Arun Vijayarengan

Angular 16

A Quick Reference Guide

Angular Dev Env Setup

Install NodeJS

Download NodeJS (version 16.0 or later) from NodeJS.org Official Website

https://nodejs.org/en/.

Install Visual Studio Code

https://code.visualstudio.com/download

Launch Command Prompt / Terminal & Type the following commands

```
> node -v

> npm -v

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

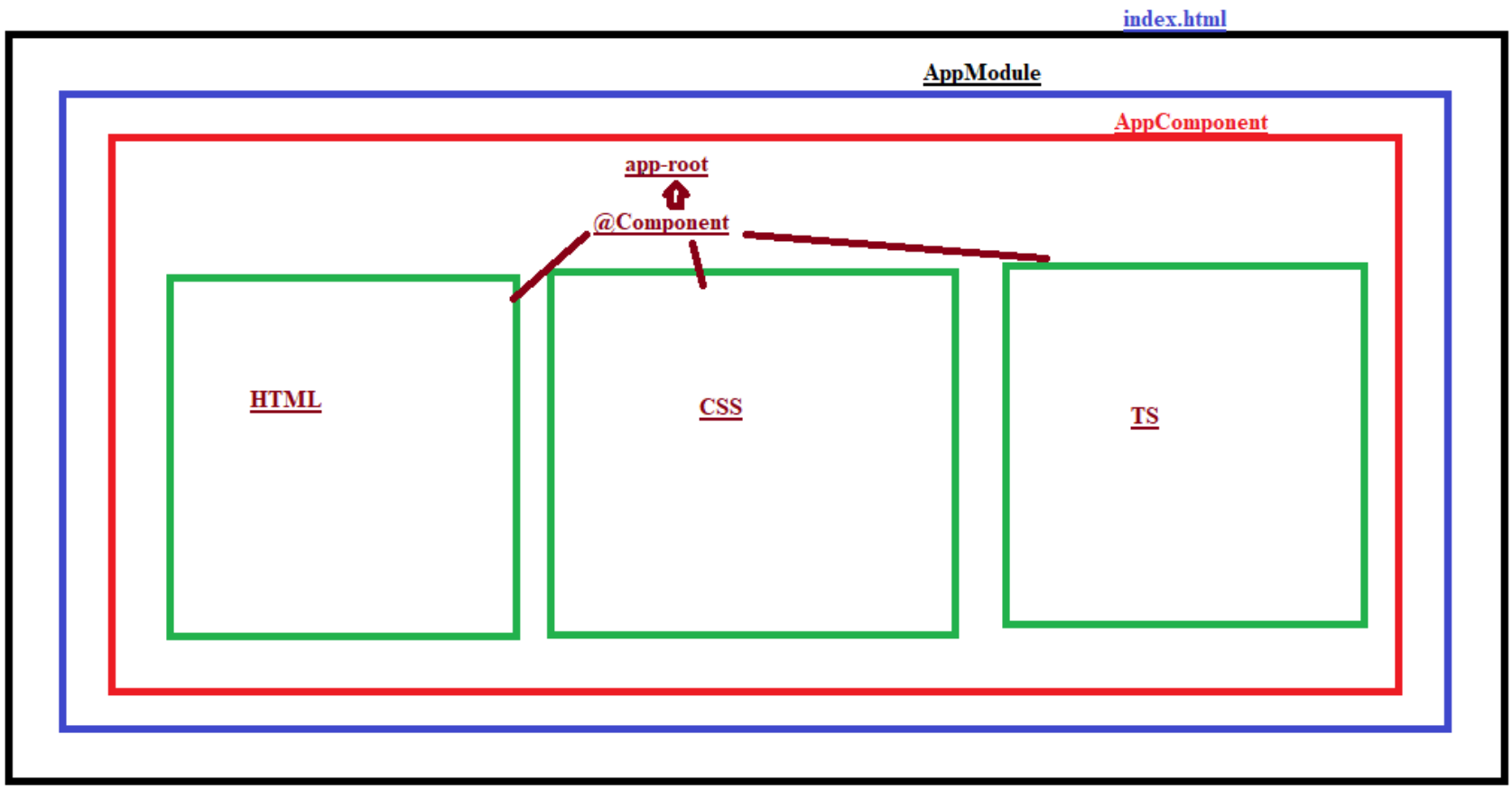
Then, create your first Angular 16 project.

```
> ng new my-app                    (project name is my-app)

> cd my-app

> ng serve                         (to start the server)
```

Project Files Walkthrough

Refer https://angular.io/guide/file-structure

How Angular App is bootstrapped?



Angular app is bootstrapping a Module and Module in turn is bootstrapping a component. Component consists of HTML, CSS, TS and exposed in a selector called app-root. It is referred in index.html and thus the App is bootstrapping.

Angular Building Blocks

```
Modules

Component

Directives

Services
```

```
    Models/Classes

    Interfaces

    Pipes

    Guards
```

Modules

Module is a bigger building block in can accommodate other building blocks inside. For each feature of our app we can create a feature modules and also an associated feature routing module.

```
> ng g m contact
```

Components

Main Building Block in Angular App is Component

```
> ng g c contact
```

Component is made up of HTML, CSS and TS with data in variables and methods.
Component is a reusable block of code. It can be used by the selector of the component.

Routing

- Helps us build SPA
- Stops page reload and updates the main section of the page

Step 1: Add the links and set up paths in nav.component.html

```
<ul class="nav navbar-nav">

    <li class="active"><a href="/">Home</a></li>

    <li ><a href="/concepts">Concepts</a></li>

    <li ><a href="/contacts">Contacts</a></li>

    <li ><a href="/about">About</a></li>

    <li ><a href="/get-in-touch">Get in Touch</a></li>

</ul>
```

Step2: Let's have the routes configured in app-routing.module.ts

```
//syntax : path should have string and the component when that path
is hit

const routes: Routes = [

  { path: '', component: ConceptsComponent },

  { path: 'contacts', component: ContactsComponent },

  { path: 'about', component: AboutComponent },

  { path: 'get-in-touch', component: GetInTouchComponent },

];
```

Step3: What should be replaced? Add router outlet in layout file app.comp.html

```
<div style="text-align:center; margin-top:80px;">

  <router-outlet></router-outlet>

</div>
```

Step 4: Check the app. The routes should work. But the page will reload. Let's fix it.

in nav.comp.html replace href with routerLink

```
<ul class="nav navbar-nav">

    <li class="active"><a routerLink="/">Home</a></li>

    <li ><a routerLink="/concepts">Concepts</a></li>

    <li ><a routerLink="/contacts">Contacts</a></li>

    <li ><a routerLink="/about">About</a></li>

    <li ><a routerLink="/get-in-touch">Get in Touch</a></li>

</ul>
```

Step5: Check the app. page should not reload. Let's have active class. have routerLinkActive in li.

```
<li routerLinkActive="active">
```

Step6: Now, when you see the navigation menu, active class will be added in another link also. Let’s fix it.

```
<ul class="nav navbar-nav">

    <li routerLinkActive="active"

      [routerLinkActiveOptions]="{exact: true}"><a
routerLink="/">Home</a></li>

    <li routerLinkActive="active"><a routerLink="/
concepts">Concepts</a></li>
```

```
</ul>
```

Data Binding

Data Binding is the concept of binding the the data from component.ts and displaying in component.html

1. String Interpolation - {{ }}

TS => HTML

2. Property Binding - [ ]

TS => HTML

3. Event Binding - ()

TS => HTML

4. Two Way Binding - [(ngModel)]

TS <=> HTML

FormsModule should be adde under imports:[ ] section in app.module.ts or feature.module.ts

5. Custom Property Binding

-- helps us in Parent to Child comp Communication

6. Custom Event Binding

-- helps us in child to parent comp Communication

Cross Component Communication

1. Parent to Child Component Communication can be implemented with Custom Property Binding

2. Child to Parent Component Communication can be implemented with Custom Event Binding and @ViewChild() and @ViewChildren()

3. Any component to Any component is possible if you have common shared service class.

Possible Project Structures in Angular Projects #1

```
src/

  app/

    shared/

      header/

      footer/

      nav/

   auth/

      login/

        login.comp.ts

        login.comp.html

        login.service.ts

        login.directive.ts

      signup/

      reset-pw/
```

```
    dashboard/

      ...

      ...

    reports/

      ...

      ...
```

#2

```
src/

  app/

    components/

      shared/

      concepts/

      contacts/

      about/

    directives/

    services/

    guards/

    pipes/

    models/
```

```
    interfaces/
```

Debugging

Use Chrome Inspector

-- Go to source Tab locate webpack/projectfolder/ and open .ts for Sourcemaps

-- or press ctrl + p then specify the file you want to debug

Augury Extension to have a detailed representation of comp, modules.

Install Angular Dev Tools chrome extension

https://chrome.google.com/webstore/detail/angular-devtools/ienfalfjdbdpebioblfackkekamfmbnh

Directives

Some instruction to manipulate the Document Object Model (DOM) of the Component just like it happens thru jQuery plugin.

1) Attribute Directives

<div [ngClass]="">

<div [ngStyle]="">

2) Structural Directives

*ngIf

*ngFor (to understand let keyword: https://leanpub.com/understandinges6/read)

ngSwitch

3) Custom Attribute Directives

```
> ng g d directives/colorizr
```

Forms

1. Template Driven Forms

- Built using html

Adv

1. Easy to implement
2. Quick to implement

Disadv

1. Challenging to do some validations

2. Reactive Forms

- Built using Ts / Angular form related API's

Adv

1. Good if you are doing complex validations
2. You will have more control over the form field
3. Good for unit testing

Disadv

1. Little bigger Learning curve

2. Challenging to implement

Services

```
> ng g s contact
```

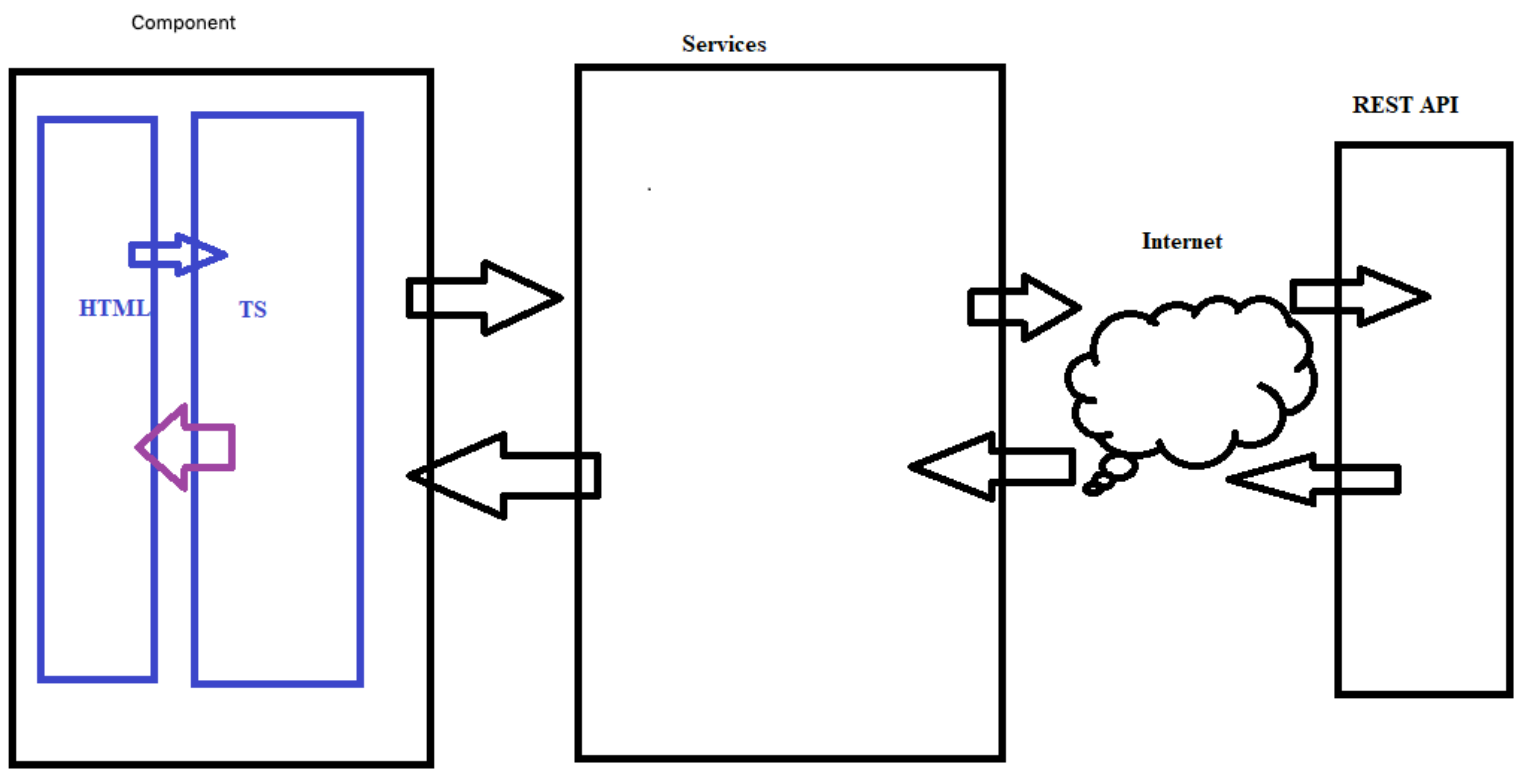
- Services are JavaScript functions that are responsible for doing a specific task. Services are injected using Dependency Injection.

- Building block that helps component connect with REST api

What's Dependency Injection?

Dependency Injection is a design pattern that passes object as dependencies in different components across the application. Examples: renderer, elementRef

- Intermediate layer b/w components and rest api



- Responsible for connecting to rest api end point
- get the data from comp
- send the data to rest api
- receive response from rest api
- send the response back to comp

RxJS and Observables

What's Observable?

- It's a collection that returned over time.
- It's a push, that sends multiple values
- Observable is like restaurant kitchen.
- If you subscribe to observable, it decides when to give information

Now, What's RxJs?

Various method of operators, you can chain on Observables to filter data, sort, pipe, and map.

Refer this example: http://jsbin.com/kuzifehivi/4/edit?html,js,console

Testing

Running Tests

Take build using the following command

THE FOUR TYPES OF TESTS

End to End

A helper robot that behaves like a user to click around the app and verify that it functions correctly.

Sometimes called "functional testing" or e2e.

Integration

Verify that several units work together in harmony.

Unit

Verify that individual, isolated parts work as expected.

Static

Catch typos and type errors as you write the code.


```
> ng test
```

Linting Angular Apps

Take build using the following command

```
> ng lint
```

Taking Build of Angular App

Take build using the following command

```
> ng build —prod —base-href /app-name/
```

Then, copy the list folder and deploy it in your webserver.

Security

Some Reference Articles related to Security in Angular App

How angular protects XSS?

https://hackernoon.com/how-angular-protects-us-from-xss-attacks-3cb7a7d49d95

Angular – How to Prevent XSS Attacks – Code Examples

https://vitalflux.com/angular-prevent-xss-attacks-code-examples/

Angular recommendations to prevent Server XSS Attacks

As part of server-side processing, escape all data before sending them as Http response. That would mean that if response data consisted of HTML/Javascript tags, they will get escaped.

Avoid generating Angular templates as part of server-side processing. This may lead to template injection thereby resulting in DOM manipulation when the page loads in the browser.

Angular recommendations to prevent Client XSS Attacks

Read here: https://vitalflux.com/angular-prevent-xss-attacks-code-examples/

https://vitalflux.com/angular-top-10-security-best-practices-vis-vis-security-risks/

Angular CLI Commands

Refer the list of commands here

https://github.com/angular/angular-cli/wiki

Typescript

JavaScript + Data typing + OOPS = Typescript

Optionally typed language

Compiled-to-Javascript language

Typescript compiler compiles the TS code into JS

TypeScript Code Example

```
var x: number = 10; //explicit

var y = 20; // implicit

var myName:string = "Arun";

var isLoggedIn: boolean = true;

var skillList: Array<string> = [

];

//another way

var skillList1: string[] = [

];

var myProfile: { } = {

}
```

```
var everything: any = "Arun";

everything = 235356;



class Car extends Vehicle{

    constructor()  {

    }

    drive() {

    }

    reverse() {

    }

    static park(){



    }

}



var car = new Car()

car.drive();
```

Try typescript playground

http://www.typescriptlang.org/play/index.html