Angular Fundamentals

.NET

Angular is an application design framework and development platform for creating efficient and sophisticated single-page applications.

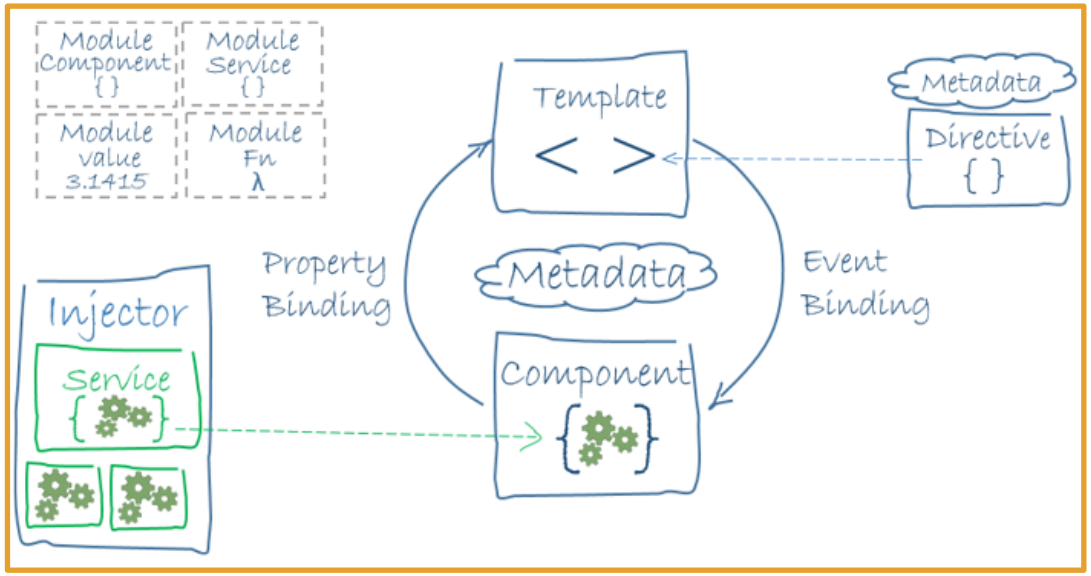
HTTPS://ANGULAR.IO/DOCS

What is Angular

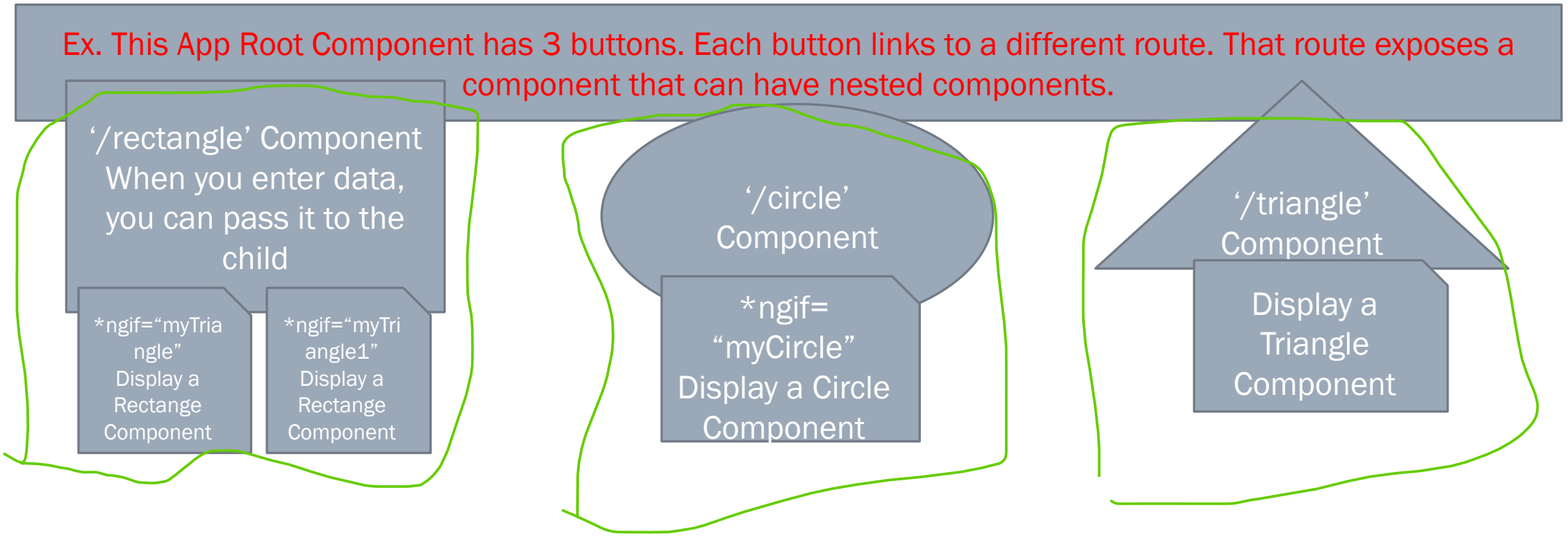
https://hackr.io/blog/angular-interview-questions
https://angular.io/guide/aot-compiler
https://www.geeksforgeeks.org/what-is-spa-single-page-application-in-angularjs/

AngularJS is a JavaScript-based front-end web framework that bases data transfer on bi-directional UI data binding, is used to design Single Page Applications, and leverages Ahead-of-Time Compiling to render webpages more quickly.

- Single-Page Application – Single Page Applications are web applications that load a single HTML page and only a part of the page instead of the entire page gets updated with every click of the mouse. The page does not reload or transfer control to another page during the process. This ensures high performance and loading pages faster.
- Ahead-of-Time Compiling – Angular component templates cannot be directly rendered by the browser. Therefore, Angular applications require a compilation process before they can run in a browser. The Angular ahead-of-time (AOT) compiler converts Angular HTML and TypeScript code into JavaScript during the build phase before the browser downloads and runs that code. This provides a faster rendering in the browser.



Angular Structure Diagram use shapes for example.



TS/Angular Workspace SetUp (1/2)

https://angular.io/guide/setup-local
https://code.visualstudio.com/docs/typescript/typescript-compiling
https://angular.io/tutorial/toh-pt0#create-a-new-workspace-and-an-initial-application

---

Following the steps from here to create your first Angular App.

1. Make sure you have Node.js with node –v in Command Line. If not, go to nodejs.org to get it.
2. (This is required only once ever) Install Angular CLI globally with:
   - npm install -g @angular/cli
3. Create a ***WorkSpace*** (accept all the default settings) for your app and install the default starter app with:
   - ng new <my-app-name>
4. App name must start with a letter and only contain numbers, letters, and dashes.
5. Install the Angular ***npm*** packages needed with:
   - ng new

TS/Angular Workspace SetUp (2/2)

https://angular.io/guide/setup-local
https://code.visualstudio.com/docs/typescript/typescript-compiling
https://angular.io/tutorial/toh-pt0#create-a-new-workspace-and-an-initial-application

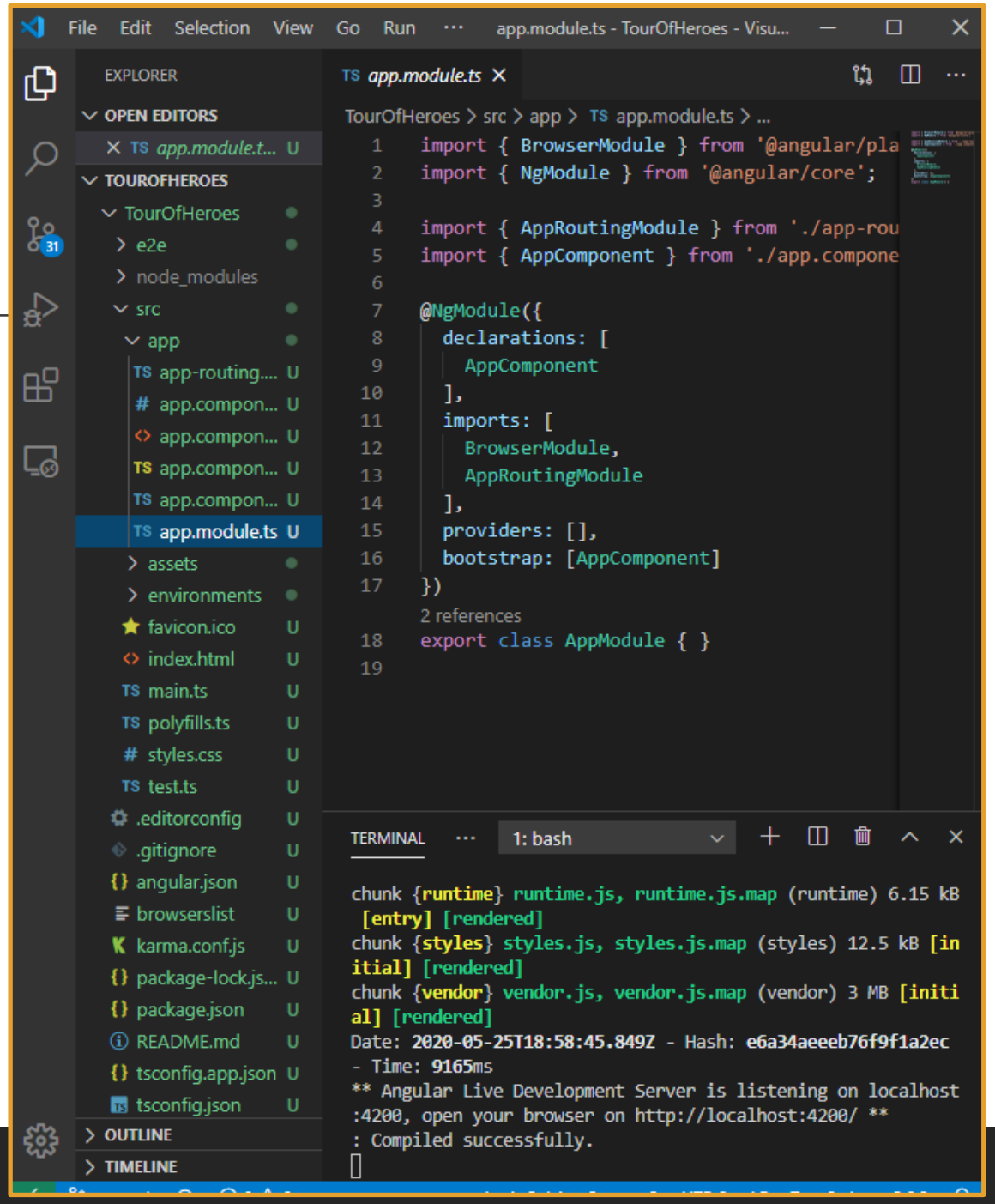
---

6. Navigate in the CLI to your app folder with:
   - cd <my-app-name>
7. Launch the server and open the browser with the default sample project with:
   - ng serve --open (2 dashes)
8. In VS Code, you can install the ***Angular Extension Pack*** to get additional useful tools.
9. VS Code extensions recommendations: C#, C# Extensions, Bracket Pair Colorizer 2, Nuget Gallery, Material Icon Theme,
10. Use this Angular Cheat Sheet for quick reference!

Angular WorkSpace

https://angular.io/tutorial/toh-pt0#set-up-your-environment

A workspace contains all the files for one or more projects. A project is the set of files that comprise an app, a library, unit tests, integrations tests, or end-to-end (e2e) tests.



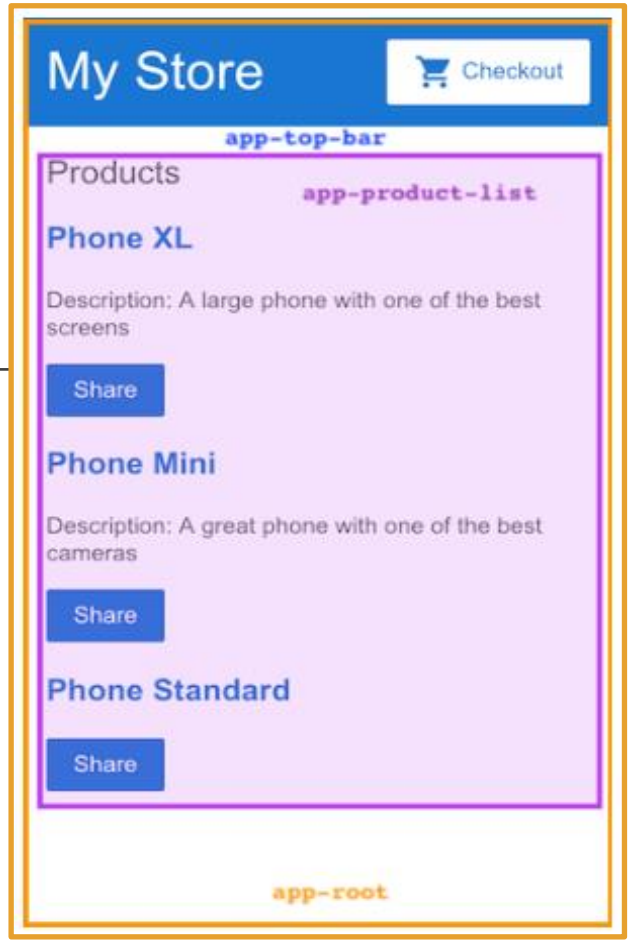
Angular Component

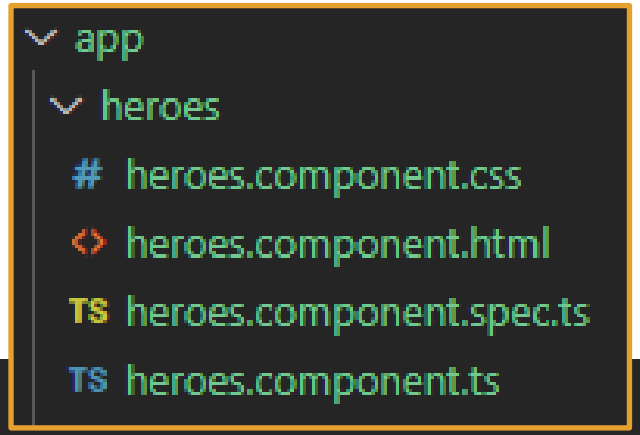
https://angular.io/tutorial/toh-pt0#set-up-your-environment
https://angular.io/guide/component-interaction

Components are the fundamental building blocks of ***Angular*** applications. They display data on the screen, listen for user input, and act based on that input.

An ***Angular*** application comprises a tree of ***components***. Each ***Angular component*** has a specific purpose and responsibility. In the example to the right, there are 3 components displayed:

- ***app-root*** (orange box) is the application shell. This is the first component to load and the parent of all other components. You can think of it as the base page.
- ***app-top-bar*** (blue banner) is the store name and checkout button.
- ***app-product-list*** (purple box) is the product list.





Angular Component

https://angular.io/tutorial/toh-pt1#create-the-heroes-component

The command ng generate component <name> creates a new ***component***. The ***CLI*** creates a new folder for each ***component*** and generates a .css, .ts, and .html, .spec.ts inside it.

Always import { Component, OnInit } from @angular/core;

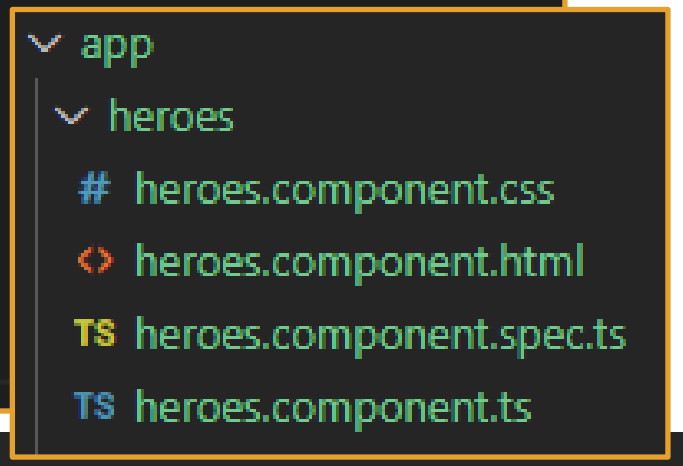
Annotate the ***component class*** with @Component(). @Component is a ***decorator*** function that specifies the Angular metadata for the ***component***:

1. The selector name to use for CSS and if importing this component into a .html page.
2. The relative .html location.
3. The relative .css location.

Use export to make the class available for import by other components.

ngOnInit() is a ***lifecycle hook***. It's the best place for @Component initialization logic, such as getting current data from a ***Service*** or initializing variables.

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-heroes',
5    templateUrl: './heroes.component.html',
6    styleUrls: ['./heroes.component.css']
7  })
   7 references
8  export class HeroesComponent implements OnInit {
9
       0 references
10     constructor() { }
11
       2 references
12     ngOnInit(): void {
13     }
14
15 }
16
```



Connect a new Component

https://angular.io/tutorial/toh-pt1#show-the-heroescomponent-view

Every ***component*** must be declared in the app.module.ts @NgModule to function.

Angular CLI automatically imports new components into app.module.ts and declares them under the @NgModule.declarations array.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms'; // <-- NgModel lives here

import { AppComponent } from './app.component';
import { HeroesComponent } from './heroes/heroes.component';

@NgModule({
  declarations: [
    AppComponent,
    HeroesComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Angular TypeScript Interface

https://angular.io/tutorial/toh-pt1#create-a-hero-interface

Interfaces are useful for when you want to define a class or object (with its types), then import it into components where needed.

Create an ***interface*** with:

- ng generate interface <ComponentName>.

Then import that ***interface*** into the ***Component*** in which you want to use it from the relative file location.

src/app/hero.ts

```
export interface Hero {
  id: number;
  name: string;
}
```

```
1  import { Component, OnInit } from '@angular/core';
2  import { Hero } from '../hero';
```

TypeScript Modules

https://www.typescriptlang.org/docs/handbook/modules.html

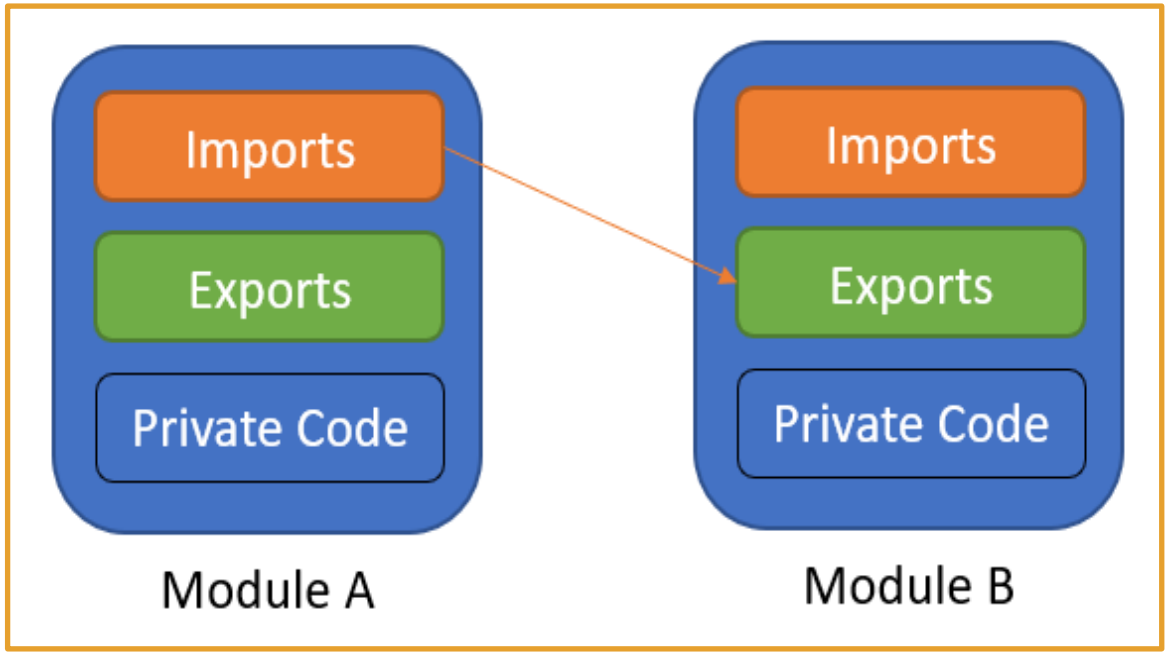
TS shares the *JS* concept of ***Modules***. ***Modules*** in *TS* have their own scope. A module must be explicitly exported to make its members visible.

To consume a property ***exported*** from a different ***module***, it must be ***imported*** using an ***import*** method.

The relationships between ***modules*** are specified in terms of ***imports*** and ***exports*** at the file level.

In *TS*, any file containing a top-level ***import*** or ***export*** is considered a ***module***.

A file without any top-level ***import*** or ***export*** declarations is treated as a script whose contents are available in the global scope (and therefore in ***modules*** as well).



TypeScript - Exporting a Declaration

https://www.typescriptlang.org/docs/handbook/modules.html#export

Any declaration (variable, function, class, type alias, interface) can be ***exported***.

1. Use the ***export*** keyword to make a class, function, or variable available to other ***modules*** from within the ***module*** (***component***).
2. ***Import*** the class, function, or variable into the ***module*** (***component***) where you want to implement it.

```
export interface StringValidator {
  isAcceptable(s: string): boolean;
}
```

```
import { StringValidator } from "./StringValidator";

export const numberRegexp = /^[0-9]+$/;

export class ZipCodeValidator implements StringValidator {
  isAcceptable(s: string) {
    return s.length === 5 && numberRegexp.test(s);
  }
}
```

Angular Interpolation

https://angular.io/guide/interpolation

Text interpolation allows you to incorporate dynamic string values into your HTML templates.

Interpolation refers to embedding ***template expressions*** into marked up text. This allows the .ts file to contain information that the .html will display dynamically.

By default, interpolation uses the double curly braces {{ and }} as delimiters.

src/app/app.component.ts

```
currentCustomer = 'Maria';
```

src/app/app.component.html

```
<h3>Current customer: {{ currentCustomer }}</h3>
```

Current customer: Maria

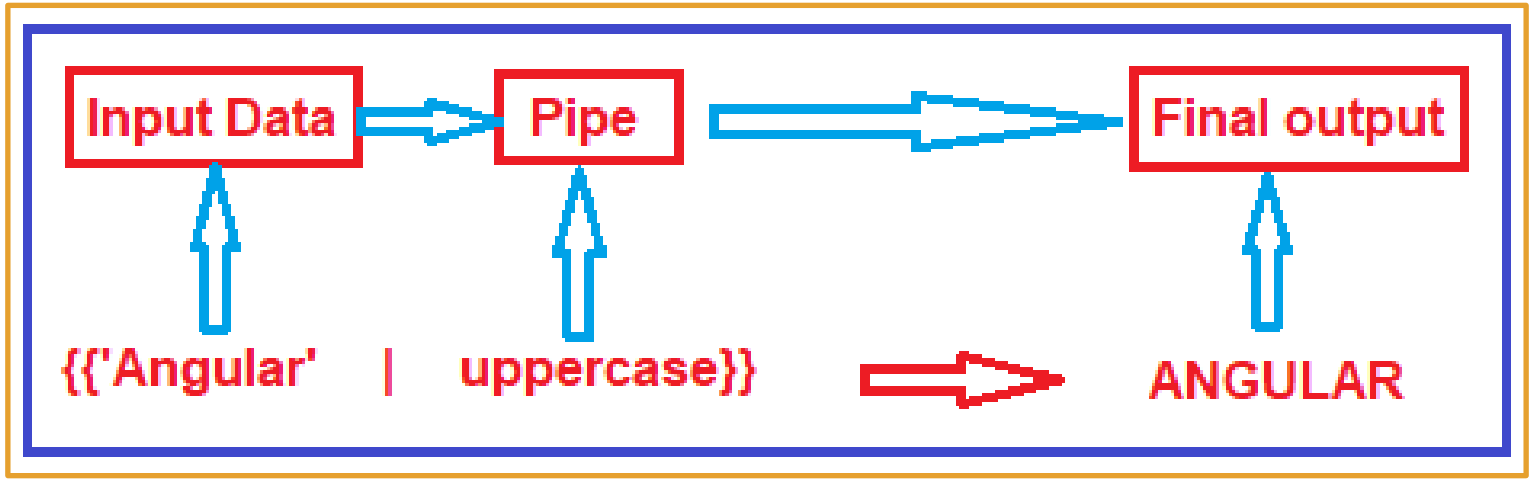
Angular Pipes (1/2)

https://angular.io/guide/pipes

Use Angular ***pipes*** in template .html files to transform strings, currency amounts, dates, and other data for display in a browser.

Pipes are simple ***functions*** you can use in template expressions to accept an input value and return a transformed value.

Angular provides built-in pipes for typical data transformations



Angular Pipes (2/2)

https://angular.io/guide/pipes
https://angular.io/api/common#pipes

Command	Name	Useage
...| date }}	Datepipe	Formats a date based on locale. Ex. {{ dateObj | date:'mm:ss' }}
...| uppercase }}	UpperCasePipe	Transforms text to all upper case. Ex. {{ value | uppercase }}
...| lowercase }}	LowerCasePipe	Transforms text to all lower case. Ex. {{ value | lowercase }}
...| currency }}	CurrencyPipe	Transforms a number to a currency string based on locale. Ex. {{ b | currency:'CAD':'symbol':'4.2-2' }}
...| number }}	DecimalPipe	Transforms a number into a string with a decimal point based on locale. Ex. {{ pi | number:'4.1-3' }}
...| percent }}	PercentPipe	Transforms a number to a percentage string based on locale. Ex. {{ b | percent:'4.3-5' }}
...| titlecase }}	TitlecasePipe	Capitalizes the first letter of each word and transforms the rest of the word to lower case. Ex. {{ 'a string' | titlecase }}

How to stop a running Angular Program

https://anthonygiretti.com/2018/03/26/how-to-avoid-port-4200-is-already-in-use-error-with-angular-cli/

---

1. In Command Line, use netstat -ano | findstr :yourPortNumber. (Usually it's 4200 with Angular.) to get your process number (PID). It's on the right or 'Listening'
2. In Command Line, use tskill [yourPID#].
3. In Command Line, use ng serve –open to recompile and reopen your app.

Deployment

https://angular.io/start/start-deployment

karma.config changes

still needed.

```
trigger:
  branches:
    include:
    - main

#the below is required only if you have multiple .sln files in your repo.
  paths:
    include:
    - AngularDemo/memesaver

pool:
  vmImage: 'windows-latest'

variables:
  solution: 'AngularDemo/memesaver'
  buildPlatform: 'Any CPU'
  buildConfiguration: 'Release'

steps:
- task: SonarCloudPrepare@1
  displayName: sonarcloud prepare
  inputs:
    SonarCloud: 'sonarclourangular'
    organization: '03012021batch'
    scannerMode: 'CLI'
    configMode: 'manual'
    cliProjectKey: '03012021Batch_memesaverangular'
    cliProjectName: 'memesaverangular'
    cliSources: '$(solution)/src'
    extraProperties: 'sonar.javascript.lcov.reportPaths=$(solution)/coverage/memesaver/lcov.info'

- task: NodeTool@0
  inputs:
    versionSpec: '10.x'
  displayName: 'Install Node.js'

- task: Npm@1
  displayName: 'NPM Install'
  inputs:
    command: 'install'
    workingDir: $(solution)

- task: Npm@1
  displayName: 'NPM Build Angular'
  inputs:
    command: 'custom'
    workingDir: $(solution)
    customCommand: 'run build-prod'

- task: Npm@1
  displayName: 'NPM Test'
  inputs:
    command: 'custom'
    workingDir: $(solution)
    customCommand: 'run test-headless'

- task: SonarCloudAnalyze@1
  displayName: 'SonarCloud Analyze'

- task: SonarCloudPublish@1
  displayName: 'SonarCloud Publish'
  inputs:
    pollingTimeoutSec: '300'
```

Assignment –
Rock, Paper, Scissors in Angular

https://www.codementor.io/@brijmcq/creating-a-rock-paper-scissors-game-in-angular-fmmrknce8