**Angular**

**What is Angular?**

**Angular** is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

**How does an Angular application work?**

Every Angular app consists of a file named angular.json. This file will contain all the configurations of the app. While building the app, the builder looks at this file to find the entry point of the application.

*"main": "src/main.ts",*

Inside the build section, the main property of the options object defines the entry point of the application which in this case is main.ts.

The main.ts file creates a browser environment for the application to run, and, along with this, it also calls a function called bootstrapModule, which bootstraps the application. These two steps are performed in the following order inside the main.ts file:

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

platformBrowserDynamic().bootstrapModule(AppModule)

In the above line of code, AppModule is getting bootstrapped. The AppModule is declared in the app.module.ts file. This module contains declarations of all the components. As one can see in the above file, AppComponent is getting bootstrapped.

This component is defined in app.component.ts file. This file interacts with the webpage and serves data to it.

**What are some of the advantages of Angular over other frameworks?**

Features that are provided out of the box - Angular provides a number of built-in features like, routing, state management, rxjs library and http services straight out of the box. This means that one does not need to look for the above stated features separately. They are all provided with angular.

Declarative UI - Angular uses HTML to render the UI of an application. HTML is a declarative language and is much easier to use than JavaScript.

Long-term Google support - Google announced Long-term support for Angular. This means that Google plans to stick with Angular and further scale up its ecosystem.

**List out differences between AngularJS and Angular?**

Architecture: Angular JS uses MVC, Angular replaced controllers to components.

Language: Angular JS uses JavaScript and angular uses TypeScript.

Mobile Support: Angular provide mobile support.

Structure: Maintaining the code is easier.

Expression Syntax: Angular syntax is much easy.

**What is AOT compilation? What are the advantages of AOT?**

Every Angular application consists of components and templates which the browser cannot understand. Therefore, all the Angular applications need to be compiled first before running inside the browser.

Angular provides two types of compilation:

* JIT(Just-in-Time) compilation
* AOT(Ahead-of-Time) compilation

Diagram

Description automatically generated

In JIT compilation, the application compiles inside the browser during runtime. Whereas in the AOT compilation, the application compiles during the build time.

The advantages of using AOT compilation are:

Since the application compiles before running inside the browser, the browser loads the executable code and renders the application immediately, which leads to **faster rendering**.

In AOT compilation, the compiler sends the external HTML and CSS files along with the application, eliminating separate AJAX requests for those source files, which leads to **fewer ajax requests**.

Developers can detect and handle errors during the building phase, which helps in **minimizing errors.**

The AOT compiler adds HTML and templates into the JS files before they run inside the browser. Due to this, there are no extra HTML files to be read, which provide **better security** to the application.

**Explain Components, Modules and Services in Angular?**

In Angular, components are the basic building blocks, which control a part of the UI for any application.

A component is defined using the @Component decorator. Every component consists of three parts, the template which loads the view for the component, a stylesheet which defines the look and feel for the component, and a class that contains the business logic for the component.

A **module** is a place where we can group components, directives, services, and pipes. Module decides whether the components, directives, etc can be used by other modules, by exporting or hiding these elements. Every module is defined with a @NgModule decorator.

By default, modules are of two types:

* Root Module
* Feature Module

Every application can have only one root module whereas, it can have one or more feature modules.

A root module imports BrowserModule, whereas a feature module imports CommonModule.

**Services** are objects which get instantiated only once during the lifetime of an application. The main objective of a service is to share data, functions with different components of an Angular application. A service is defined using a @Injectable decorator. A function defined inside a service can be invoked from any component or directive.

**What are lifecycle hooks in Angular? Explain a few lifecycle hooks.**

Every component in Angular has a lifecycle, different phases it goes through from the time of creation to the time it's destroyed. Angular provides hooks to tap into these phases and trigger changes at specific phases in a lifecycle.

Graphical user interface, application

Description automatically generated

**ngOnChanges**( ) This hook/method is called before ngOnInit and whenever one or more input properties of the component changes.

This method/hook receives a SimpleChanges object which contains the previous and current values of the property.

**ngOnInit**( ) This hook gets called once, after the ngOnChanges hook.

It initializes the component and sets the input properties of the component.

**ngDoCheck**( ) It gets called after ngOnChanges and ngOnInit and is used to detect and act on changes that cannot be detected by Angular.

We can implement our change detection algorithm in this hook. **ngAfterContentInit**( ) It gets called after the first ngDoCheck hook. This hook responds after the content gets projected inside the component.

**ngAfterContentChecked**( ) It gets called after ngAfterContentInit and every subsequent ngDoCheck. It responds after the projected content is checked.

**ngAfterViewInit**( ) It responds after a component's view, or a child component's view is initialized.

**ngAfterViewChecked**( ) It gets called after ngAfterViewInit, and it responds after the component's view, or the child component's view is checked.

**ngOnDestroy**( ) It gets called just before Angular destroys the component. This hook can be used to clean up the code and detach event handlers.

**What is Data Binding?**

Communication between html template and ts code.

**What is String interpolation and Property binding?**

String interpolation and property binding are parts of data-binding in Angular. String interpolation and property binding allow only one-way data binding.

String Interpolation({{}}) in Angular 8 is a one-way data-binding technique that is used to transfer the data from a TypeScript code to an HTML template (view). It uses the template expression in double curly braces to display the data from the component to the view.

Property binding([]) in Angular helps you set values for properties of HTML elements or directives. With property binding, you can do things such as toggle button functionality, set paths programmatically, and share values between components.

**What is Event Binding?**

Event binding( () ) is used to handle the events raised by the user actions like button click, mouse movement, keystrokes, etc. When the DOM event happens at an element(e.g. click, keydown, keyup), it calls the specified method in the component.

**What are Directives and different types of directives?**

A directive is a class in Angular that is declared with a @Directive decorator. Components are directives. With Angular's built-in directives, you can manage forms, lists, styles, and what users see.

Directives are used as a attributes on the HTML tags.

Component directives:

These form the main class in directives. Instead of @Directive decorator we use @Component decorator to declare these directives. These directives have a view, a stylesheet, and a selector property.

Attribute directives—directives that change the appearance or behavior of an element, component, or another directive.

[ngStyle] = “{backgroundColor:getColor()}”

[ngClass] = “{online:serverStatus===’online’}”

Structural directives are the directives which change the structure of DOM, most start with \* , i.e \*ngIf

**What is the use of directives?**

Consider an application, where multiple components need to have similar functionalities. The norm thing to do is by adding this functionality individually to every component but, this task is tedious to perform. In such a situation, one can create a directive having the required functionality and then, import the directive to components which require this functionality.

**What are Pipes in Angular?**

A pipe takes in data as input and transforms it to a desired output. For example, let us take a pipe to transform a component's birthday property into a human-friendly date using date pipe.

**How are observables different from promises?**

The first difference is that an Observable is lazy whereas a Promise is eager.

Chart, table, treemap chart

Description automatically generated

The next difference is that Promises are always asynchronous. Even when the promise is immediately resolved. Whereas an Observable, can be both synchronous and asynchronous.

**Can one make an angular application to render on the server-side?**

Yes, angular provides a technology called Angular Universal, which can be used to render applications on the server-side.

The advantages of using Angular Universal are :

First time users can instantly see a view of the application. This benefits in providing **better user experience**.

Many search engines expect pages in plain HTML, thus, Universal can make sure that your content is available on every search engine, which leads to **better SEO.**

Any server-side rendered application **loads faster** since rendered pages are available to the browser sooner.