Для использования Angular на ПК должен быть установлен Node.js. Скачать и установить можно на оф сайте https://nodejs.org/uk/.

После того как пакетный менеджер npm установлен, переходим к установке Angular. В командной строке исполняем команду (установка займёт какое-то время):

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

Для создания проекта можно использовать хоть обычный блокнот, но для удобства будем использовать лёгкий и быстрый редактор кода Visual Studio Code, который уже преднастроен для нашей работы и понимает различные нужные нам синтаксисы «с коробки».

Откроем директорию, в которой будем создавать проект (в примере это D:\For_university\TRPZ_4\Lection_Exes>, но это не так и важно). Далее будем работать с терминала самого редактора. Для создания нового проекта Angular в терминале пропишем команду:

```
ng new Lec-Ex
```

, где Lec-Ex – название проекта

Сразу будет задано несколько вопросов. Не соглашаемся на строгую типизацию данных и добавления роутинга в проект. Роутинг в данной работе не понадобиться, но его можно потом подключить вручную. Но если включить строгую типизацию, проект не соберётся. Мы не будем использовать css-препроцессоры, поэтому выбираем чистый css. В итоге имеем:

```
D:\For_university\TRPZ_4\Ex>ng new Lec-Ex
? Do you want to enforce stricter type checking and stricter bundle budgets in the workspace?
  This setting helps improve maintainability and catch bugs ahead of time.
  For more information, see https://angular.io/strict No
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
```

После начнётся процесс создания. По завершению будет создана папка с названием проекта:

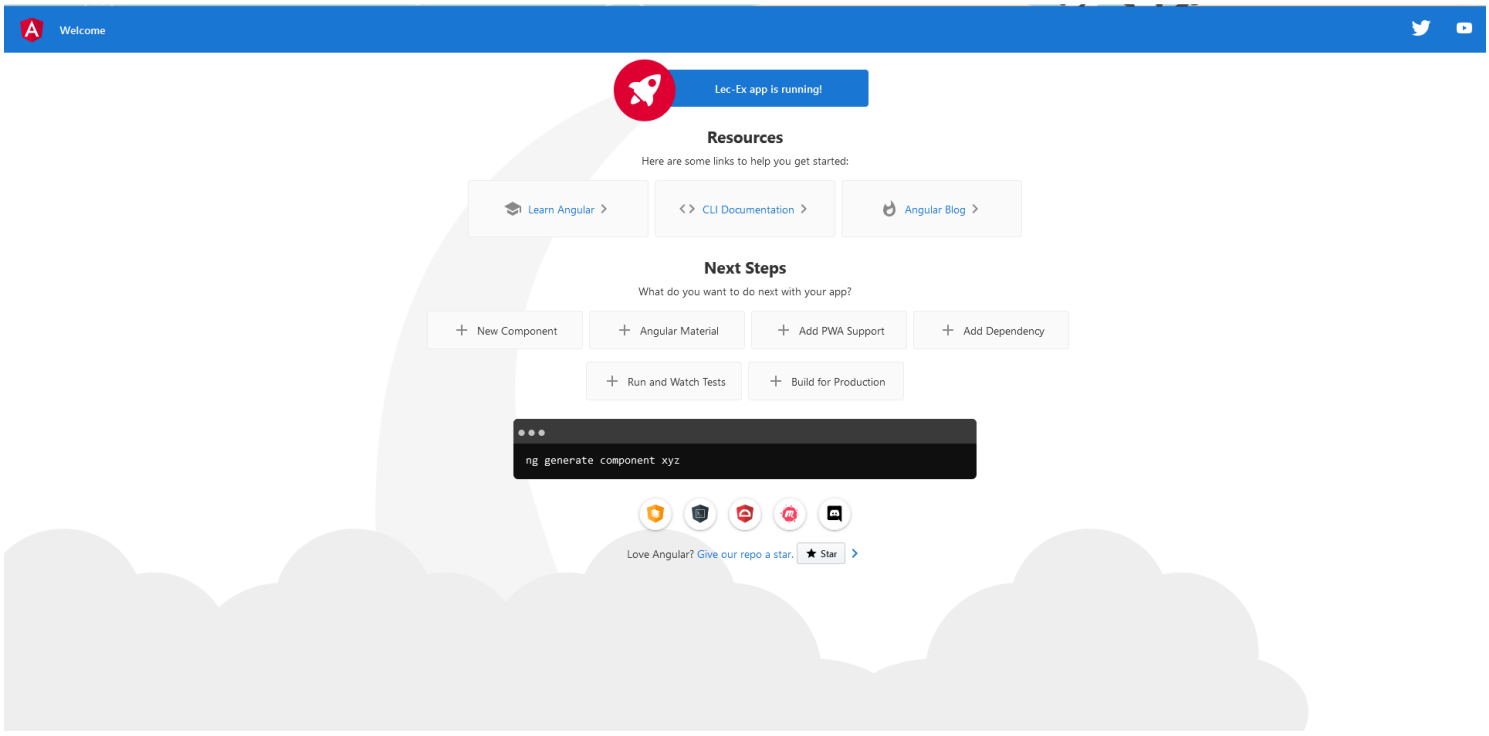
Перейдём в эту папку:

```
cd Lec-Ex
```

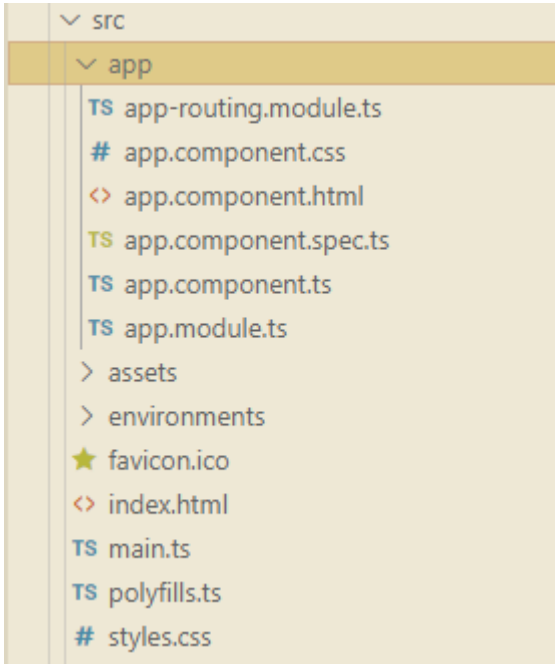
Убедиться, что всё установлено верно, можно запустив макет созданного проекта:

```
ng serve –open
```

Макет имеет следующий вид:



По умолчанию в проекте Angular создаётся папка Src, в которой будут находиться все исходники проекта. В данной папке сразу генерируется основной компонент проекта App:



Для каждого компонента проекта можно задавать свою разметку, стили (влияют только на этот компонент) и поведение. В компоненте App это `app.component.html`, `app.component.css` и `app.component.ts` соответственно.

Для начала в папке App Создадим два файла. `item.ts` с содержанием:

```
export class Item {
    static nextId = 0;

    static items: Item[] = [
      new Item(null,'Teapot','stout'),
      new Item(1, 'Lamp', 'bright'),
      new Item(2, 'Phone', 'slim' ),
      new Item(3, 'Television', 'vintage' ),
      new Item(4, 'Fishbowl')
    ];


    constructor(
      public id?: number,
      public name?: string,
      public feature?: string,
      public url?: string,
      public rate = 100,
      ) {
      this.id = id ? id : Item.nextId++;
    }

    clone(): Item {
      return Object.assign(new Item(), this);
    }
}
```

И файл `item-switch.component.ts` с содержанием:

```
import { Component, Input } from '@angular/core';
import { Item } from './item';

@Component({
  selector: 'app-stout-item',
  template: `I'm a little {{item.name}}, short and stout!`
})
export class StoutItemComponent {
  @Input() item: Item;
}


@Component({
  selector: 'app-best-item',
  template: `This is the brightest {{item.name}} in town.`
})
export class BestItemComponent {
  @Input() item: Item;
}
```

```
@Component({
  selector: 'app-device-item',
  template: `Which is the slimmest {{item.name}}?`
})
export class DeviceItemComponent {
  @Input() item: Item;
}

@Component({
  selector: 'app-lost-item',
  template: `Has anyone seen my {{item.name}}?`
})
export class LostItemComponent {
  @Input() item: Item;
}

@Component({
  selector: 'app-unknown-item',
  template: `{{message}}`
})
export class UnknownItemComponent {
  @Input() item: Item;
  get message() {
    return this.item && this.item.name ?
      `${this.item.name} is strange and mysterious.` :
      'A mystery wrapped in a fishbowl.';
  }
}

export const ItemSwitchComponents =
  [ StoutItemComponent, BestItemComponent, DeviceItemComponent, LostItemComponent, UnknownItemCo
mponent ];
```

В данном файле описано множество компонентов. Вообще приветствуется пользоваться правилом «Один компонент – одна папка». Но это просто компоненты для демонстрации работы директив, они маленькие и создание многих папок и файлов не целесообразно и приведёт к их излишеству.

Видим, что каждый класс начинается с директивы `@Component`. Его свойства следующие:

`selector` – название тега представляющего этот компонент в разметке,

`template` – содержание этого тега (может быть указано через ссылку на HTML файл как в компоненте App).

Создадим второй компонент item-detail, как принято. Для этого в папке App создадим папку item-detail, а в ней файлы `item-detail.component.ts` с содержанием:

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

import { Item } from '../item';

@Component({
  selector: 'app-item-detail',
  templateUrl: './item-detail.component.html'
})
export class ItemDetailComponent {

  @Input() item: Item;
  constructor() { }
}
```

И файл `item-detail.component.html` с содержанием:

```
<div>
    <span>{{item?.name}}</span>
</div>
```

BUILT-IN STRUCTURAL DIRECTIVES

ngIf

Директива `ngIf` позволяет удалить или, наоборот, добавить элемент при определенном условии. Для демонстрации работы содержание `app.component.ts` заменим на:

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {

  isSpecial = true;
  isActive = true;

  nullCustomer = null;
  currentCustomer = {
    name: 'Laura'
  };
```

```
  item: Item; // defined to demonstrate template context precedence
  items: Item[];

  currentItem: Item;

  itemsNoTrackByCount   = 0;
  itemsWithTrackByCount = 0;
  itemIdIncrement = 1;

  ngOnInit() {
    this.resetItems();
  }

  isActiveToggle() {
    this.isActive = !this.isActive;
  }

  giveNullCustomerValue() {
    this.nullCustomer = 'Kelly';
  }

  resetItems() {
    this.items = Item.items.map(item => item.clone());
    this.currentItem = this.items[0];
    this.item = this.currentItem;
  }

  resetList() {
    this.resetItems();
    this.itemsWithTrackByCount = 0;
    this.itemsNoTrackByCount = 0;
  }

  changeIds() {
    this.items.forEach(i => i.id += 1 * this.itemIdIncrement);
    this.itemsNoTrackByCount = ++this.itemsNoTrackByCount;
    this.itemsWithTrackByCount = ++this.itemsWithTrackByCount;
  }

  trackByItems(index: number, item: Item): number { return item.id; }
  trackById(index: number, item: any): number { return item.id; }
}
```

А в `app.component.html` добавим:

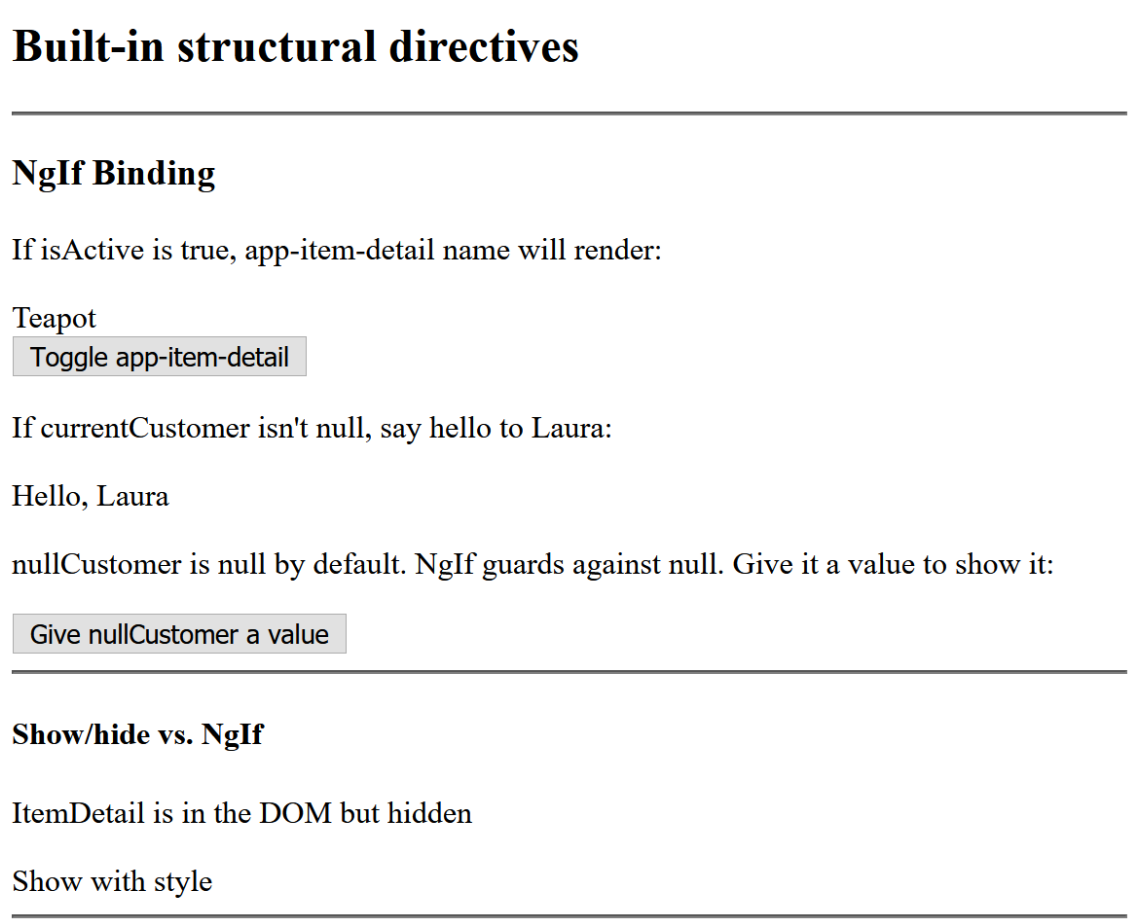
```
<h2>Built-in structural directives</h2>
<hr>
<h3 id="ngIf">NgIf Binding</h3>

<div>
  <p>If isActive is true, app-item-detail name will render:</p>
  <app-item-detail *ngIf="isActive" [item]="item"></app-item-detail>
  <button (click)="isActiveToggle()">Toggle app-item-detail</button>
</div>

<div>
  <p>If currentCustomer isn't null, say hello to Laura:</p>
  <div *ngIf="currentCustomer">Hello, {{currentCustomer.name}}</div>
  <p>nullCustomer is null by default. NgIf guards against null. Give it a value to show it:</p>
  <div *ngIf="nullCustomer">Hello, <span>{{nullCustomer}}</span></div>
  <button (click)="giveNullCustomerValue()">Give nullCustomer a value</button>
</div>

<hr>

<div>
  <h4>Show/hide vs. NgIf</h4>
  <p>ItemDetail is in the DOM but hidden</p>
  <app-item-detail [attr.hidden]="isSpecial" [item]="item"></app-item-detail>

  <div [style.display]="isSpecial ? 'block' : 'none'">Show with style</div>
  <div [style.display]="isSpecial ? 'none'  : 'block'">Hide with style</div>
</div>

<hr>
```

В первом div по нажатию кнопки вызывается функция `isActiveToggle()` которая изменяет состояние `isActive`. Условие рендеринга тега `app-item-detail` записывается как «`*ngIf="isActive"`». То есть при `isActive = true`, будет выводится название item.

Второй div демонстрирует возможность рендеринга при определённом объекте. Так currentCustomer не null и текст с её приветствием выводится. nullCustomer по умолчанию null и как видим, её текста нету. По нажатию кнопки вызовется функция, которая присвоит nullCustomer значение и текст появится на странице

Третий div демонстрирует разницу между использование директивы и скрытием элемента с помощью стилей. При последнем элемент скрыт, но виден в DOM-дереве, при первом же варианте элемент вообще не добавляется в него.

Страница примет вид:



NgFor

Директива `ngFor` позволяет перебрать в шаблоне элементы массива. В нашем примере для перебора массива items используем конструкцию вида «`*ngFor="let it of items"`», Если необходимо использовать индекс конструкция будет следующая «`*ngFor="let it of items; trackBy: trackByItems"`».

Итак, дополним `app.component.html`:

```
<h3 id="ngFor">NgFor Binding</h3>
<div class="box">
  <div *ngFor="let it of items">{{it.name}}</div>
</div>

<p>*ngFor with ItemDetailComponent element</p>
<div class="box">
  <app-item-detail *ngFor="let it of items" [item]="it"></app-item-detail>
</div>

<h4 id="ngFor-index">*ngFor with index</h4>
<p>with <i>semi-colon</i> separator</p>
<div class="box">
  <div *ngFor="let it of items; let i=index">{{i + 1}} - {{it.name}}</div>
```

```
</div>

<p>with <i>comma</i> separator</p>
<div class="box">
 <div *ngFor="let it of items, let i=index">{{i + 1}} - {{it.name}}</div>
</div>

<h4 id="ngFor-trackBy">*ngFor trackBy</h4>
<button (click)="resetList()">Reset items</button>
<button (click)="changeIds()">Change ids</button>

<p>with trackBy</p>
<div class="box">
  <div #withTrackBy *ngFor="let it of items; trackBy: trackByItems">({{it.id}}) {{it.name}}</div
>

  <div id="withTrackByCnt" *ngIf="itemsWithTrackByCount">
    Item DOM elements change #{{itemsWithTrackByCount}} with trackBy
  </div>
</div>

<hr>
```

На странице появится следующее:

NgFor Binding

Teapot
Lamp
Phone
Television
Fishbowl

*ngFor with ItemDetailComponent element

Teapot
Lamp
Phone
Television
Fishbowl

***ngFor with index**

1 - Teapot
2 - Lamp
3 - Phone
4 - Television
5 - Fishbowl

***ngFor trackBy**

Reset items | Change ids

with trackBy

(0) Teapot
(1) Lamp
(2) Phone
(3) Television
(4) Fishbowl

Первые два списка выводятся без индексов. Разница между ними в том, что первый список – это вывод названий элементов. Второй же – тоже вывод названий элементов, но с помощью элемента определённого в компоненте.

То есть для первого списка в DOM-дереве мы увидим набор div-ов:

```
<div class="box" _ngcontent-opl-c41="">
    <div _ngcontent-opl-c41="">Teapot</div>
    <div _ngcontent-opl-c41="">Lamp</div>
    <div _ngcontent-opl-c41="">Phone</div>
    <div _ngcontent-opl-c41="">Television</div>
    <div _ngcontent-opl-c41="">Fishbowl</div>
    <!--
    bindings={ "ng-reflect-ng-for-of": "[object
    Object" }
    -->
  </div>
```

Для второго же это набор <app-item-detail>:

```
<div class="box" _ngcontent-opl-c41="">
  <app-item-detail _ngcontent-opl-c41="" ng-reflect-item="
  [object Object]"> … </app-item-detail>
  <app-item-detail _ngcontent-opl-c41="" ng-reflect-item="
  [object Object]"> … </app-item-detail>
  <app-item-detail _ngcontent-opl-c41="" ng-reflect-item="
  [object Object]"> … </app-item-detail>
  <app-item-detail _ngcontent-opl-c41="" ng-reflect-item="
  [object Object]"> … </app-item-detail>
  <app-item-detail _ngcontent-opl-c41="" ng-reflect-item="
  [object Object]"> … </app-item-detail>
  <!--
  bindings={ "ng-reflect-ng-for-of": "[object Object],[objec
  Object" }
  -->
</div>
```

Третий список демонстрирует список с индексацией.

Рассмотрим четвёртый, последний, список. Кнопками над ним можно изменять его индексацию, что будет приводить к изменению элементов на странице. Использование `trackBy` не очевидно. Его цель помочь Angular отслеживать элементы, которые изменились, чтобы изменять только их. Без его использования будет заново отображаться всё DOM-дерево.

NgSwitch

С помощью директивы `ngSwitch` можно встроить в шаблон конструкцию switch..case и в зависимости от ее результата выполнения выводить тот или иной блок. Дополним `app.component.html`:

```
<h3>NgSwitch Binding</h3>

<p>Pick your favorite item</p>
<div>
 <label *ngFor="let i of items">
   <div><input type="radio" name="items" [(ngModel)]="currentItem" [value]="i">{{i.name}}
   </div>
 </label>
</div>

<div [ngSwitch]="currentItem.feature">
 <app-stout-item    *ngSwitchCase="'stout'"    [item]="currentItem"></app-stout-item>
 <app-device-item   *ngSwitchCase="'slim'"     [item]="currentItem"></app-device-item>
 <app-lost-item     *ngSwitchCase="'vintage'"  [item]="currentItem"></app-lost-item>
 <app-best-item     *ngSwitchCase="'bright'"   [item]="currentItem"></app-best-item>
 <div *ngSwitchCase="'bright'"> Are you as bright as {{currentItem.name}}?</div>
 <app-unknown-item  *ngSwitchDefault           [item]="currentItem"></app-unknown-item>
</div>

<br><br>
```

На странице добавится

NgSwitch Binding

Pick your favorite item

- (•) Teapot
- ( ) Lamp
- ( ) Phone
- ( ) Television
- ( ) Fishbowl

I'm a little Teapot, short and stout!

В первом div создаётся уже известен нам список, но с возможностью выбора элемента. Выбранный элемент сохраняется в переменную currentItem.

Во втором div продемонстрированно использование ngSwitch. Так в зависимости от выбранного currentItem будет выведен определённый элемент. Так выбирая разные элементы можем видеть, как текст внизу меняется.

И на конец для более опрятного вида страницы добавим в app.component.css стили:

```
button {
    font-size: 100%;
    margin: 0 2px;
}

div[ng-reflect-ng-switch], app-unknown-item {
  margin: .5rem 0;
  display: block;
}

#noTrackByCnt,
#withTrackByCnt {
    color: darkred;
    max-width: 450px;
    margin: 4px;
}

img {
    height: 100px;
}

.box {
    border: 1px solid black;
    padding: 6px;
    max-width: 450px;
}

.child-div {
    margin-left: 1em;
    font-weight: normal;
}

.context {
    margin-left: 1em;
}

.hidden {
    display: none;
}
```

```css
.parent-div {
    margin-top: 1em;
    font-weight: bold;
}

.course {
    font-weight: bold;
    font-size: x-large;
}

.helpful {
    color: red;
}

.saveable {
    color: limegreen;
}

.study,
.modified {
  font-family: "Brush Script MT", cursive;
  font-size: 2rem;
}

.toe {
    margin-left: 1em;
    font-style: italic;
}

.to-toc {
    margin-top: 10px;
    display: block;
}
```

Весь проект можно скачать по ссылке:

https://github.com/MadVitaliy/trpz_angular_directives_example

После скачивание с директории проекта в командной строке нужно выполнить

```
npm i
```

Для скачивания и установки всех необходимых зависимостей.