Angular - Lec.1

Marina Magdy

Agenda

- What is angular ?
- Single Page Application
- Typescript overview
- Getting started with angular
- Angular Application Structure
- UI libraries : Bootstrap

What is Angular ?

- Angular is a **Javascript framework** which allows you to create single page applications. It's only one HTML file and a bunch of JavaScript code we got from the server that changes content of this HTML.
- Angular is using **Typescript** which is a superset of javascript and compiles to Javascript .
- Angular current stable version is 12.2.9
- Angular docs : https://angular.io/docs

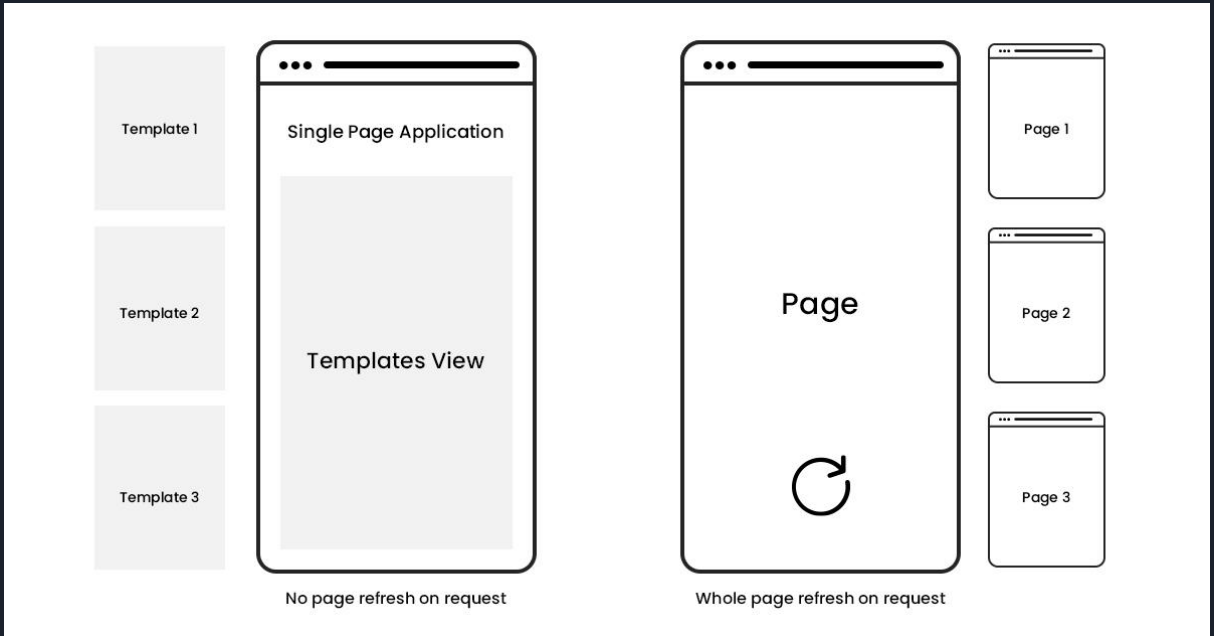
What is Angular ?

- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
- A suite of developer tools to help you develop, build, test, and update your code

Single page application

- Single Page Applications (SPAs) all the HTML generation happens in the browser. The server only returns one basic HTMl page for all incoming requests (no matter the URL).
- But that single HTML page contains a lot of JavaScript code (typically outsourced into separate files) which is responsible for changing the HTML code (technically, the DOM).
- Single page application examples : gmail,facebook,netflix ...etc

Single page application



What is Typescript ?

- TypeScript is an open-source, object-oriented language developed and maintained by Microsoft
- TypeScript extends JavaScript by adding data types, classes, and other object-oriented features with type-checking. It is a typed superset of JavaScript that compiles to plain JavaScript.

Why using Typescript !

- JavaScript is a dynamic programming language **with no type system**.
- JavaScript variables are declared using the var keyword, and it can point to any value. JavaScript doesn't support classes and other object-oriented features.
- The type system increases the code quality,readability and makes it easy to maintain and refactor codebase. More importantly, errors can be caught at compile time rather than at runtime.
- TypeScript compiles into simple JavaScript.

Typescript : Getting Started

TypeScript code is written in a file with .ts extension and then compiled into JavaScript using the TypeScript compiler. A TypeScript compiler needs to be installed on your platform. Once installed, the command tsc <filename>.ts compiles the TypeScript code into a plain JavaScript file. JavaScript files can then be included in the HTML and run on any browser.

- npm install -g typescript //install typescript
- tsc -v //check typescript version and make sure that it's installed
- tsc file.ts //run command to compile ts file to js

Typescript : Types

- **Duck typing** is the ability of a language to determine types at runtime. JavaScript follows the duck typing paradigm; it has types but they are determined dynamically and developers don't declare them.
- TypeScript goes one step further and adds types to the language. Having data types has been proven to increase productivity and code quality, and reduce errors at runtime.
- It's better to catch errors at compile time rather than have programs fail at runtime.

Typescript : Types

Example :

```
let projectStatus  = 1;

projectStatus = 'Success';

//Error: Type 'Success' is not assignable to type 'number'
```

Reason :

TypeScript determined the type as number and then alerted us when we were trying to assign a string.

Typescript : Interfaces

An interface is an abstract type. It only defines the 'signature' or shape of custom types. During transpilation, an interface will not generate any code, it is only used by Typescript for type checking during development.

Example :

```
interface Product {
  productName: string;
}
Let productItem : Product {
  productName:”Lipton”;
}
```

Typescript : Interfaces

```
 user {  name , age , education , job , experience } -> type object -> Type User

Custom Type for Users :

Interface User {
      Name : string;
      Age : number ;
      Education : Array<String> // [‘education1’ , ‘Education2’]
      Job : string; // ‘Job’
      Experience : Array<String>
}
```

Typescript : Classes

TypeScript is object oriented JavaScript. TypeScript supports object-oriented programming features like classes, interfaces, etc. A class in terms of OOP is a blueprint for creating objects. A class encapsulates data for the object.

A class can include the following:

- **Constructor :** method which is called when creating a new instance of class.
- **Properties :** is any variable declared in a class
- **Methods :** any functions represent actions an object can take.

Typescript : Constructors

The constructor is a special type of method which is called when creating an object. In TypeScript, the constructor method is always defined with the name "constructor".

If the class includes a parameterized constructor, then we can pass the values while creating the object.

```
class Product {
  productName: string;
  constructor() {}
}

Let product = new Product();
```

Typescript : Access modifiers

Access modifiers change the visibility of the properties and methods of a class.

TypeScript provides three access modifiers :

- **Private :** The private access modifier ensures that class members are visible only to that class and are not accessible outside the containing class.
- **Protected :** The protected modifier allows properties and methods of a class to be accessible within same class and within subclasses.
- **Public :** By default, all members of a class in TypeScript are public. All the public members can be accessed anywhere without any restrictions.
- **Readonly :** We can access read only member from the outside of a class, but its value cannot be changed.

Typescript : Modules

A module is a way to create a group of related variables, functions, classes, and interfaces, etc. It executes in the local scope, not in the global scope. In other words, the variables, functions, classes, and interfaces declared in a module **cannot** be accessible outside the module directly. We can create a module by using the **export** keyword and can use in other modules by using the **import** keyword.

File1.ts : export var greeting : string = "Hello World!";

File2.ts

Import { greeting } from file1 ;

console.log(greeting);

Angular : Getting Started

- Install npm (node package manager) :

  https://nodejs.org/en/

  What is NPM ?

  Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls Node.js packages in your application. It is also an online repository for open-source Node.js packages. The node community around the world creates useful modules and publishes them as packages in this repository.

Angular : Getting Started

- Install Angular CLI (command line interface) globally :

  npm install -g @angular/cli

  What is CLI ?

  The Angular CLI is a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell.

Angular : Getting Started

To create your first angular app , run the following command in CMD :

- `ng new project-name` (create angular app)
- `cd ./project-name` (enter project folder)
- `ng serve -o` (run application and -o refers to open application automatically in browser)

Angular : Getting Started

Let’s discover and have a deep look at our angular app structure

useful resources :
https://angular.io/guide/file-structure
https://angular.io/guide/bootstrapping#launching-your-app-with-a-root-module

Angular : Getting Started

app.module.ts

Defines the root module, named AppModule, that tells Angular how to assemble the application. Initially declares only the AppComponent. As you add more components to the app, they must be declared here.

- ***declarations***—array tells Angular which components belong to that module. As you create more components, add them to declarations.
- ***imports***—It tells Angular about other NgModules that this particular module needs to function properly.
- ***providers***—array is where you list the services the app needs. When you list services here, they are available app-wide.
- ***bootstrap***—the *root* component that Angular creates and inserts into the index.html host web page,The application launches by bootstrapping the root AppModule, which is also referred to as an entryComponent

Bootstrap

Bootstrap

- **Install Bootstrap :** npm install bootstrap
- Add bootstrap to **angular.json** file in styles array : "./node_modules/bootstrap/dist/css/bootstrap.min.css"
- And add “node_modules/bootstrap/dist/js/bootstrap.bundle.min.js” to scripts array

you can also bootstrap angular component from **ngBootstrap** :

https://ng-bootstrap.github.io/#/getting-started

Install command : ng add @ng-bootstrap/ng-bootstrap

Thank you

Portfolio

Replace code found in app.component.html and start create your portfolio page using Bootstrap.

Portfolio will contain the following sections :

- Hero section (name and job title).
- Bio and about me (education and experiences) section with button to download CV .
- Skills section with progress bar for each skill
- Portfolio and projects section
- Footer contains contact us section with email and social media links (facebook , github , linkedin) [Will use fontawesome]

Note : Installing ng-bootstrap and use one of its component (Bonus)