Angular

Package.json

Dependencies on the angular Project are found here

Angular.json

Project specific configurations like root project

Tsconfig.json

Typescript configurations details

Src/index

main index file, file to render in the browser

src/styles.cc

global styles

src/app

components, services, htmls

app.config.ts

configure angular models

app.routes.ts

define angular routes

angular cli

command line interface tool to initialize, develop and maintain angular apps from command shell

component standalone property

angular components marked as standalone do not need to be declared in an ngmodule, this components manage their own template dependencies (components, directives, pipes) via the imports property

export class (Component) in order to be available for main.ts

A screen shot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

Interpolation

Title:string = “this is loaded dynamically”

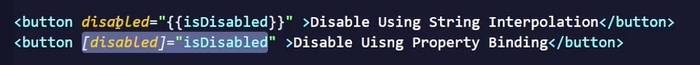
<h1>{{title}}</h1>

Any value in the {{}} is converted into a string

Property binding

imgUrl =”http image”

<img [src]=”imgUrl”>



A white box with black text

AI-generated content may be incorrect.

Event binding

<button (click)=”updateCounter()”>Click</button>

<button (mouseover)=”updateCounter()”>Click</button>

</input type=”text” (keyup)=”keyEntered($event)”>

keyEntered(event:any){console.log(event.keycode)}

OR

</input type=”text” (keyup.enter)=”keyEntered()”>

Template variable

Keep data into a template variable, keep not only the value but also all properties from the element

</input type=”text” (keyup.enter)=”keyEntered(user)” #user>

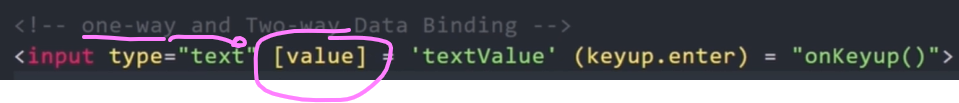
keyEntered(user:HtmlInputElement){console.log(user.value))

two way data binding with ngModel

<input type=”text” [(ngModel)]=”username”>

username:string =’john Doe’

<h3>{{username}}</h3>



A data binding diagram with text

AI-generated content may be incorrect.

A white background with blue text

AI-generated content may be incorrect.

Directives

Allow to manipulate the dom and add additional behavior to elements

Types

Components directive, directive with template

Structural directives, alter layout of dom (ngIf, ngFor, ngSwitch older, @if, @for @switch new)

Attribute Directives, change appearance or behavier of an element (ngClass, ngStyle)

Custom directive

Old

<h2 \*ngIf=”isLoggedIn; else message”>{{username}}</h2>

<ng-template #message>

<h2>user not logged in</h2>

</ng-template>

new

@if (isLoggedIn){

<h2>{{username}}</h2>

}

@else {

<h2>user not logged in</h2>

}

A computer screen with white text

AI-generated content may be incorrect.

Ng-template

Allow to define content that can be used later in the app, it is not render in the dom unless we call it

Ng-template-outlet

Similar to ng-template, to use as a directive instead of else

<ng-template #callToAction>

<button>Join</button>

</ng-template>

<div [ngTemplateOutlet]=”callToAction”>

<h5>Sidebar<h/>

\*\*\*it will add ng-template callToAction here\*\*\*

</div>

Access array index

<h4>{{users[3]}}</h4>

ngFor (old)

<h3 \*ngFor=”let user of users”>{{user}}</h3>

@for(new)

@for(user of users; track user){

<h3>{{user}}</h3>

}

@for with arrays

usersObj: Array<any>=[

{

Id: 1, name: ‘John’, email:’john@gmail.com’

}]

@for(user: usersObj; track user){

<li>{{user.id}} {{user.name}} {{user.email}}</li>

}

Add object to array

Array.push(newObj);

Array get index from obj

Index: number = Array.indexOf(user);

Array remove obj using index

Array.splice(index, 1) //from index delete 1 element

For index and count

(old) <li \*ngFor=”let user of users; let i=index; let counter= count” (click)=”deleteUser(i)”>Delete</li>

@for(user of users; track user; let i =$index; let counter = $count){

<li (click)=deleteUser(i)>Delete</li>

}

@empty

If evaluates empty is true it renders its content, , if empty it will just render it once

@for(user of users; track user; let i =$index; let counter = $count){

<li (click)=deleteUser(i)>Delete</li>

}

@empty{

<p>nothing to display</p>

}

@for other functionalities available at for block

$First

$last

$even

$odd

@for(user of users; let initial = $first; let last= $last; let odd = $odd; let even=$even)

Ng switch

(old)

<div [ngSwitch]=”usersRole”>

<p \*ngSwitchCase=”’Admin’”>Welcome Admin</p>

<p \*ngSwitchCase=”’Member’”>Welcome Member</p>

<p \*ngSwitchDefault>Login</p>

</div>

@switch(usersRole){

@case(‘admin’){

<p>Welcome Amin</p>

}

@case(‘member’){

<p>Welcome Member</p>

}

@default{

<p>Login</p>

}

}

ngStyle

modify the style of an element base on a condition

A computer code with text

AI-generated content may be incorrect.

ngClass

add a class based on a condition

A screenshot of a computer code

AI-generated content may be incorrect.

Structural Directive

Add or remove dom elements, leading \* symbol (old way)

Attribute directive

Change appearance or behabior of a dom element

ngStyle

ngClass

parent component is the outer component, parent component can pass data to the child component, child can also pass data to the parent component

child component is the inner component

parent

appPostTitle: string = ‘Post 1’

child receives postTitle

<app-child-component [postTitle]=”appPostTitle”>¨

@Input() postTitle: string = ‘’; //input to tell this is coming from its parent component

Get data into the parent component from the child component

@ViewChild

Child

childMessage: string = ‘hello from child component’

Parent

@ViewChild(ChildComponent) childMessage: any;

(If you try to get the value in the parents constructor it will be undefined since it is not available yet also if you try to use it in string interpolation (child not completely initialized), AfterViewInit phase will have it ready)

AfterViewInit (runs after component and its children are fully initialized)

Export class AppParentComponent implements AfterViewInit{

ngAfterViewInit(){

console.log(this.childMessage);

}

You can assign inside ngAfterViewInit() a variable so it can be used in string interpolation

@Output

For actions like button clicks or user interactions, to pass it from the child component to the parent component

Child

parentMessage:string =”message from child using click event”;

@Output() MessageEvent = new EventEmitter();

sendMessage(){ this.MessageEvent.emit(this.parentMessage); }

<button (click)=”sendMessage()”>Send Message To Parent</button>

Parent

<app-child-component (MessageEvent)=”receiveMessage($event)”>

<p>{{messageFromChild}}</p>

receiveMessage(message: string) {

console.log(message)

this.messageFromChild = message;

}

messageFromChild:string = ‘’;

@ViewChild has to look on all component for the looking value, if it is large then better to use @Output which will only report when needed

Ng-content content projection

@Input send Data to child component

Ng-content send Html blocks to child component

Parent

<app-card>

<p>This is loaded using ng-content</p>

<app-card>

Child

<ng-content>html from parent goes here</ng-content>

Multiple

In parent

A screenshot of a computer program

AI-generated content may be incorrect.

In app-card

A screenshot of a computer program

AI-generated content may be incorrect.

Lifecycle hooks

Special methods provided by angular that allow us to tap into different stages of a component’s lifecycle. Angular provides hooks that let us run custom logic at specific times.

Constructor(not part of hook lifecycle)

Called before ngOnInit, first method called, view is not ready yet

ngOnInit

called only once when the component is initialized

ngOnChanges

called every time an input **property changes**

A computer screen with text

AI-generated content may be incorrect.

ngDoCheck

called during every change detection run, even if nothing changes

ngAfterContentInit

called once after the component’s content has been initialized

after ng-content (content sent from parent) is rendered, to work with that conten after it has been projected

ngAfterContentChecked

called after every check of the component’s content, like ngDoCheck but for projected content of ng-content

ngAfterViewInit

called only once after the component’s view (DOM) has been initialized

ngAfterViewChecked

called after every check of the component’s view, triggered after every time the view changes

ngOnDestroy

called just before the component is destroyed

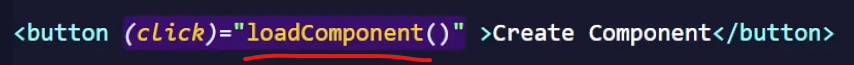
Load component inside div programatically with ngComponentOutlet

<div \*ngComponentOutlet=”loadComponent()”></div>

A blue and yellow text

AI-generated content may be incorrect.

Load component when button is clicked with ViewContainerRef



A screen shot of a computer code

AI-generated content may be incorrect.

Remove component from ViewContainerRef



A yellow text on a black background

AI-generated content may be incorrect.

Pipes

A way to transform data directly in the template

Built in pipes, angular pipes out of the box

Custom pipes, let you define your own transformation

Data -> Pipe transform -> Formatted data

Cleaner, so the component does not have that code and it can be reused in different places

Ex: built in uppercase pipe and number pipe





A purple sign with yellow letters

AI-generated content may be incorrect.









Json pipe to display complex object

{{ user }} -> display [object Object]

{{ user | json }} -> {“name”:”Fer”, “age”:39}

A computer code with green and orange text

AI-generated content may be incorrect.->

Multiple pipes



Services

By dependency injection one single instance is used instead of using a new instance each new is used

A screen shot of a computer

AI-generated content may be incorrect.

Injector

Area where all instance services are stored

Optional fields with ?, not mandatory

A computer screen shot of a computer code

AI-generated content may be incorrect.

Template Driven Form

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

A computer screen with text

AI-generated content may be incorrect.

Reactive Forms (setup and validation in components)

Form group ex:

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Nested Form Group Ex:

A computer screen shot of a program code

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Reactive Form Array inside a FormGroup

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Reactive Form Builder

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

Custom Validator

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Routing

Router link

Directive to go to a route



Router-outlet

A <router-outlet> in Angular is a placeholder directive that tells the router where to dynamically load and display different components (views) based on the current URL

Href vs router link

Href reloads the page, router link doesn’t reload the page just loads the component at the router-outler tag

Routerlinkactive

Adds a css class when the route is active

Route With parameter

A screen shot of a computer

AI-generated content may be incorrect.

ActivatedRoute to receive parameter from view router link

A screen shot of a computer program

AI-generated content may be incorrect.

Observable

A screenshot of a computer

AI-generated content may be incorrect.

Subscribe() calls the observable as If it was a function

A screen shot of a computer program

AI-generated content may be incorrect.

Next() allows to return a value from observable

A screen shot of a computer

AI-generated content may be incorrect.

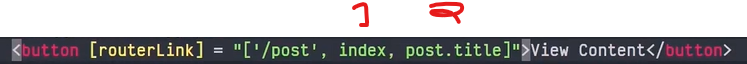
Any change on the observable and the subscribe method will trigger

Keeping subscribe will spend resources of memory, you can unsubscribe to stop listening

A screen shot of a computer program

AI-generated content may be incorrect.

Multiple router parameter



Query parameters



A screen shot of a computer code

AI-generated content may be incorrect.

Angular Routing File

A screen shot of a computer program

AI-generated content may be incorrect.

Navigate to a route programmatically from a component



A close up of a logo

AI-generated content may be incorrect.

With parameters and query parameters



Wildcard

A screen shot of a computer program

AI-generated content may be incorrect.

Rxjs

Angular Testing

Jasmine

In jasmine tests are known as specifications or specs which are grouped in test suites.

Pending() test not ready to be executed

Ng test, compiles and runs our tests specifications by the karma test runner used by angular internally, lanch a browser to test with specs ran and more information

A screenshot of a computer

AI-generated content may be incorrect.

Basic ex:

A screenshot of a computer program

AI-generated content may be incorrect.

Jasmine spies

Keeps track of an object to share information of for example times a method was called

A computer screen shot of a computer program

AI-generated content may be incorrect.

In general when creating specs we just want to create an instance of the validated class, for example a service, and all dependencies required by this instance should be mocked instead of real instances, that way we make sure if the spec fails is because of changes in the actual asserted class

A screenshot of a computer program

AI-generated content may be incorrect.

BeforeEach

Called before each of the specs:

beforeEach(() => {

loggerSpy = jasmine.createSpyObj(‘LoggerService’, [‘log’]);

calculator = new CalculatorService(loggerSpy);

});

Inject a service

Using TestBed allows to get providers, imports using dependency injection

Describe(‘CalculatorService’, ()=>{

Let calculatorService:CalculatorService,

loggerSpy:any;

beforeEach(() => {

loggerSpy = jasmine.createSpyObj(‘LoggerService’, [‘log’];

TestBed.configureTestingModule({

Providers: [

CalculatorService,

{provide: LoggerService, useValue: loggerSpy}

]

});

calculatorService = TestBed.inject(CalculatorService);

})

});

Disable all specs by adding x to the describe:

Xdescribe(‘CalculatorService’, () => { …

Disable test by adding x before it

Xit(‘should add 2 numbers’, () =>{…

Only execute current specs by adding fdescribe, will only execute this specs on ng test

Fdescribe(‘CalculatorService’, () => {…

Only execute 1 spec in the test suite

Fit(‘should sum 2 numbers’, () => {…

Testing Http Services

While using the mock HttpClientTestingModule, when the call to the service is done it is not returning the subscribe method results, until the request flush method is called which adds the data to be returned by the request and used in the subscribe method

beforeEach(() => {

TestBed.configureTestingModule(() =>{

Imports:[HttpClientTestingModule], //simulate http calls

Providers: [CoursesService]

});

coursesService = TestBed.inject(CoursesService);

httptestingController = TestBed.inject(HttpTestingController)

});

It(‘should retrieve all courses’, ()=> {

cousesService.findAllCourses().subscribe(courses => {

expect(courses).toBeTruthy(‘no courses returnd’);

expect(courses.length).toBe(12,’Incorrect number of courses’);

});

Const req = httpTestingController.expectOne(‘api/courses’)

Expect(req.request.method).toEqual(‘GET’);

Req.flush({payload: Object.values(COURSES)}); //request is going to return this object to the subscribe method as a simulated response, subscribe method will not trigger until calling the flush method

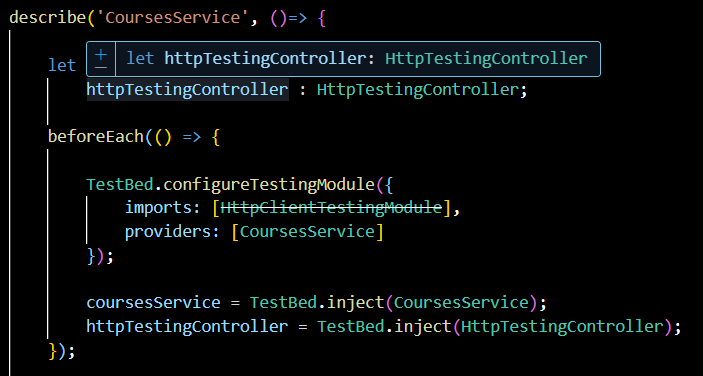
}

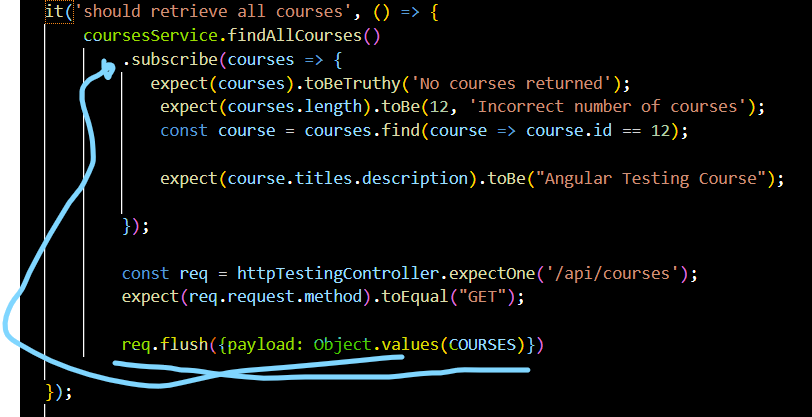
Assert no other http methods were called unintencionally

httpTestingController.verify() (at the end of the test, could use afterEach for this) //this will assert only those explicitely called with expectOne were called, If any other not specified was called this will fail

ex:

configure Test bed







Component View Testing

let el: DebugElement

el = fixture.debugElement;    //gives access to the DOM elements of the component

if changes on the view had happen we need to make sure it is reflected on the view

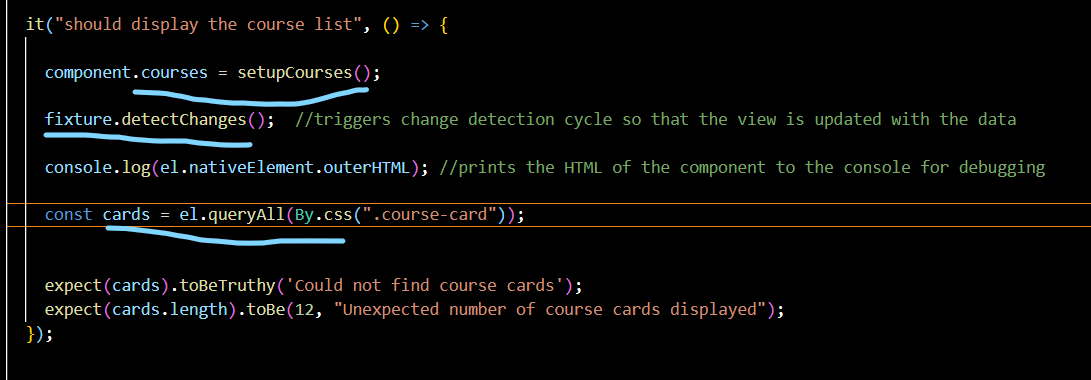
fixture.detectChanges();  //triggers change detection cycle so that the view is updated with the data

print current dom

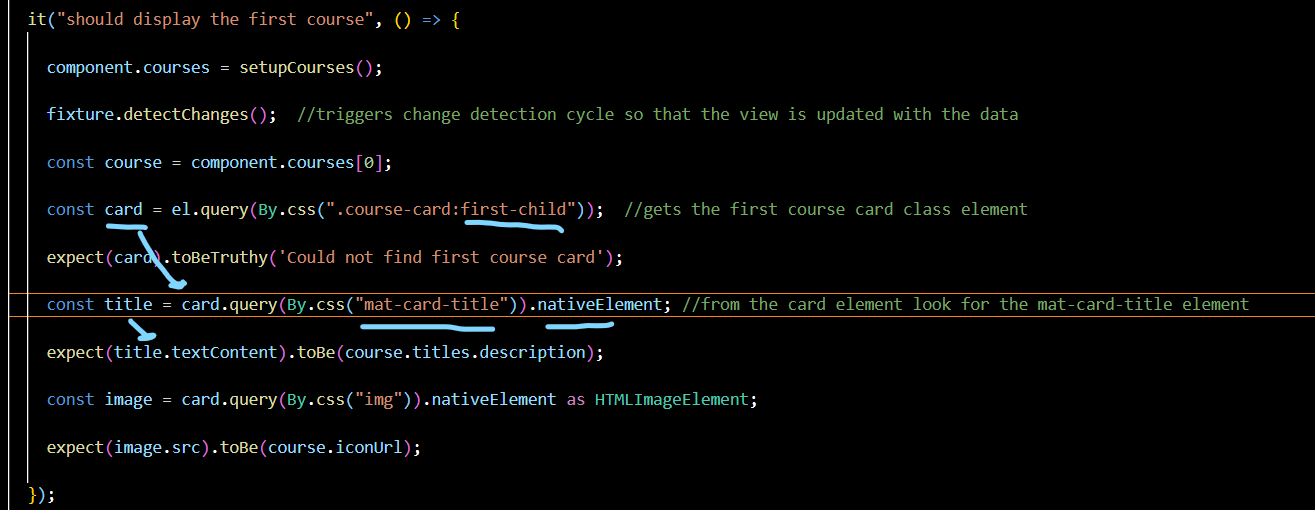
console.log(el.nativeElement.outerHTML); //prints the HTML of the component to the console for debugging

get elements in the view by css class

const cards = el.queryAll(By.css(".course-card"));



Get first element of dom by class



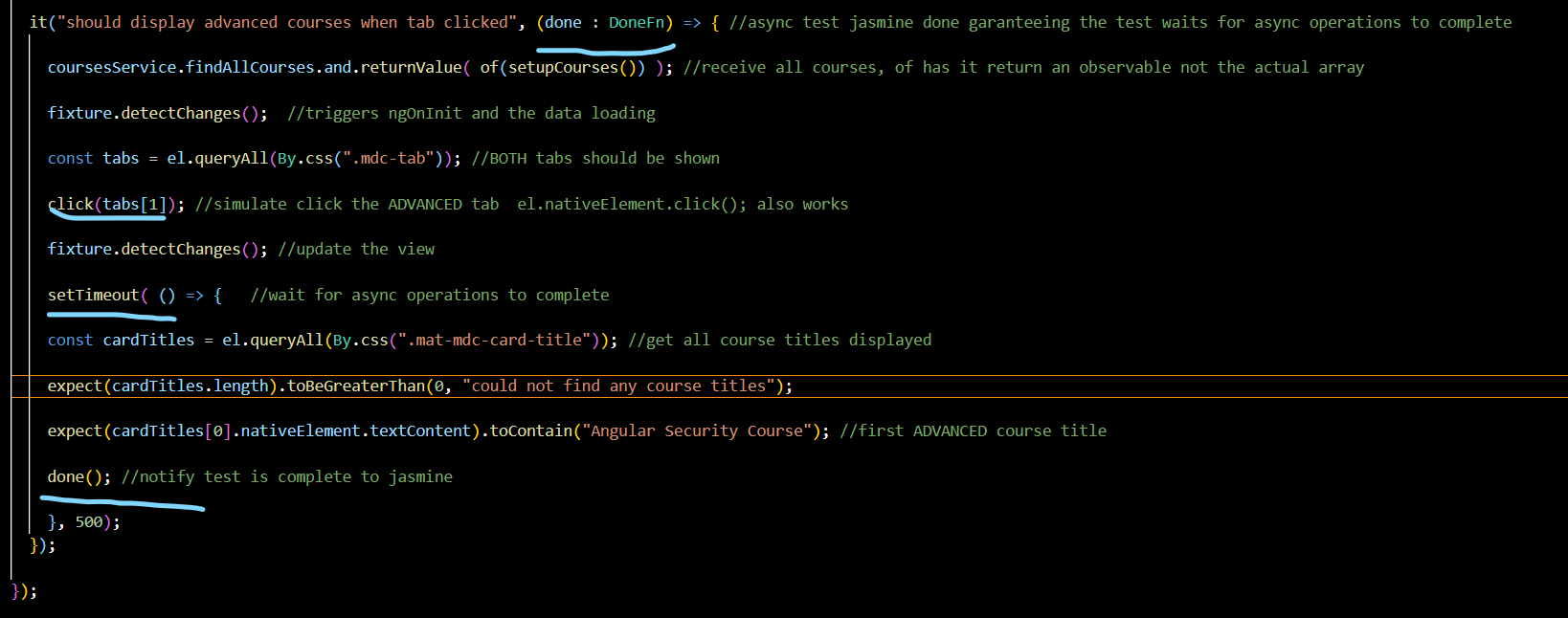
Container Component

The container component acts as the data orchestrator, fetching information from the backend via services and managing the application state. It then passes this data down to its child presentation components.

Presentation Component

The presentation components, in turn, simply render the UI based on the data received via @Input and emit events via @Output for any user interactions, which the container handles.

Async test example, not ideal for multiple asynch operations test (by using set timeout)



Angular fakeAsync

fakeAsync runs the code in a special fake async zone that simulates async operations and confirms all async operations complete before moving on

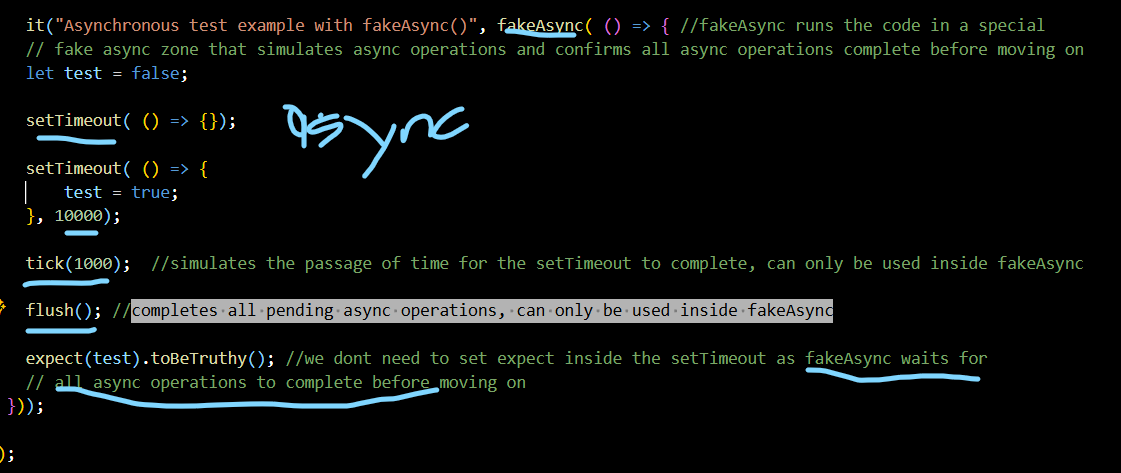
tick(miliseconds);

simulates the passage of time for the setTimeout to complete, can only be used inside fakeAsync

flush();

completes all pending async operations, like setTimeout, setInterval, Ajax calls, also microtasks like promises, can only be used inside fakeAsync

ex:



Angular zone

An Angular Zone is a core part of Angular's automatic change detection, using the zone.js library to intercept asynchronous browser APIs (like setTimeout, Promises, XHR) to know *when* to check for UI updates, keeping the view in sync with the application's state without manual checks. The NgZone service runs within a dedicated "Angular Zone" (a child of the global zone) and signals when asynchronous tasks complete, telling Angular it's time to run change detection

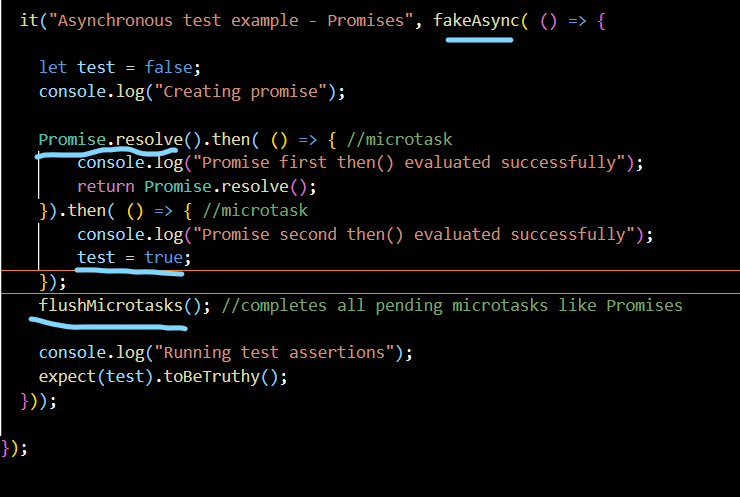
Angular plain promise

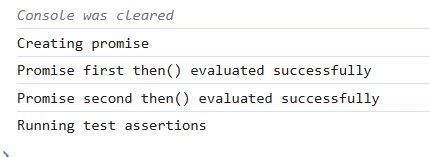
In Angular, a Promise is a built-in JavaScript object used to handle a single asynchronous operation that will eventually complete with a single value (or an error). While Angular primarily leverages RxJS Observables for most asynchronous tasks (like HttpClient requests), Promises are a simpler, valuable tool for specific scenarios, such as single-value API calls or integration with third-party libraries.

Promises are considered micro tasks, so it will be executed before normal tasks like setTimeout, setInterval, ajax calls etc

Flush Microtasks

Executes microtasks (like angular promises prior continuing the flow)

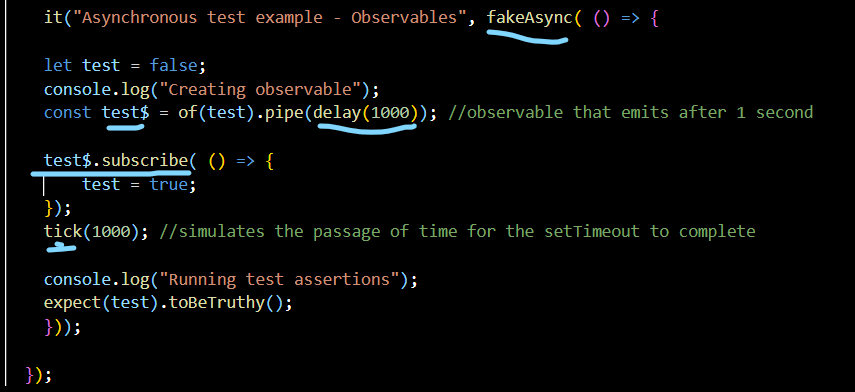




Mixed microtasks (promises) and macrotasks (setTimeout)



Example with observables



Async (waitAsync) vs fakeAsync

waitAsync supports actual http request

fake async has access to extra funcionalities with the microtasks and tasks like flush, flushMicrotasks, tick which are not available in waitAsync

End to End Tests

Use of a set of components to validate app functioning, how everything fits together

No mocking, actual components.

Cypress, library that can be used for this, it is not angular specific, it could be run for react as well for example

Ex:

Describe(‘Home Component e2e’, () => {

It(‘should display a list of courses’, () => {

Cy.fixture(‘courses.json).as(“coursesJSON”); //files and alias reponse

Cy.server(); //simulate http back running

Cy.route(‘/api/courses’,”@coursesJSON”).as(“courses”); // assign the response to a path

Cy.visit(‘All courses”); //validates the visited route has the string

Cy.wait(@courses); //wait for the data to come from route

Cy.get(“mat-card”).should(“have.length”,9);

})

});

MONGO DB

Relational vs document db (no sql databases)

Tables with fixed schema vs documents

All information in 1 document, no need to traverse multiple documents in nosql

Information across multiple tables in sql databases, require joins to combine the information

No sql no fixed schema

Mongo db has a coleccion of databases, each database can have multiple collections and each collection can have multiple documents

Mongod command to start mongo server, you can create documents in mongo server

Mongo command start mongo shell, you can use the shell to execute crud operations to the mongo server

Eureka Server

Registry

Microservices can register themselves, eureka maintain a list of available services

Discovery

Clients (other ms) can query to dinamically discover the services they need

Fault Tolerance

Eureka server continuesly monitors thealth of registered services and automatically remove unhealthy services from the registry

Client Side discovery

Each client is aware of the service registry and is responsible for locating and communicating with the desired service.

Server Side Discovery

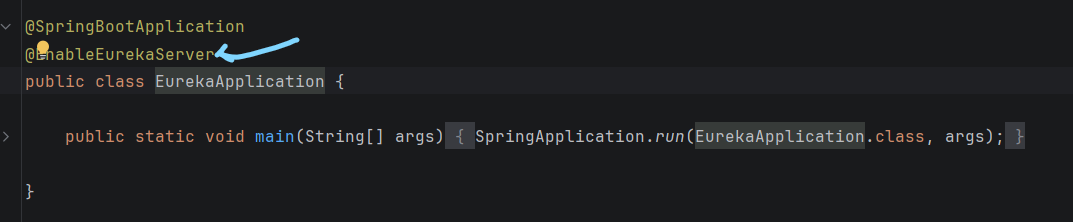
Responsibility lies in the service registry or discovery server, acts as a centralized repository where services register themselves when they start up.

Client requests to this registry to obtain addresses of the desired service

Registry performs load balancer and routing based on how they were configured

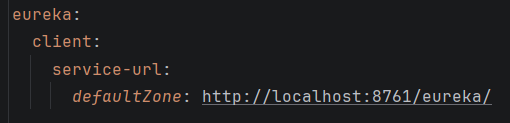
Eureka server:



This is the server config for the eureka server, @EnableEurekaServer annotation is needed at the top of the app

Eureka Client

Needs the url to the eureka server in the application.properties file, this client will be registered into the eureka server



@GeneratedValue

Creates a table hibernate\_sequence which would maintain the order of the next id for the property annotated with it



Default value to a entity column by using @Column



@LoadBalanced with Eureka



When you mark a registered eureka client with @LoadBalanced, Eureka internally will choose the instance which fits better in the balancing criteria, it will return a load balanced instance of the rest template in this case

Calling a registered eureka service from another ms

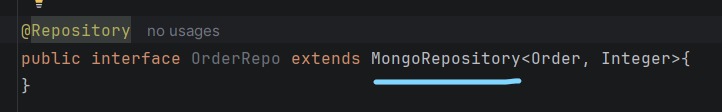


Instead of harcoding the port of the ms, using the service name as it is registered in eureka will make it point to the right balanced instance of the service, in case there are multiple instances no need to use its direction, with the service name eureka will point to the one fits better according to balance

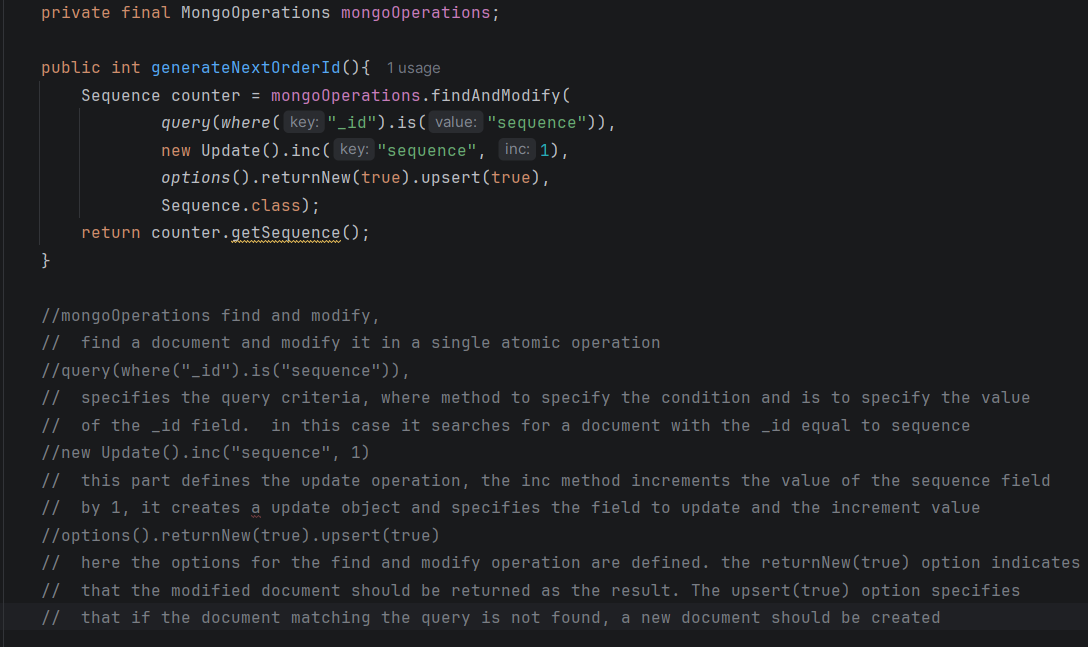
Entity for Mongo DB using @Document instead of @Entity



Mongo Repository instead of JpaRepository



Mongo sequence object example



Typescript is javascript with additional features

Object Orientation

Strong type checking

Compile time error checks

It is compiled to javascript in the end

Node Js

open source runtime environment for executing js code

Angular Cli

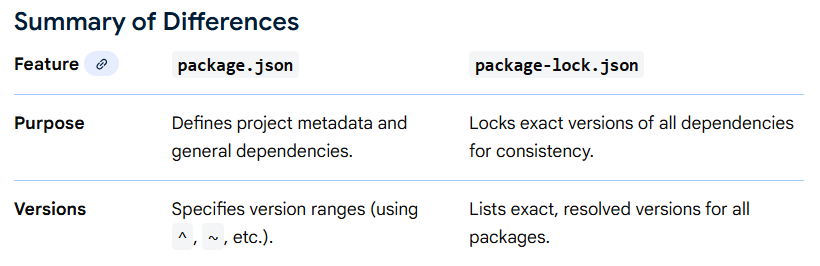
Command line interface tool used to initialized, develop and maintain angular apps

Package.json ^ and ~

* **Caret (^)** is the default for new installations in modern npm versions. It offers more flexibility, allowing projects to receive new, backward-compatible features and bug fixes automatically (Minor and Patch updates). This helps keep your project up-to-date with a lower risk of breaking changes.
* **Tilde (~)** is more conservative and prioritizes stability by only allowing bug fixes (Patch updates). This is useful for critical dependencies where introducing any new feature, even a backward-compatible one, carries a higher risk of unexpected behavior.

Package-lock.json

The primary purpose of the package-lock.json file is to lock the entire dependency tree (including sub-dependencies) to exact versions to ensure consistent installs across all environments (development, CI/CD, production).



App.module.ts

The app.module.ts file in an Angular application defines the root module (AppModule), which is the entry point that tells Angular how to assemble and bootstrap the application.

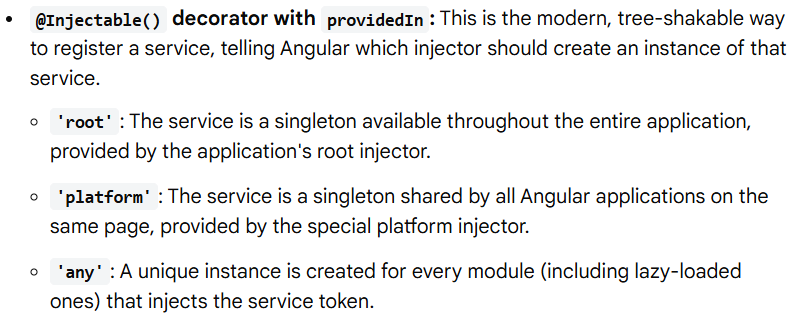
Main.ts

The main.ts file is the **primary entry point** of an Angular application. It is the initial TypeScript file that the application's build process looks for to start the application in the browser.

Index.html

In an Angular application, the index.html file is the **single host page** that serves as the entry point for the entire Single Page Application (SPA). It is a standard HTML file that contains minimal initial content and a specific placeholder tag where Angular dynamically loads the rest of the application.

Angular Services have @Injectable metadata



Injector module, injector root, injector component, for each new component new instance

Server Side Rendering

Angular server-side rendering (SSR) is a technique where an Angular application's initial HTML is generated on the server for each request, then sent to the browser. This approach dramatically improves performance metrics like initial page load time and enhances Search Engine Optimization

Static Site Generation

Angular supports **Static Site Generation (SSG), also known as prerendering,** as a built-in feature to improve performance and SEO. This process pre-renders pages into static HTML files during the build process, which are then served as static content, eliminating the need for a dynamic server for initial page loads.

Angular.json file

This file acts as the control room for the Angular CLI, instructing it on how to execute commands like ng build, ng serve, ng test, and ng generate

In angular String interpolation only for string type

{{ aString }}

Angular route when using modules:

App-routing.module.ts (root routing)

Const routes: Routes = [ { path: ‘’, redirectTo: restaurant-listing’, pathMatch: ‘full’}

This will route you from the root module into a module restaurant-listing

Now inside the module restaurant-listing there is also a routing module where you need to redirect the path:

Restaurant-listing-routing.module.ts (where root module redirect to)

Const routes: Routes = [ { path: ‘’, component: RestaurantListingComponent}

Cross Origin Issue example:

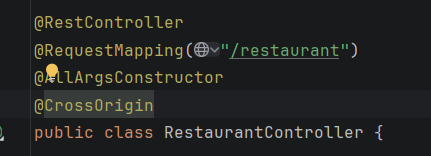
Your Angular app (e.g., http://localhost:4200) is trying to reach a backend API on a different port or domain (e.g., http://localhost:3000 or <https://api.example.com>).

An Angular cross-origin error occurs when your web application, running on one origin (domain, scheme, or port), attempts to access resources from a different origin, and the remote server does not explicitly grant permission. This is a browser security feature known as the **Same-Origin Policy**, which is enforced to prevent malicious sites from accessing sensitive data.

**In Production (Server-Side Configuration)**

For a deployed application, the backend server must be correctly configured to allow requests from your frontend's domain.

* **Configure your Backend:** Add the required Access-Control-Allow-Origin header to your server's responses, specifying the exact origin of your Angular app (e.g., https://myapp.com).
  + **Node.js/Express:** Use the cors middleware.
  + **Java/Spring Boot:** Use the @CrossOrigin annotation or a CORS filter.



Angular tsconfig file

The tsconfig.json file in an Angular project specifies the root files and the compiler options required to compile the TypeScript code into JavaScript.

Angular Activated route to get the path parameters

Angular Router to route to a location

Docker

Docker is an open-source platform that uses **containerization** to package applications and their dependencies into standardized, isolated units called containers. This ensures that applications run consistently across different environments, from a developer's laptop to production servers.

Container

A container is a runnable instance of a Docker image. It is a lightweight, isolated environment that runs as a separate process on the host machine but shares the host's operating system (OS) kernel. This design makes containers more efficient than traditional virtual machines (VMs), which require a full, separate guest OS.

Image

An image is a read-only template that provides the blueprint for creating a container. It includes the application code, runtime, libraries, dependencies, and configurations needed to run the application. Images are built from a Dockerfile.

Docker File

A text file that contains a sequence of instructions for automatically building a Docker image. It specifies the base image, necessary installations, environment variables, and commands to run when the container starts.

Common instructions:

FROM: Specifies the base image (e.g., ubuntu:22.04).

WORKDIR: Sets the working directory within the container (e.g., /app).

COPY: Copies files from the host to the container (e.g., COPY . .).

RUN: Executes commands during the image build (e.g., RUN apt-get update).

EXPOSE: Documents the ports the container will use at runtime (e.g., EXPOSE 8000).

CMD: Defines the default command to run when a container starts (e.g., CMD ["flask", "run"]).

Registry

A centralized repository for storing and distributing Docker images.

Docker Hub

 default public registry, where users can find thousands of pre-built images or store their own public and private images.

Docker Engine

The core client-server technology that builds, runs, and manages containers.

Docker Compose

A tool used to define and run multi-container Docker applications. It uses a YAML file to configure application services and manage the interconnected containers with a single set of commands.

[Docker Compose](https://docs.docker.com/compose/) es una herramienta de Docker que define y ejecuta aplicaciones multi-contenedor usando un único archivo docker-compose.yml, simplificando la gestión de servicios, redes y volúmenes para entornos de desarrollo y prueba, permitiendo levantar o detener toda la aplicación con un solo comando como docker compose up o docker compose down.

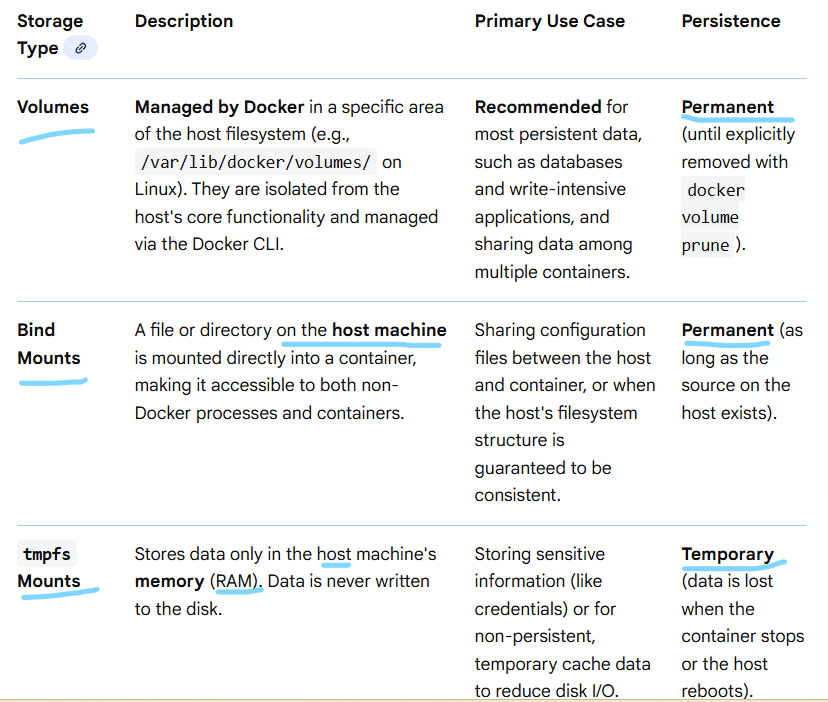
Docker File -> docker Build -> Docker Image -> docker Run -> Docker Container

Docker containers are volatile, if you remove or kill the container all of its data will be lost

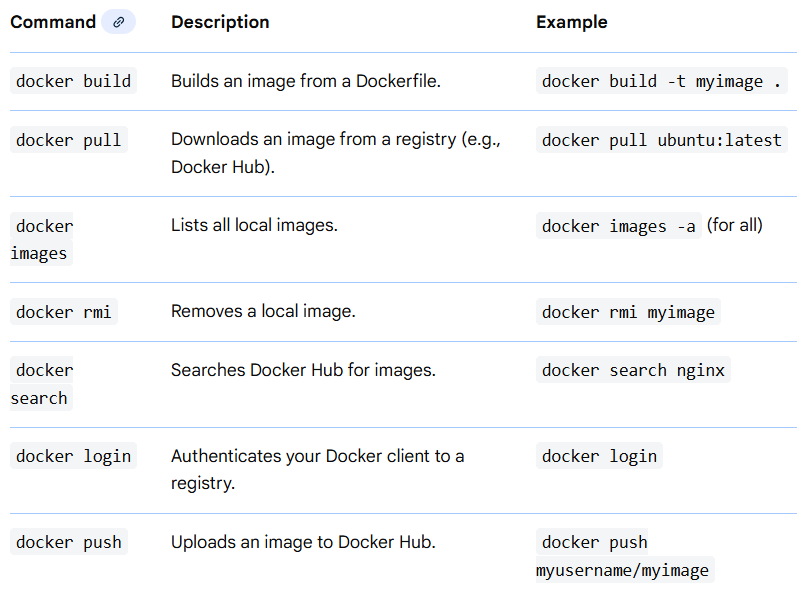
Docker Storage

Docker storage refers to how the Docker engine manages the filesystems for images and containers, and how data can be persisted beyond the lifespan of a container. Data created within a container's writable layer is temporary by default; to store data permanently, Docker offers **volumes** and **bind mounts**.

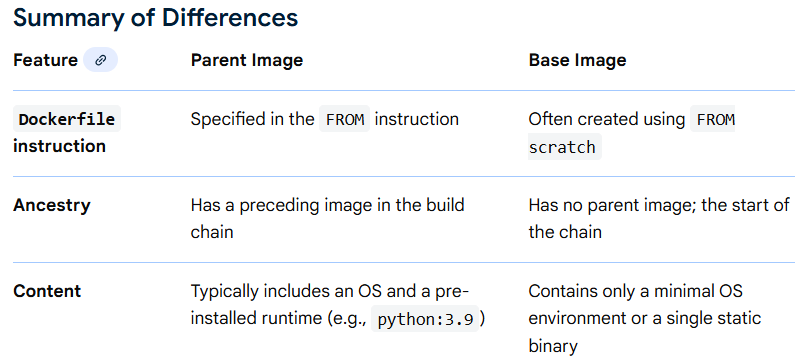
For Data that needs to persists across container restarts or be shared across containers docker provides these options:



EXPOSE and ENV PORT from docker file are not mandatory when using Kubernetes since those will be added in the Kubernetes Service Configuration



Docker Base image vs parent image



Docker from scratch

Docker from scratch" typically refers to building a Docker image using the special, empty scratch base image, or it could mean learning Docker conceptually from the very beginning.

his approach is used to create extremely small, secure containers, typically for **statically compiled binaries** (like those written in Go or C) that have no external dependencies.