
```

```
30  </div><!-- container -->
31
32  <app-artist-details  *ngIf="currentArtist"
33    [artist]="currentArtist"></app-artist-details>
34
35  </div><!-- position -->
```

Тут введена властивість artist, яка працює в файлі шаблона, а праворуч введена змінна currentArtist, яка пов'язана з кліком в списку.

Цю змінну треба прописати в компоненті. Тому перейдемо в наш головний компонент (файл app.component.ts) і допишемо рядок:

```
export class AppComponent implements OnInit {
  query: string;
  artists: object;
  currentArtist: object;
```

Також треба модифікувати в цьому ж файлі метод showArtist, який раніше викликався при кліку і змінював фон в списку. Тут також допишемо рядок:

```
showArtist(item) {
  this.query = item.name;
  item.highlight = !item.highlight;
  this.currentArtist = item;
}
```

Тепер передамо властивість artist в наш sub-компонент через секцію inputs. Відкрийте файл artist-details.component.ts і допишіть рядок:

```
@Component({
  selector: 'app-artist-details',
  templateUrl: './artist-details.component.html',
  inputs: ['artist']
```

А тепер внесіть зміни в файл шаблону головного компонента (файл app.component.html). Зробіть так, щоб група списку була показана тільки тоді, коли змінна query існує:

```
20        <div class="list-group" *ngIf="query">
```

В результаті список буде прихований до тих пір, поки не почнеться введення в полі вводу.

Також очистіть змінну query після кліку:

```
23            (click)="showArtist(artist); query=''"
```

Збережіть усі зміни. Перейдіть в браузер. Ви побачите, що список відсутній. При наборі тексту буде працювати пошук з виведенням списку знайдених елементів. А при кліку в списку буде з'являтись детальна інформація про артиста. Пошук можна продовжувати знову.