Curso Angular

NgModule: é um decorator, uma função que serve para aplicar metadados em uma classe, atributos, métodos ou argumentos de métodos. É aonde ficam as declarations, imports, providers e bootstrap.

O webpack é o responsável por fazer o bootstrap.

Componente: são pequenas partes independentes e reusáveis, em angular são classes que possuem um determinado ciclo de vida, possuem um template para definir a aparência e um selector(tag) para ser usado por outras partes da aplicação. Componentes são elementos personalizados.

Decorator @component: utilizado para definir o nome da tag que será utilizado no componente, nele se encontra o selector e a templatUrl ou template(recomendada apenas para aplicações pequenas e simples).

Criando um novo componente no angular:

ng generate component NomeCOmponent –spec=false/true:

Onde generate(g) componente(c) -spec(serve para testes, se desejar pode inicar como false)

Property Binding: serve para linkar o valor de uma propriedade de um elemento há uma expressão angular. É é necessário usar o property binding para coisas que são mais complexas, por exemplo em um input boolean, se não colocar o property binding o valor assumira a string “true/false” ao invés do valor booleano, Ou seja: Sempre que precisar passar um valor que não seja string, deve ser usado property binding.

One way binding: quando a atualização do property binding ocorre apenas em um sentido, sempre do componente para o template. Exemplo:

Component: user = {name=”Israel Gomes”}

Template: <input type=”text” [value]=”user.name”/> // quando o valor do name user for mudado no componente, automaticamente o value do input será mudada

Template interpolation: {{}}

Quando utilizamos o hidden a visibilidade do elemento é controlado via CSS, ou seja, caso tenha feito de forma certa e mesmo assim o elemento continua visível pode ser um erro no css. Por isso é aconselhável utilizar ngIf.

Diretivas: Componentes são diretivas com templates, uma diretiva comum e simples serve para adicionar comportamento ao um elemento do dom.

Tipos de diretivas : Componentes, estruturais(muda o template estrutura do dom, como ngFor ngIf, etc) , atributos(você associa um atributo ao elemento do dom)

Operador de navegação segura:

Quando você invoca um componente que espera diretivas para ter valores, se você invoca-lo sem nenhuma diretiva ele irá dar erro, indicando que está tentando ler propriedades definidas no template que esperam objetos para representa-las. Uma das opções para evitar esse tipo de erro é o operador de navegação segura(?). Exemplo

No componente: <h3>Student name : {{student?.name}}</h3>

Dessa forma ele diz que caso student seja indefinido(undefined) não tente ler o nome desse objeto.

\*Eventos de um componente

Rotas: As rotas servem para você utilizar uma mesma base e colocar nela um conteúdo dinâmico, mudando apena o conteúdo do body da página, para isso se é utilizado o <router-outilet></router-outlet>. Para isso é necessário mapear cada componente na rota. Rotas são basicamente um array que contém o mapeamento de cada caminho para um componente.

Você pode criar um arquivo chamado app.routes.ts, adicionar suas rotas nele, após isso deve ser adicionado no modulo principal da aplicação(app.module.ts) com o forRoot(ROUTES) no RouterModule. Se utiliza a diretiva [routerlink] para utilizar o esquema de rotas.

\*Os links ficam juntos no começo, para arrumar isso foi utilizado a diretiva routerLinkActive=”active”

Injeção de dependência: é um padrão de projeto que a aplicação deixa de instanciar seus projetos manualmente e passa a depender do framework.

Serviços: são classes comuns em angular que você pode usar para injetar em componentes e em outros serviços, serviços são geralmente usados em uma aplicação para encapsular o acesso a api de backend

@Injectable(): quando serviços solicitam a injeção de outros serviços,serve apenas para receber injeçõs do framework

Alternativa de mudar o titulo da página:

@Component({  
viewProviders: [TItle]

}]

Constructor (title: Title){

title.setTitle (‘: : My title :: ’)

}

Decorator @Input(): utilizado para que outros components parents possam passer valores para o component pai. Ex:

  @Input() restaurant: Restaurant; //Restaurant é o model.

No component parents pode se criar um objeto comum de restaurants:  
restaurants: Restaurant[] = [

  {

    id: "bread-bakery",

    name: "Bread & Bakery",

    category: "Bakery",

    deliveryEstimate: "25m",

    rating: 4.9,

    imagePath: "assets/img/restaurants/breadbakery.png",

  },

  {

    id: "burger-house",

    name: "Burger House",

    category: "Hamburgers",

    deliveryEstimate: "100m",

    rating: 3.5,

    imagePath: "assets/img/restaurants/burgerhouse.png",

  }]

Agora vem a utilização do @Input, no template do restaurant vamos enviar os valores que para o input do restaurants do outro componente:

              <mt-restaurant [restaurant]="restaurant"></mt-restaurant>

Onde [restaurant] é o que está marcado com o decorator @Input()