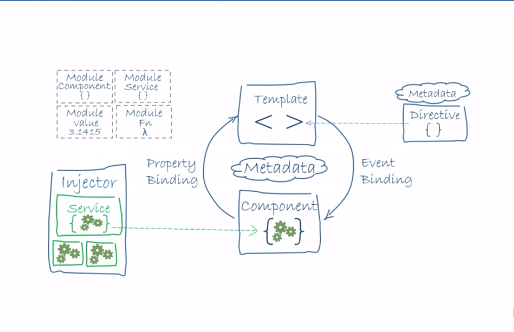
Angular

O Framework possui 8 blocos principais:

1. Componentes
2. Diretivas
3. Roteamento
4. Serviços
5. Template
6. Metadata
7. Data Binding
8. Injeção / Dependência



* **Componentes**: (view), ela encapsula:
  + Template
  + Metadata (processamento das classes)
  + Dado a ser mostrado na tela (Data Binding) (uma div por exemplo)

Quando criamos aplicação em Angular, criamos aplicações que são orientadas à componentes. Temos então: pai/mãe (que vai ser o componente raiz) e partir dele, deriva-se novos componentes.

Sendo que cada componente também pode ter um componente filho (como uma árvore binária)

O componente pode interagir com o back-end em inúmeras linguagens, como: NodeJS, Java, .NET, Ruby, Python... Quem faz o intermédio, entre o componente e o back-end é o **“Serviço”** ele pode ser injetado em outras classes

Por mais que o Angular trabalhe como SPA (Single Page Application), isso não impede que tenha várias abas dentro da mesma aplicação, o responsável pela navegação entre as páginas é o **Router.**

Além disso, temos uma Diretiva, a qual é responsável por modificar elementos DOM (Document Object Model) e/ou seu comportamento, exibir uma div, modificar valores, etc.

Utilizando Angular CLI para criar novos componentes:

$ ng generate componente <name>

Ou:

$ ng g c <name>

Para inicializar o projeto:

$ npm start

**TYPESCRIPT**

Para trabalhar com ecma script, precisaremos de um transpiller... um dos mais conhecidos é o babel.js.  
mas também é possível usar o ‘tsc’ no console para transpillar para js

$ tsc <nome-arquivo.ts>

Hoje em dia, temos os decorators para fazer essa parte. Depois procurar saber melhor

As variáveis podem ser tipadas, ou seja, definir o tipo primitivo dela, assim como C e Java. para isso: funciona dessa forma:

Var n1: string = “teste string recebida”;

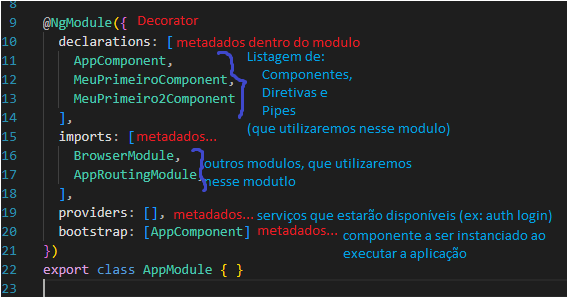
 n1 = 4 (causará erro, pois só aceita tipo string)

var n2: any = “outro teste”;

n2 = 5 (não causará erro, pois a n2 aceita qualquer tipo)

**Módulo**

Por que separar em módulos? Organizar melhor o código, escolher o que cada modulo expõe, etc.



Para gerar um modulo pelo CLI:

$ng generate modules <nome>

Ou

$ng g m <nome>

a

**Templates**

Com os templates é possível utilizar os dados dinamicamente. No component.ts, precisa declarar os atributos (de forma tipada) que utilizará no html. Por Exemplo:

Para fazer a interpolação nessa linha:

<p>Lista de cursos do portal: {{ nomePortal }}</p>

O atributo nomePortal já precisa estar declarado na classe do componente, e sua declaração é feita dessa forma:

export class CursosComponent {

  nomePortal: string;

  constructor() {

    this.nomePortal = 'https://loiane.training';

  }

}

**Serviços (injeção de dependências)**

$ng g s <nome do serviço>

A injeção de dependência serve para que o angular automaticamente forneça uma instância de uma classe de serviço, ao invés de usar um HttpRequest, isso é feito através de um decorator: @Injectable(); esse decorator fica acima da declaração da classe. Para usá-lo, seguem os passos:

* Precisa receber a classe como argumento do método construtor:

  constructor(private cursosService: CursosService) {

* (após isso, consigo utilizar um método dessa classe dentro do bloco do constructor):

    this.cursos = this.cursosService.getCursos();

* E também, preciso passar como um provider no “index” do modulo:

  providers: [

    CursosService

  ]

* Não esquecer de importar:

import { CursosService } from './cursos.service';

Plugins / Extensões:

Angular Snippets

Auto Import

**Novo projeto: Data Binding**

Data Binding -> associar informações que estão no componente para o template, e vice-versa

No angular existem 4 formas

**Interpolação:** “ {{ valor }} “

**Property Binding:** “ [propriedade]= “valor”; ”

**Event Listener:** “ (evento) = “handler”; “ -> (oposto do property binding)

**2 way database:** “[(ngModel)] = “propriedade” “ -> mantem os 2 componentes atualizados ao mesmo tempo.

**Class e Style Binding**

Para pegar o valor de um combo (um select) precisa criar uma variável local dentro do template, e para isso, precisa “escutar” quando o valor muda. Nesse caso:

<select name="" #classe (change)="0">

Class Binding pode ser usado com property Binding, mas também é possível usar a interpolação.

1. Class Binding com property binding:

<div class="alert" role="alert" [class.alert-success]="classe.value == 'alert-success'">Sucesso</div>

1. Class binding com interpolação:

<div class="alert {{ classe.value }}" role="alert" >Texto colorido conforme valor do combobox</div>

Event Binding

No property Binding, usamos colchetes para indicar qual a propriedade a ser alterada. No caso do evento, usamos parênteses. Ex:

<button (click)="botaoClicado()" class="btn btn-primary">Me Clique!</button><br><br>