**Detalhes:**

**1 –** Função JSON.stringify(), transforma um objeto em string;

**2 –**  Ao usar um serviço typeScript podemos usar o “método construtor” para inicializar algumas funções, porém também podemos usar o “método OnInit” que e um inicializador que ocorre após o construtor ter sido processado. Exemplo:

Export class AppComponent implements OnInit {

Constructor(){

This.funcionarioService = new FuncionarioService();

}

ngOnInit(){

}

}

**3 – Injectable():**

E usando quando desejamos injetar um serviço dentro do outro ou seja quando desejamos usar funcionalidades de outras classes de serviços, antes da “export class NomeClasse{

}

Devemos adicionar o @Injectable():

@injectable()

export class NomeClasse{

}

**Módulos em Angular:**

Os módulos em Angular, servem para que nós possamos separar nossos componentes, diretivas, pipes, etc.

Para criar um módulo devemos usar “ng g m nome”;

Informações @NgModule:

Declarations: Declaramos o que pertece ao módulo, por exemplo, componentes, pipes, diretivas, etc.

**Usar módulos dentro de outros módulos:**

**1 –** Primeiro momento é necessário adicionar o módulo que deseja usar no

@NgModule - > “imports:[]”, do módulo onde deseja usar. Por exemplo:

Para usar o módulo funcionários dentro do Módulo pessoas, e preciso adicionar o módulo dos funcionários no “@NgModule” no Módulo de pessoas. Obs: no momento que você adicionar o módulo dentro do outro módulo, confira se nos “Import { FuncionariosModule } from ‘./Funcionarios/funcionários.module’ está importado.

**2 –** Segundo momento é necessário exportar os componentes que deseja usar, usando o exemplo a cima, depois de importado o módulo de funcionários no módulo de pessoas, devemos ir no arquivo “ts” do módulo de pessoas e exportar os componentes que desejamos exportar. Exemplo do arquivo “ts” do módulo de funcionários :

@NgModule({

Imports: [

CommonModule

],

declarations: [ CadastrarFuncionarioComponent, ListarFuncionariosComponent ],

exports: [ CadastrarFuncionarioComponet ]

})

Nesse nosso exemplo, estamos exportando apenas o componente para os cadastros de funcionários.

**Serviços em Angular:**

Classes de serviços, são para centralizar regras de negócios e executar chamadas remotas.

**Usando um Serviço Angular:**

**1 –** Para usar um Serviço, depois de criado, você vai até o arquivo “ts” do componente que deseja usar os serviços, no arquivo deve-se estanciar o serviço, por exemplo:

export class FuncionarioService {

funcionários = [{ id: 1, nome: ‘João’ }];

adicionar(nome: String) {

const funcionario = {

id: 1,

nome: nome

};

}

}

Digamos que tenho a classe de Serviço chamada “funcionário.service.ts” onde dentro dessa classe temos uma “classe” chamada funcionarioService, ao acessar o arquivo “ts” do componente devemos fazer assim:

Export class FuncionarioFormComponent {

* Instanciação: funcionarioService: FuncionarioService;

Constructor() {

This.funcionarioService = new FuncionarioService();

}

Adicionar(nome: String){

This.funcionarioService.adicionar(nome);

}

}

**Injeção de dependências dos Serviços:**

Quando se usa a injeção de dependências você acaba economizando tempo em instancias das “classes”, como exemplo, vou usar a classe do arquivo “ts” do AppComponent:

Export class AppComponent implements OnInit {

* Instanciação: funcionarioService: FuncionarioService;

Constructor() {

This.funcionarioService = new FuncionarioService();

}

Adicionar(nome: String){

This.funcionarioService.adicionar(nome);

}

}

Dessa forma que iriamos fazer as instanciações, mas se usarmos a injeção de dependências, podemos simplesmente ir no arquivo de modulo onde está nosso componente, no nosso caso é no “app.module.ts”, e adicionarmos a nossa classe no “providers:[]”. Por exemplo:

@NgModule({

Declarations: [

AppComponent,

FuncionarioCardComponent,

FuncionarioFormComponent

],

Imports: [

BrowserModule

],

**Providers: [ FuncionarioService ],**

Bootstrap: [ AppComponent ]

})

**Ai na “classe” do componente ficaria assim:**

Export class AppComponent implements OnInit {

Constructor( private funcionarioService: FuncionarioService ) {}

Adicionar(nome: String){

This.funcionarioService.adicionar(nome);

}

}

**Usando Decorador @Inject:**

O Decorador @inject serve como tolken de acesso, por exemplo:

Digamos que criamos a classe de serviço “log.service.ts”:

Export class LogService {

Constructor(

@inject(‘LogPrefixo’) private prefixo: String

) { }

log( msg: string ) {

console.log( `${ this.prefixo }: ${msg}` );

}

}

Ai no meu arquivo do modulo “ts” utilizamos assim:

@NgModule({

Declarations: [

AppComponent,

FuncionarioCardComponent,

FuncionarioFormComponent

],

Imports: [

BrowserModule

],

**Providers: [**

**FuncionarioService ,**

**LogService,**

**{ provide: ‘LogPrefixo’ , useValue: ‘ LOG ’ }**

**],**

Bootstrap: [ AppComponent ]

})

**Data Binding:**

**1 –** Em angular podemos usar 4 tipos de Binding:

1. Interpolação: ‘{{ valor }}
   * Associa informações para o template (HTML);
2. Property Binding: [propriedade]=”valor”
   * Associa informações para o template;
3. Event Binding: (evento)=”handler”
   * Associa informações do template (HTML) para o componente
4. Two-Way Data Binding: [(ngModel)]=”propriedade”
   * Associa informações entre ambos, ou seja, mantém ambos atualizados (componente e template (HTML))