Angular Basics – Learning Journal

Hi there! I'm currently learning Angular by following the Angular Basics course from Simplilearn.
This document summarizes the key concepts I've studied so far.

❤️ What is Angular?

Angular is a powerful TypeScript-based framework for building Single Page Applications (SPA). Instead of loading multiple pages, a SPA loads a single page and dynamically updates content, providing a smoother and more app-like experience.

❤️ Key Features

- **DOM (Document Object Model)**
  Interface that treats HTML/XML as a tree structure to manipulate structure, style, and content.
- **TypeScript**
  Superset of JavaScript. Install globally:

  ```
  npm install -g typescript
  ```
- **Data Binding**
  Dynamically connects UI and application logic (e.g., forms, calculators).
- **Testing**
  Use the Jasmine testing framework for writing and running tests.

❤️ Angular Architecture

Model – View – Controller (MVC) structure enhances organization and scalability.

Components

The core building block of Angular apps.
- Created using @Component decorator
- Must be declared in a module (or be standalone)
- Use lifecycle hooks and templates
- Only one component per DOM element

Command:

```
ng generate component my-component --standalone
```

❤️ Angular Setup & HelloWorld Project

1. **Install Node.js**
   Download: https://nodejs.org

2. **Install Angular CLI**

   ```
   npm install -g @angular/cli
   ```
3. **Create a new project**

   ```
   ng new hello-world
   ```
4. **Start the app**

   ```
   cd hello-world
   ng serve
   ```

   Visit: http://localhost:4200

❤️ Useful VS Code Tip

PowerShell sometimes causes issues. You can switch the terminal:
1. Press Ctrl + Shift + P
2. Select Terminal: Select Default Profile
3. Choose Command Prompt or Git Bash
4. Open a new terminal with Ctrl + Shift + `

❤️ Rendering & Routing

- **SSR (Server-Side Rendering)**
  Improves SEO and loading time by rendering HTML on the server.
- **SSG (Static Site Generation)**
  Pre-renders static HTML files at build time — perfect for blogs and portfolios.
- **Server Routing & App Engine APIs**
  Server-side routing and experimental APIs (Developer Preview) for more advanced control.

❤️ Angular Files Overview

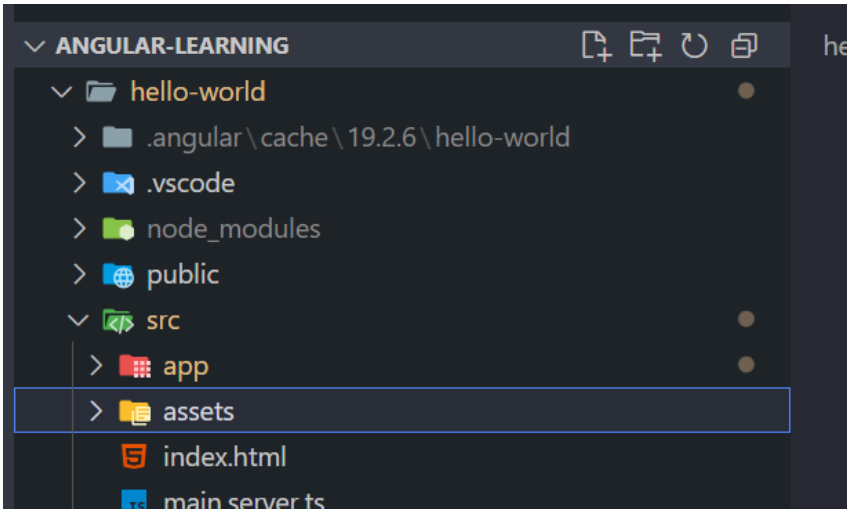
- `index.html` → Loads <app-root> component
- `main.ts` → Bootstraps the app
- `app.module.ts` → App configuration (if not using standalone)
- `app.component.ts` → Main component logic
- `app.component.html` → UI structure
- `app.component.css` → Styling

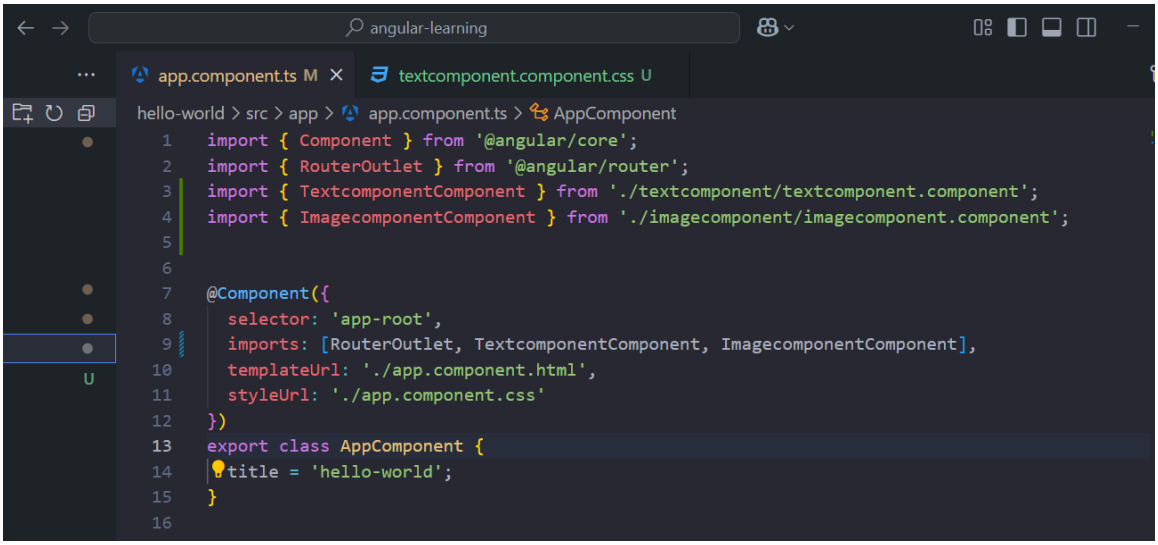
❤️ Styling & Assets

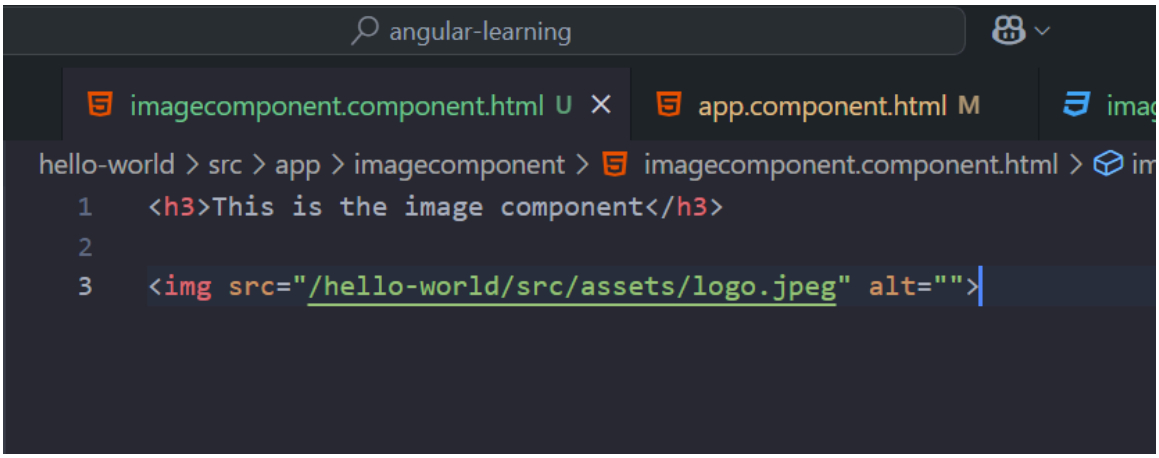
Place images inside the src/assets folder.
Make sure angular.json includes:

```
"assets": [
 "src/assets",
```

"src/favicon.ico"
]







app.component.ts M imagecomponent.component.html U

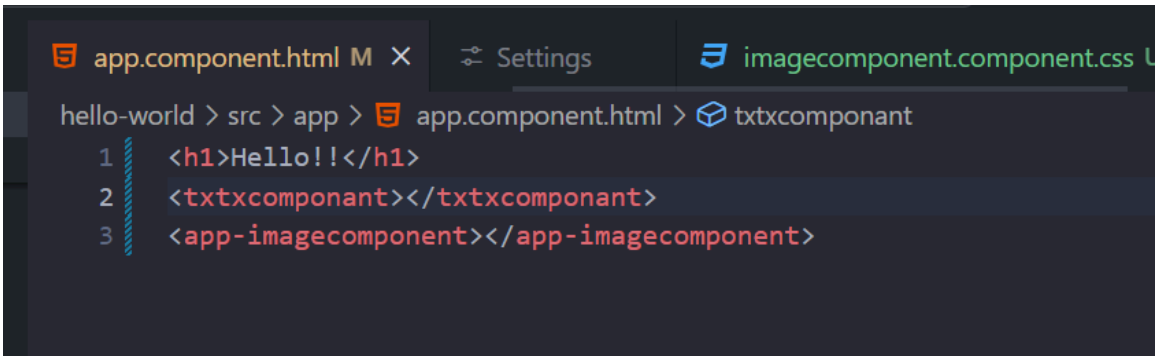
hello-world > src > app > app.component.html > app-imagecompone

```
1  <h1>Hello!!</h1>
2  <app-textcomponent></app-textcomponent>
3  <app-imagecomponent></app-imagecomponent>
```

❤️ Component Metadata

- Selector – The custom HTML tag for the component

- Template – Inline HTML (for small components)
- TemplateUrl – External HTML file (recommended)



- StyleUrls – External CSS file(s)
- Providers – Injected services

❤️ Angular Services & Dependency Injection (DI)

Use services to share logic and data between components.

```
@Injectable({ providedIn: 'root' })
export class MyService {
 getMessage() {
  return 'Hello from the service!';
 }
}
```

Injecting into a component:

```
constructor(private myService: MyService) {
```

```
  this.message = myService.getMessage();
}
```

✅ Loose coupling
✅ Easier to test
✅ More maintainable code

❤️ Animations

Angular animations bring elements to life:

```
   npm install @angular/animations
```

Example:

```
animations: [
  trigger('myAnimation', [
    state('open', style({ height: '200px', backgroundColor: 'lightgreen' })),
    state('closed', style({ height: '100px', backgroundColor: 'lightcoral' })),
    transition('open <=> closed', animate('0.3s ease-in-out'))
  ])
]
```

❤️ Practice Steps Summary

1. Generate a component

```
   ng g c textcomponent
```

2. Generate a service

```
   ng g s records
```

3. Use `standalone: true` for newer Angular versions (17+)

```
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [TextcomponentComponent],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

❤️ Practising Service and DI

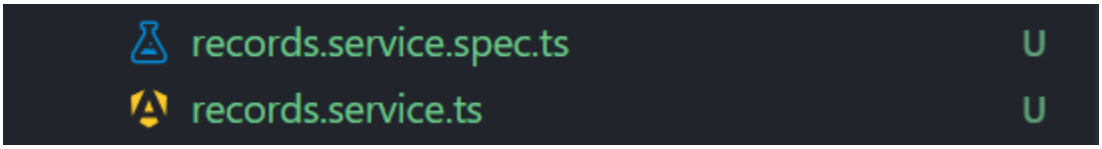
1- Create a component

ng g c textcomponent

```
C:\Users\naomi\OneDrive\Área de Trabalho\angular-learning\hello-world>ng g c emp
CREATE src/app/emp/emp.component.html (19 bytes)
CREATE src/app/emp/emp.component.spec.ts (594 bytes)
CREATE src/app/emp/emp.component.ts (213 bytes)
CREATE src/app/emp/emp.component.css (0 bytes)
```

2- Create a service

ng g s records



records.service.ts

If you are using standalone components

Then you need to import the CommonModule directly in @Component:

emp.component.ts

emp.component.ts U | app.component.html M | emp.component.html

hello-world > src > app > emp > emp.component.ts > EmpComponent

```ts
import { Component, OnInit } from '@angular/core';
import { RecordsService } from '../records.service';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-emp',
  imports: [CommonModule],
  templateUrl: './emp.component.html',
  styleUrl: './emp.component.css',
  providers: [RecordsService]
})
export class EmpComponent implements OnInit {

```

emp.component.ts U

hello-world > src > app > emp > emp.component.ts > EmpComponent

```ts
import { Component, OnInit } from '@angular/core';
import { RecordsService } from '../records.service';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-emp',
  imports: [CommonModule],
  templateUrl: './emp.component.html',
  styleUrl: './emp.component.css',
  providers: [RecordsService]
})
export class EmpComponent implements OnInit {

  infoReceived1: string[]=[];
  infoReceived2: string[]=[];
  infoReceived3: string[]=[];

  constructor(private rservice: RecordsService){}
  ngOnInit(): void {
  }
  getInfo1() {
    this.infoReceived1 = this.rservice.getInfo1();
  }
  getInfo2() {
    this.infoReceived2 = this.rservice.getInfo2();
  }
  getInfo3() {
    this.infoReceived3 = this.rservice.getInfo3();
  }

```

emp.component.html U

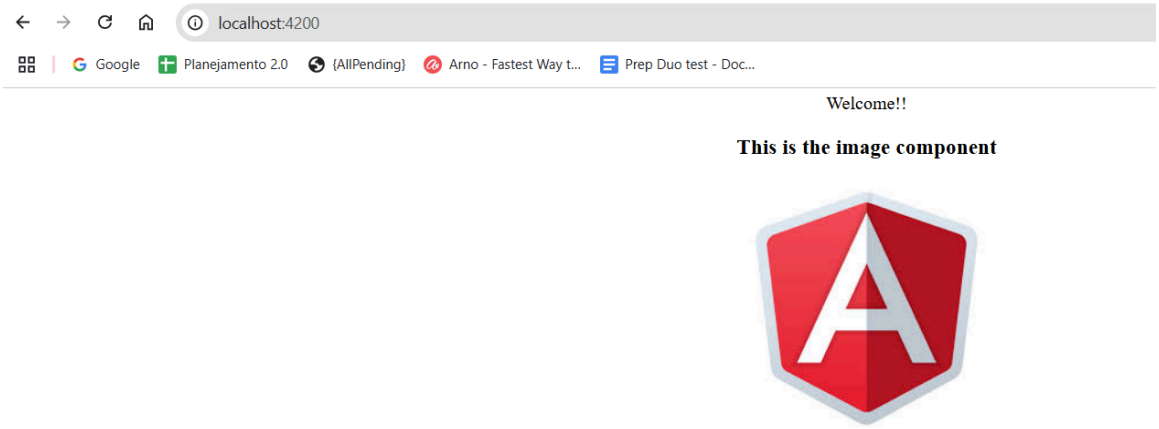
hello-world > src > app > emp > emp.component.html > button

```
<button type="button" (click)="getInfo1()">Employee1</button>
<ul class="list-group">
  <li *ngFor="let info of infoReceived1" class="list-group-item">{{info}}</li>
</ul>

<button type="button" (click)="getInfo2()">Employee2</button>
<ul class="list-group">
  <li *ngFor="let info of infoReceived2" class="list-group-item">{{info}}</li>
</ul>

<button type="button" (click)="getInfo3()">Employee3</button>
<ul class="list-group">
  <li *ngFor="let info of infoReceived3" class="list-group-item">{{info}}</li>
</ul>

```



Employee Details

Employee1

- Adam
- E123
- at@abc.net

Employee2

- Nina
- E231
- nn@abc.net

Employee3

Made with ❤️ while learning Angular with Simplilearn – Angular Basics.