para gerar um arcabouço do projeto em angular, deve ser executado o seguinte comando na pasta onde ficará o projeto:

ng new 'nomeprojetosemaspas' --prefix='algumprefixo'

nome do projeto é óbvio, o prefix será algo que utilizado na frente de cada componente que será criado

após isso, deve-se esperar xD

apareceu duas perguntas:

Would you like to add Angular routing? y/N

Which stylesheet format would you like to use?

Tem como escolher uma opção default, que não pergunta isso

escolhi yes e css

padrão

depois da um:

ng serve

Se compilar e funcionar, no endereço localhost:4200 terá uma página inicial bem nice

da pra rodar as configs default, só procurar

====================================================================

o que verei?

{

principais arquivos gerados pelo angular cli (ng new),

processo de como é feito o startup da aplicação,

o que o webpack ta fazendo, o que é isso?,

criação do primeiro componente,

}

primeiramente

o que é criado? um projeto que segue a estrutura do node

com o package.json , que é um arquivo de configuração com as dependências, alguns scripts etc.

algumas dependências de dev

outro arquivo importante de se conhecer

-> angular-cli.json (não apareceu no meu projeto, mas tudo bem, aparentemente é meu arquivo angular.json)

neste arquivo pode ser customizado algumas coisas relacionadas ao projeto

como:

arquivos de stylo usados na aplicação

arquivos de script usados na app

assets principais

temos a index

a main (arquivo principal)

o que é webpack?

é uma biblioteca em javascript responsável por criar bundles na aplicação, ele compila os arquivos e separá-los em bundles específicos que são mostrados no console:

quais os bundles específicos?

polyfills ~> série de scripts adicionados para aumentar a compatibilidade com browsers antigos

main ~> scripts da aplicação

styles ~> estilos que estão sendo utilizados dentro da aplicação, todos arquivos css incluídos

vendor ~> tem scripts de terceiros

e o próprio script responsável por carregar o webpack no browser, responsável por carregar toda estrutura e fazer que o angular seja iniciado no navegador

o webpack carrega esses scripts no browser, e a mágica começa a acontecer

====================================================================

bootstrap da aplicação

como o angular é statado?

tudo começa no ng serve (ou npm start) onde é disparado um servidor interno do webpack utilizado apenas para dev (chamando live development server)

a partir disso o webpack olha os arquivos da aplicação, começando do main.ts e investiga todos os outros arquivos, atrás de todos imports, scripts e dependências para fazer um mapeamento de tudo que é usado e separar isso nos bundles que serão utilizados pela aplicação.

Uma vez separado esses bundles, quando é pedido pro server a index.html, o servidor injeta esse bundles no final do arquivo index.html

Se parar para olhar o arquivo index.html, pode ser visto que não tem referência para arquivo css e também não tem para arquivo script, porém no browser tem uma série de referência a scripts no final do arquivo

Quando o web pack injeta esses scripts de forma dinâmica, o browser recebe esse html com esses scripts injetados e starta a execução desses scripts

o main.bundle.ts starta o bootstrap da aplicação

quais os detalhes que é preciso conhecer?

o main.bundle.js(no caso dele, o meu é só main.js) carrega o conteúdo do src/main.ts, arquivo responsável por fazer o bootstrap da app

Como está organizado o main.ts?

primeiro vem os imports, começando com uma função capaz de habilitar o modo de produção, isso modifica uma série de coisas, por exemplo, tira a possibilidade de fazer debug,

tem uma outra função chamada platformBrowserDynamic essa função é importada sempre que está sendo iniciado o angular a partir do browser

tem o módulo principal, que é gerado pelo angular cli

tem uma constante que ajuda a web pack a determinar se a aplicação está em modo de execução ou não ( import { environment } from ... )

o que realmente starta a aplicação é a seguinte linha

platformBrowserDynamic().bootstrapModule(AppModule)

.catch(err => console.error(err));

É possível ver nessa linha que é chamado o AppModule, pelo import dá pra ver onde esse arquivo está e navegar até ele

Dando uma olhada nele:

Esse arquivo é um módulo em angular (não confundir com módulo do es2015)

módulo do es2015 é uma forma de declarar o que vai ser exportado!

já o módulo em angular, agrupa uma série de componentes da aplicação ou do próprio framework, como por exemplo no arquivo app.module.ts

ele tem uma classe, e um conjunto de metadados aplicados a essa classe

vendo esse arquivo de cima para baixo

alguns imports, como o browserModule, conjunto de dependências básicas do framework, deve ter esse módulo pois será executado no browser, angular cli da video aula também importa o formsModule e o HttpModule, pois são muito usados na aplicação

também é importado o NgModule, mas o que é esse módulo? NgModule é um decorator, uma função que serve para aplicar metadados em uma classe, atributo, em um método ou em um parâmetro de um método. No exemplo do arquivo app.module, esses métodos são aplicados na class AppModule

Então são adicionados informações ao módulo

Essas informações são:

declaration: componentes dentro de uma aplicação

imports: o que é preciso usar, quais são as dependências

providers: será explicado depois

bootstrap: qual o componente dentre os componentes q é o responsável pelo bootstrap da aplicação, nesse campo vem o AppComponent que é gerado pelo angular cli

Como é a estrutura desse component?

app.component.ts

É uma classe que também importa outras coisas, no caso, outro decorator

que é uma função que aplicar metadados a classes atributos, métodos ou argumentos. No caso desse decorator em especifico, ele atribui metadados específicos para componentes

Como o selector: se eu quiser usar um componente em uma página, como eu chamo?

geralmente, nesse arquivo é o prefixo-root (prefixo definido na construção do projeto)

outros atributos: templateUrl ~> qual a cara que vai ter o meu componente

stylo: como eu vou personalizar isso

obrigatórios: template e selector, o template pode ser inline também

Tendo uma ideia

Toda aplicação Angular terá pelo menos um módulo(root) + um component (component de bootstrap)

como não vamos usar nesse momento o stylo, pode deletar o arquivo

pode deletar também o arquivo app.component.spec.ts, pois ele é relacionado a testes, que não serão usados agora

ao inspecionar o html index já gerado, é possível ver o select definido nesse arquivo

no meu caso <gg-root>

Se tiver algo dentro dessas tags, ele será mostrado antes da aplicação carregar, no caso, o conteúdo daquele component será colocado no lugar da tag

o que acontece? o bootstrap carrega o arquivo main que carrega o módulo principal, este diz qual o componente de bootstrap e o angular pega o template do component bootstrap e coloca no na página onde está a tag correspondente ao component

Observando a cara do component principal, pode-se ver que no título

vai ser carregado dinamicamente com algum conteúdo do component

====================================================================

Criando o primeiro component!

O que é um component?

pequenas partes independentes e reutilizáveis!

Em angular, são classes que tem um determinado ciclo de vida, possuem um template para definir a aparência e um selector que é a tag, para ser usada por outras partes da aplicação.

Components portanto são elementos personalizados

Uma página vai ser composta por partes menores (components) que podem funcionar de forma independente e podem comunicar entre si através de eventos

O framework prega que components são dispostos em estrutura de árvore onde a raiz

há sempre um component parent com seus filhos

Como definir um component?

A classe de um component é uma classe que segue a sintaxe do ES2015 mais as features do TS

Sempre que declaramos a classe marcamos ela com a palavra chave export para ela ser referenciada depois em outros arquivos de configurações exigidos pelo framework, portanto, quando essa classe é marcada com o export, essa classe vira um módulo do ES2015, mas existem outras coisas que devemos informar ao angular, como: qual a tag que devemos usar para usar esse component?

Para isso, é usado o decorator @Component

Nesse decorator, devemos informar o selector, que será a tag e qual o template que o component vai ter. esse template pode ser um arquivo externo através do atributo templateUrl, essa url pode ser relativa, absoluta ou uma url http

a segunda forma é declarar o template direto no arquivo usando o atributo template, sem o url, segunda forma é recomendada apenas se o template for pequeno e simples

Esses templates podem ter expressões que resolvem as propriedades do component, e isso é chamado de template interpolation (parece EL)

templates também podem acessar propriedades mais complexas usando o operador ponta, como de um objeto

Uma vez criado o component, é necessário informar ao angular em qual módulo angular o componente pertence

modulo ES2015 !== Módulo Angular

modulo angular = é responsavel por saber quais components, serviços, pipes e diretivas fazem parte da aplicação, e cada um desses precisa estar em um modulo. Se a declaração for feita no modulo raiz, toda a aplicação tem acesso a aquele componente

modulo angular é aquele declarado com a anotação

@NgModule({

declarations: [MyComponent]

})

export class AppModule {}

NgModule, a'ng'ular module, hehe, bem obvio

====================================================================

Como criar um component pelo console?

na pasta raiz do projeto utilizar o comando

ng generate component 'nomecomponentsemaspas'

ou

ng g c 'nomecomponentsemaspas'

atenção no comando que será usado de parâmetro

--spec=false, para indicar que não queremos o arquivo de testes desse component

então o comando completo segue:

abreviado é cool

ng g c header --spec=false

ao rodar o comando, uma nova pasta será criada com o conteúdo do component

ou seja, o component, o template e o stylo (css)

O angular cli faz uma coisa muito importante, sempre que é criado um component é preciso listar este em um módulo, por enquanto só tem um modulo (modulos serão explicados mais a frente no curso), este é o módulo raiz, o modulo raiz contém a declaração dos nossos components, até aula passada só tinha o AppComponent, aquele component de inicialização, agora terá também o HeaderComponent criado com o angular cli.

Agora, verificando a classe do component gerado

Ela é simples, basicamente igual ao componente analisado primeiramente

tem a mesma anotação @Component, que é um decorator, com os param selector, templateUrl, styleUrls

a diferença, é que agora a classe tem uma implementação

Nela, existe uma interface sendo implementada chamada OnInit, que contém um método caso eu queira colocar alguma funcionalidade na inicialização do componente. Ela obriga a implementar o método ngOnInit(){} que será chamado pelo menos uma vez durante o ciclo de vida de um component

Tá bom, criado o component, e agora?

Agora usamos, já que ele está no mesmo contexto do módulo raiz da aplicação

podemos colocar uma tag indicando o uso daquele módulo no html do component inicializador

Mas qual a tag?

Só olhar no component criado, campo select

Podemos estilizar o component com o css específico criado e setado pela classe, só ir no arquivo com final css

====================================================================

Property binding

O q diabos é isso? é quando você deseja linkar o valor de uma propriedade de um elemento a uma expressão angular q pode avaliar para uma propriedade de um componente, método ou expressão mais elaborada

Sua sintaxe é com colchetes em volta do elemento que você deseja atribuir um valor []

e pode ser aplicada a qualquer propriedade do elemento do dom

um exemplo

Vamos imaginar que temos um objeto lindinho \*--\*

let user = { nome: 'Geraldão' }

vamos imaginar que eu quero linkar o valor da propriedade desse objeto a um elemento do dom

Se eu quiser linkar essa prop com a propriedade value de um elemento input

use:

<input type="text" [value]='user.nome'>

Quando o valor da propriedade mudar, o input também irá mudar.

Nesse caso, como a atualização é sempre em um sentido se chama one-way binding

One-way binding é sempre do componente para o template

Agora ele tá explicando elementos do dom, me abstenho de anotar sobre

outro exemplo sobre o binding

imagina que adicionamos outra propriedade ao objeto

let user = { nome: 'Geraldão', isMago:'Mago'}

se existisse uma área onde somente magos podem ter acesso, a seguinte sintaxe pode ser usada

<input type='text' [value]='user.nome'>

<div [hidden] = '!user.isMago'>

location of the mago templete

</div>

também é possível pegar alguma expressões e associá las a uma condição booleana, se uma var for verdadeira o angular associa uma classe a uma certa tag do dom

exemplo:

<div [class.light]="user.isMago"><div>

~> Se for true

<div class="light"></div>

====================================================================

Voltando ao código, mostrando o property binding no código

lembrando

property binding é uma forma de linkar uma propriedade, um atributo de um component para uma propriedade de um elemento do dom, no template daquele component

pode-se fazer isso com elementos da especificação HTML como também components que serão criados

primeiro fazendo binding com components existentes

para esse exercício, primeiro deve criar um novo componente

criado o novo component, iremos adicionar atributos ao component criado

como:

name: string = 'gege';

isMago: boolean = true;

tendo isso, usamos essas propriedades do component no template

<div>

O mais foda do baile {{name}}

<div [hidden]="!isMago">

Geraldo é um mago

</div>

</div>

Aqui estamos usando template interpolation e também o property binding

div torna-se oculta se não for mago

usando isso no template principal, aparece aquele div só se a propriedade mago for true, se não dá ruim

======================================================================

Algumas considerações

component basicamente é uma classe exportada com alguns metadados inseridos pela a notation decorator @Component

esse component deve estar declarado em algum módulo angular, tendo em vista que módulo angular é onde se declara os components, alguns outros imports e um component inicializador, o component, como classe, pode ter alguns atributos, e esses atributos podem ser usados no template desse component, via property binding ou template interpolation

====================================================================

Como passar valores pra propriedade de components?

Nessa aula vai ser falado como passar valores do dom para os components (que são classes plus)

As propriedades dos components são públicas, porém é necessário informar ao angular quais dessas propriedades podem receber dados dos seus components parents através do property binding

Ou é listado o atributo que queremos expor na lista de inputs do decorator @Component

ou

Ou marca o atributo com outro decorator chamado @Input

Agora começa a brincadeira

Quando marcamos um atributo com esse decorator, estamos indicamos que podemos passar o valor para o atributo marcado a partir de outro component

como assim gg?

imagina o lance: marcamos um atributo de um component com esse decorator

blz!

ai quero q esse atributo seja variável para cada lugar que vou usar ele

Logo, quando eu for usar, coloco lá o q eu quero q o atributo receba

Lembrando que essa decorator @Input pode receber parâmetros, então devemos usar com parenteses.

fica mais ou menos assim

@Input() tittle: string;

Lembrando também que é preciso importar esse decorator do @angular/core

O parâmetro recebido pelo decorator input é uma renomeação do atributo

Quando eu for usar esse component dentro de component parent, uso lá no template dele o select, tudo bem certinho, e coloco um atributo (html) com o valor igual ao do atributo do component ou o nome que passei pelo decorator input

lembrando que também posso utilizar isso com template interpolation ou property binding

abaixo segue exemplos:

<!-- template interpolation -->

<gege-student tittle="{{!isMago ? 'ENFRENTE O MAGO' , 'POXA VIDA'}}"

<!-- property binding -->

<gege-student [tittle]= "!isMago ? 'ENFRENTE O MAGO' , 'POXA VIDA'">

====================================================================

Agora vamos ao código

No component do student, marcamos os atributos name e isMago como @Input() para que elas possam receber valor do component parent

É explicado que não precisa ser usado esse decorator, podendo passar o nome das propriedades que quero expor pelo decorator @Component, com o atributo inputs

Porém a forma mais comum e geralmente usada é a de marca com @Input() mesmo

Depois disso, é retirado os valores padrões, e será passado agora esse valores aonde o component é usado

Se o que eu tiver que passar para o component for um valor literal sem ser string, é necessário o uso de property binding

dá pra usar de uma forma em que o component student recebe um objeto de um tipo definido em uma model, essa model define uma interface.

E então no component em que vamos usar esse outro component, a gente passa um objeto quando for usar.

Enfim, essa parte ficou confusa, mas deu pra eu entender

====================================================================

O que serão Diretivas?

Vamos ver hoje

Components são diretivas com templates, então, components herdam tudo de diretiva.

Uma diretiva comum e simples, adiciona comportamento a um elemento do DOM mas não tem o template de um component

No angular, há 3 tipos de diretivas

Componentes ~> mais usadas e mais comuns

Estruturais ~> mudam o template ou estrutura do DOM

o angular trás algumas como ngFor e NgIf

Atributos ~> onde você associa um atributo a um elemento do DOM e um determinado comportamento é aplicado aquele elemento

Então, falando sobre diretivas estruturais:

<!-- ng if -->

o ng if permite renderizar um conteúdo quando uma expressão associada é verdadeira, se não for, o conteúdo não é exibido

As diretivas estruturais trabalham com um padrão de templates do HTML 5.

O \* (asterisco) é a forma abreviada da diretiva, pra facilitar, não precisando digitar o elemento template, a versão não abreviada é bem mais verbosa, e a forma abreviada é a mais usada.

versão abreviada

<input type="text" [value]="user.name">

<div \*ngIf="user.isJedi">

location of the jedi temple

</div>

modo não abreviado

<input type="text" [value]="user.name">

<template [ngIf]="user.isJedi">

<div>

location of the jedi temple

</div>

</template>

<!-- ng for -->

A diretiva ng for vai repetir o conteúdo de um elemento para cada item de uma coleção de objetos

Como ela é também estrutural ela vai repetir o template de cada elemento

Ela tem uma sintaxe onde se declara uma variável que pode ser usada para representar cada elemento da coleção

Segue exemplo

<ul>

<li \*ngFor="let user of users">{{ user.name }}</li>

</ul>

Nesse exemplo é a variável user representando cada item da lista de users.

Pode-se também declarar uma variável onde pode-se mostrar o índice de cada elemento da lista

exemplo

<ul>

<li \*ngFor="let user of users; let i = index">

{{ i + 1 }} - {{ user.name }}

</li>

</ul>

onde o primeiro começa com o valor 0

Repare bem que a variável é declarada como valor do elemento do DOM

<!-- ngSwitch -->

Existe também a diretiva ngSwitch, que avalia uma condição

e a diretiva ngSwitchCase renderiza o conteúdo equivalente

ngSwitchDefault mostra o conteúdo se nenhum dos cases for verdadeiro

Exemplo

<div [ngSwitch]="profile">

<p \*ngSwitchCase="root">LEIA E ESCREVA</p>

<p \*ngSwitchCase="user">SOMENTE LER</p>

<p \*ngSwitchDefault>Opa pera lá, mt calma ladrão</p>

</div>

====================================================================

Nessa aula, vai rolar um refactor no projeto, e usar as diretivas estruturais aprendidas na aula passada

Começando pelo component student, onde está sendo usada uma property binding [hidden], que controla o css da página e aquele conteúdo continua sendo renderizado, se não tiver cuidado, esse estilo pode ser sobrescrito, ou pior, pode haver falhas de segurança

Uma melhor maneira de esconder um certo conteúdo em uma página é trocando a propriedade [hidden] por um ngIf\* xD

o que vai depois das aspas é exatamente a mesma coisa. (que é uma expressão angular)

utilize o asterisco, o bagulho é doido

Resultado visual dessa alteração é a idêntico, porém, agora o código fonte da página não estará visível

Outra coisa que pode-se melhor, é o component principal onde foi colocado um usuário depois do outro, mas pode ser melhorado com o ngFor\* xD

Vamo meter alteração

No component principal, temos 3 propriedades com os mesmos elementos cada, então podemos colocar tudo em uma array e foda-se, xD

Mudando isso, deve-se alterar o template do component raiz

já que vai haver uma repetição ngFor nele

tem q ter atenção na declaração da variável no valor do \*ngFor

um exemplo

let student of students -> students pode ser um array ou um método por exemplo, depois da declaração, pode-se usar o property binding na mesma "tag"

A vantagem de usar diretiva é pq isso é mt melhor boyz pelo amor de DEUS

=======================================================================

usando operador de navegação segura

Se nesse momento for dado uma olhada no template do componente principal pode ser visto que estamos usando o component-student com ngFor, mas o que aconteceria se fosse passado um componente de student sem passar nada para esse atributo?

Se tu fizer isso amigo, irá pipocar erro no console, pq gg?

pq vai rolar uma tentativa de acessar uma propriedade que não foi definida no objeto, porém definidas no template

Uma coisa que pode ser feita para evitar esse tipo de situação é usando o operador de navegação segura, que é uma fodendo interrogação

um exemplo, tendo o seguinte template, colocarei o fodendo interroga em locais estratégicos

<div>

Esse é o jovem maluco doido {{student?.nome}}

<div \*ngIf="student?.isMago">

Se for mago entra aqui, um mago {{student?.lendario ? 'LENDÁRIO' : ' NORMAL'}}

</div>

</div>

Ta vendo onde eu coloquei o fodendo '?', com isso, estou avisando para o angular o seguinte: 'Se liga menor, se for undefined, nem tenta ler a o que vem a seguir nesta propriedade'

Isso é de muita utilidade quando essa web app for integrada com apis, pois a aplicação pode não conseguir buscar a fodendo propriedade por causa de n fatores.

Po, mas isso deixa exibindo algo ainda na minha página, tirou os erros? sim, mas, e se eu não quiser que isso seja exibido?

Nós podemos utilizar um ngIf para esconder tudo caso o studant seja inválido ou undefined

Como não temos um else nesse ngIf, podemos usar um outro ngIf bem bonito para a opção da propriedade ser inválida

Assim que é usado o operador de navegação segura, xD

======================================================================

Eventos de um componente

Processar eventos é um importante aspecto de uma aplicação web

Nessa aula será visto como o componente responde os componentes internos (ocorrem dentro do próprio template)

Em outra aula, será visto como o componente pode produzir eventos que podem ser consumidos por outros componentes interessados

Eventos tem uma sintaxe única para ser usados

Sua sintaxe é formada por parênteses ao redor de um evento e é linkado ao método de um component

Exemplo

import { Component } from ‘@angular/core’

@Component({

selector: ‘mt-clickable’,

template: ‘<button (click)=”clicked()”>’Click!</click>

})

export class ClickableComponent{

clicked(): void {

console.log(‘Button clicked!’);

}

}

Nesse exemplo, é definido um botão que quando clickado, dispara o evento clicked xD

Pode também se ter uma referencia ao evento gerado passando como argumento o ‘$event’

Segue ex:

@Component({

selector: ‘mt-clickable’,

template: ‘<button (click)=”clicked($event)”>’Click!</click>

})

export class ClickableComponent{

clicked(event): void {

console.log(‘Button clicked!’);

}

No método, podemos inspecionar para o evento para maiores detalhes

Em certos tipos de eventos, como o keydown, dá pra associar uma tecla

exemplo

@Component({

selector: ‘mt-clickable’,

template: ‘<input (keydown.space)=”keyDown($event)”>

})

export class KeyDownComponent{

keyDown(event): void {

console.log(`Key down: ${event}`);

}

Espaço foi apertado, evento ocorrido

======================================================================

Eventos de componentes

Um componente também pode emitir um evento personalizado

para isso é usado uma classe chamada eventEmitter, que é um adapter para a biblioteca rxJs e possui um método emitt que chamamos quando chegar o momento de emitir o evento

Voltando ao exemplo anterior:

import { Component, Output, EventEmitter } from ‘@angular/core’

@Component({

selector: ‘mt-clickable’,

template: ‘<button (click)=”clicked()”>’Click!</click>

})

export class ClickableComponent{

@Output() myEvent = new EventEmitter();

clicked(): void {

this.myEvent.emit();

}

}

Para usar eventos próprios, deve-se importar as classes Output e EventEmitter

E usar essa sintaxe maluca passada, em que não entendi muita coisa xD

Usando o component é possível atribuir o método que será chamado quando o evento for emitido

<mt-clickable (myEvent)=”willBeCalled()”></mt-clickable>

willBeCalled(): void {

console.log(‘Event from clickable’);

}

======================================================================

Adicionando eventos a um elemento do template

Como pode ser criado eventos dentro do component do projeto inicial?

Vamos criar um link no nome do student no template do component student e colocaremos o evento click entre parênteses seguido de um = “” onde entre as aspas ficará o método a ser chamado ao ser disparado aquele evento

No component, é implementado um método que será executado quando houver a ação que dispara a ação

Outra coisa falada nessa aula

Em angular você pode pegar referência a um elemento do dom com uma variável de template, como obtemos uma referência pra um elemento ou criamos essa variável

No exemplo, criamos um textarea e dentro da tag utilizamos a sintaxe #’algosemaspas’

O que isso possibilita? vou poder utilizar uma referência daquele elemento dentro de um template ou passar isso pra um método caso eu queira, com essa referência eu posso acessar atributos e também métodos

O podemos fazer por exemplo

Tendo o cenário que temos um template com o seguinte html

<textarea #descricao></textarea>

podemos fazer com que um botão passe o foco para esse elemento nomeado

<button (click)=”descricao.focus()”>YEA</button>

Então vemos aqui que é muito simples acessar um elemento do dom nomeado, tranquilin

======================================================================

Lidando com erros

A medida que desenvolvemos os componentes, podemos cometer vários erros, e esses erros serão reportados em dois lugares.

Quais esses lugares? Não sei, primeiro vamos criar um erro

Se produzirmos um erro no módulo principal do angular, tirando uma vírgula onde se declara os components, o compilador vai avisar que algo está errado

no meu caso, rolou isto

ERROR in src/app/app.module.ts(13,5): error TS1005: ',' expected.

Ta mostrando ali a linha e coluna do erro

Esse é um erro simples, mas, quando a gente tiver no desenvolvimento e fazendo grandes alterações, repare sempre o console (por isso que vsCode é bom)

Para que não faça uma grande burrada

Esse é o primeiro lugar a ser observado com problemas de compilação

O segundo lugar é o console do browser, que mostrará os erros de tempo de execução, mesmo sendo usado o typeScript que facilita a identificação de erros, ele só facilita, não extingue todos os erros da face do teu projeto

Por exemplo, se tirar o componente inicializador do projeto e deixar aquele array do módulo principal vazio, da uma olhada no console do browser

Fique de olho nos consoles