**Cornell Notes Template**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | **ng:** ferramenta do angular que auxilia o dev, ele |
|  | ele cria coisas que facilita para o dev como compo- |
| **Material Design angular io:** | nente, modulo, servicos, diretivas. |
| Framework, do angular |  |
| Mas tbm podemos usar | **File** **main.ts:** primeiro arquivo que e executado, vai |
| Qq outro como por ex | Fazer toda a parte do bootstrap, que é a parte do |
| Materialize, bootstrap. | Boot da aplicacao |
|  |  |
|  | **AppModule:** é o principal modulo do projeto, e nele |
|  | Tem os principais components que vao ser executa |
|  | Dos na aplicacao |
|  |  |
| X | **bootstrap:[] file app.module** : vai indicar ao angular |
|  | Qual component sera carregado no momento que o |
|  | App for chamado |
|  |  |
|  | **Seletores:** |
|  | * **Tag**: **‘nome’** <nome></nome> |
|  | * **Diretiva: ‘[nome]’** <div nome></div> |
|  | * **Classe** **‘.nome’** <div class=’nom’> </div> |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Data binding**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | * **Como um component pode comunicar** |
|  | **Entre a parte do template e a parte logica comporta** |
|  |  |
|  | * **String interpolation:** como a partir do template |
|  | Eu posso acessar um dado, variavel, atributo que |
|  | Foi declarado na class |
|  |  |
|  | * **Property binding:** é associar a uma determina |
|  | Propriedade, atributo de um component o valor |
|  | De alguma variavel que esteja dentro da classe |
|  | [propriedade]=”nome\_variavel” |
|  |  |
| X | **ngOnInit : funcao** que é executado automaticament |
|  | A partir do momento que a parte visual templ esteja |
|  | carregada |
|  |  |
|  | **Event Binding:** associar eventos a components |
|  | que estao no template |
|  |  |
|  | **Two way data binding:** usar uma variavel para lincar |
|  | a um atributo, concomitante a isso a uma variacao |
|  | do valor do component seja refletido a variaval |
|  |  |
|  | **Two way data binding:** para trabalhar com isso usam- |
|  | os o conceito do atributo **[(ngModel)]=””** – tem essa |
|  | forma declarativa. A finalidade é o valor que definer |
|  | na variavel vai para value do component e qualquer |
|  | alteracao do valor por meio do evento vai influir |
|  | Na variavel. Lembrando que o ng model está ligado |
|  | A utilizacao de forms |

**DIRETIVAS**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | **Single page application:** trabalham diretamente com |
|  | Manipulacoa de objeto dom, e podemos fazer isso com |
|  | O uso das diretivas. |
|  |  |
|  | **Diretivas:** é a ideia de a gente manipular os objetos |
|  | Dom da page do cliente. |
|  |  |
|  | **Diretivas de atributos:** sao diretivas que promove |
|  | Mudanças em objetos já existentes. |
|  |  |
|  | **Diretivas:** **estruturais:** acrescentam ou removem |
|  | objeto Da pagina. |
| X |  |
|  | **[ngStyle]=”{fontSize: var, color: var}”** define em |
|  | tempo de execucao o stilo do component. |
|  |  |
|  | [**ngClass]=”var”** segue o raciocinio do anterior. Pode ser |
|  | alterado em tempo real. |
|  |  |
|  | **\*ngIf=”var”** usa expressao logica e com isso podemos |
|  | Habilitar ou nao determinado componente |
|  |  |
|  |  |
| **\*ngFor=”let name of names”** | possibilita criar vario s components por meio de um loop |
| **\*ngFor=”let name of names;** | **let** **i=index”** para usar o for vc pode o mat-list p ex |
| **[ngSwitch]=”var”** | **\*ngSwitchCase=”’valor’”** |
| **<ng-template>** renderiza html | Normalmente usado em diretiva, component mais com |
| complexo |  |
| **Variavel de referencia: #var** | Vc pode nomear as tags com uma variavel, dar nome |
| Aos componentes |  |

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
| **<ng-container>** é um pouco | Semelhante a div. mas ele vai conter elementos |
| Que atendam a determinadas | Requisites. Possibilitar trablhar com as diretivas ngIf |
| Com ngFor tudo junto | por exemplo |
|  |  |
| **<ng-content>** ele vai usar | O conteudo posto entre as tags do componentes |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Comunicação entre componentes**

**NOTES WRITTEN**

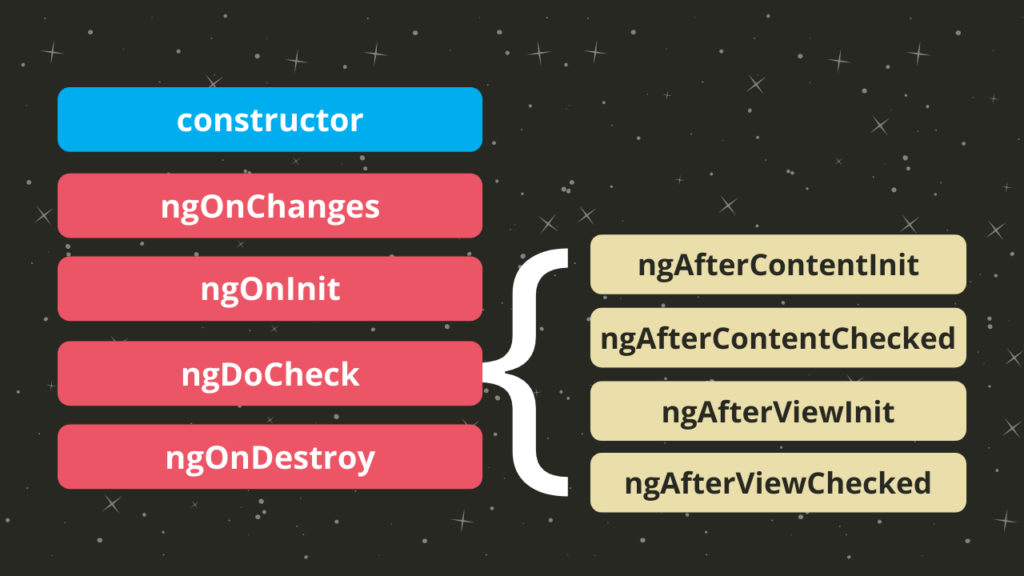
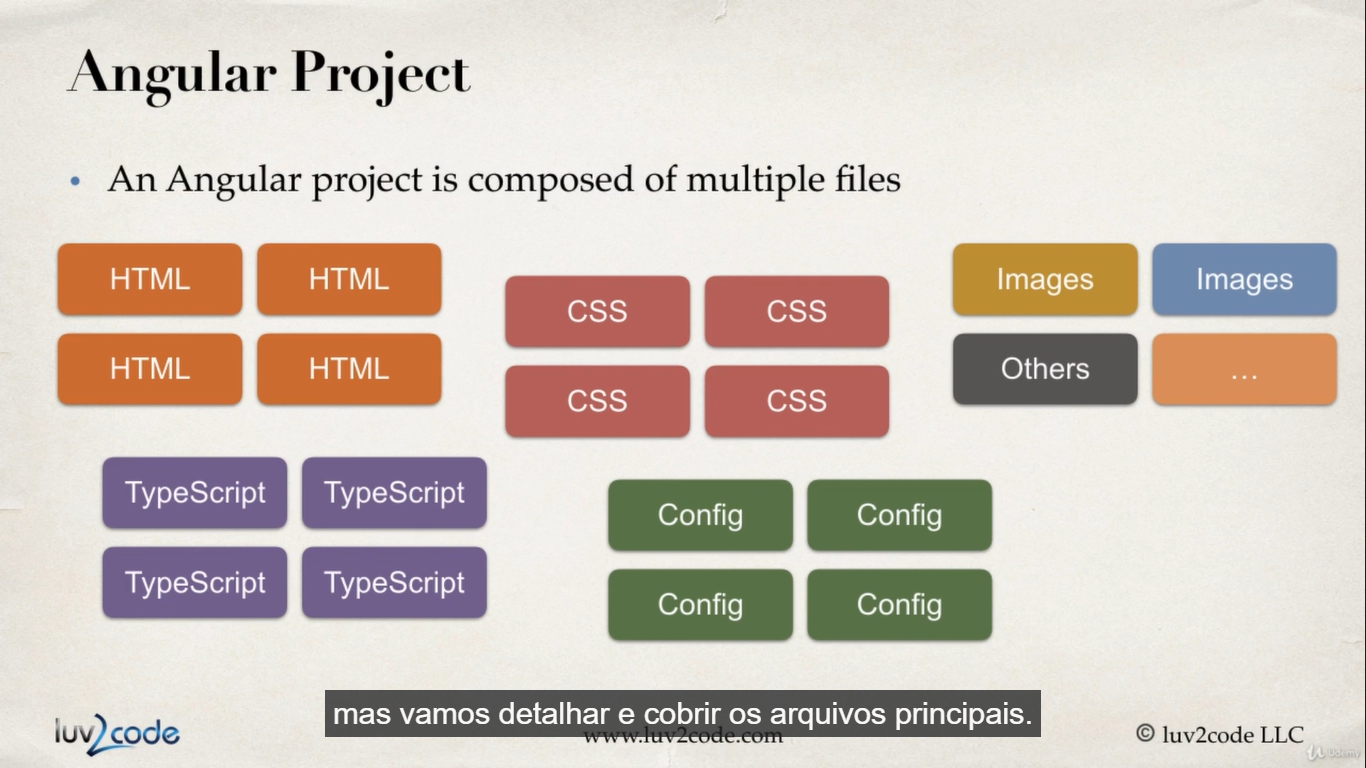
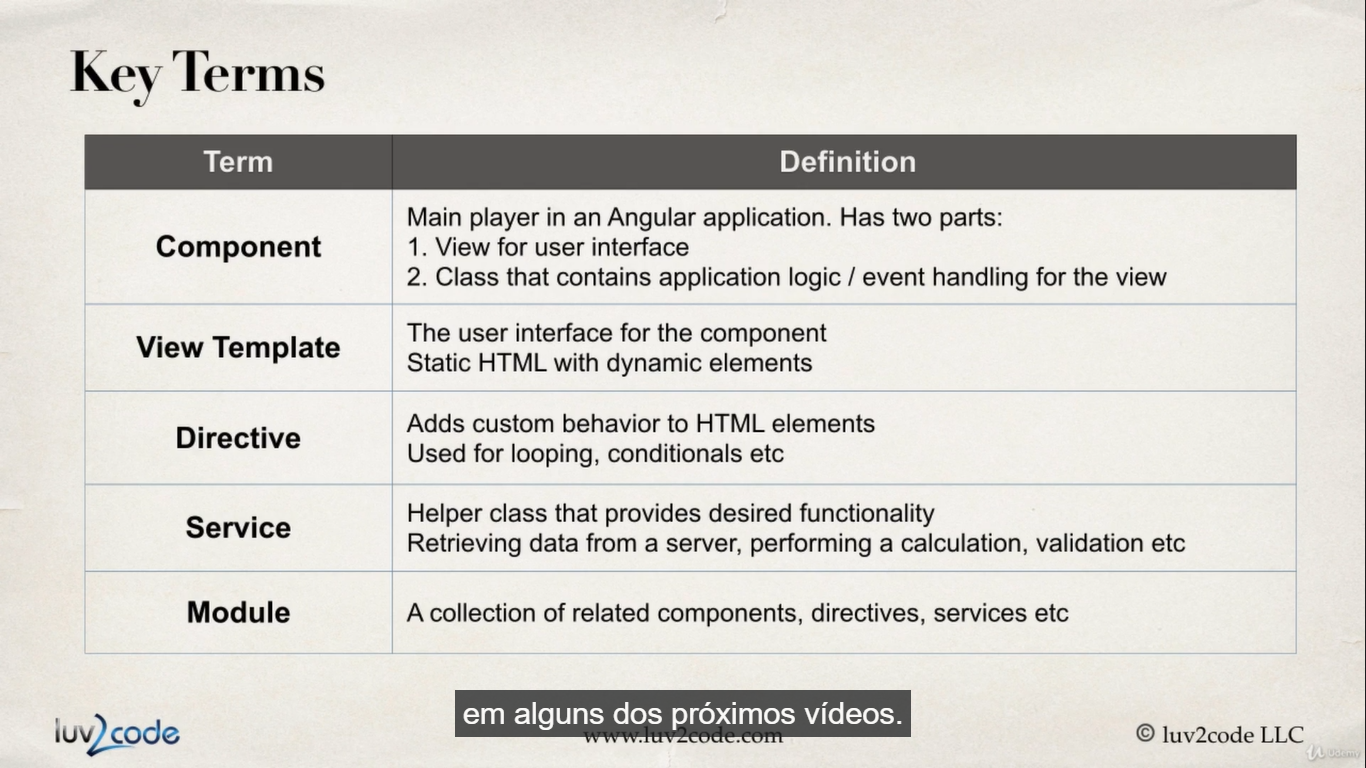
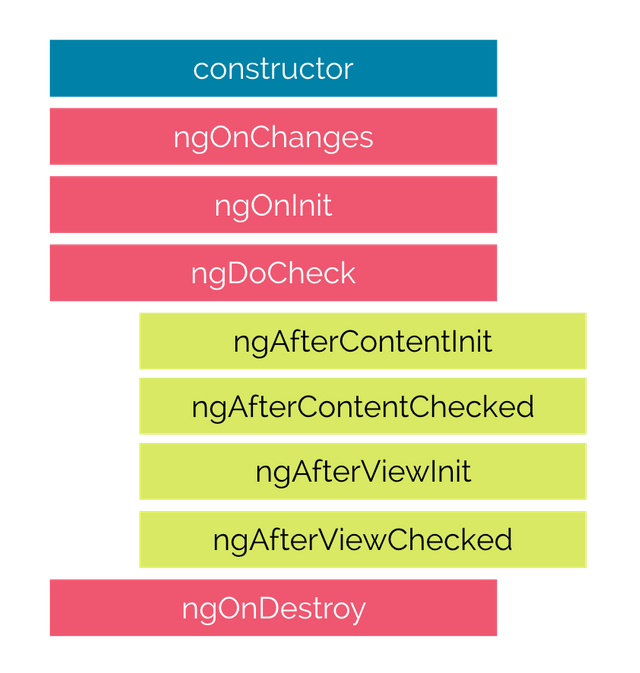
**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
| **Como os components se** | **Input Biding: @Input () @Input(‘name’)** |
| **Comunicam? Como se inter** | <component var=”qqn”> |
| **Agem entre si?** |  |
|  |  |
|  | **EventEmitter**: gerador de eventos **@Output()** |
|  |  |
|  | **Variável de template: #var** vai nomear o component |
|  |  |
|  | **@ViewChild(nome\_compoente)::** essa notacao quer |
|  | Dizer que o angular vai pegar o 1º component que |
|  | Estiver dentro do templante com essa declaracao |
|  | “nome\_compoentne” e vai injetar dentro da class. |
| X | Ele pega o primeiro component declarado dentro |
|  | Do template, se vc estiver usando mais de uma |
|  | Poderá usar a variavel de template para diferenciar |
|  | **@ViewChild(“var\_template”).** Por meio dele pegamo |
|  | A referencia deste objeto |
|  | **OnChanges :**: interface que captura mudanças. Essa |
|  | Interface implementa o metodo |
|  | **ngOnChanges(changes: {[propkey: string]:** |
|  | **SimpleChange})** |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| Pode usar o set e get para inter | ceptar as mudanças dos inputs |
|  |  |
|  |  |
|  |  |
|  |  |

**Ciclo de vida do componente**

**NOTES WRITTEN**

|  |  |
| --- | --- |
|  | **Hooks**: é um metodos executados em determinados estagio do ciclo de vida do comp |
|  | As interfaces que possuem esses metodos a serem implementados sao opcionais |
|  | **HOOKS FOR COMPONENTS OR DIRECTIVES** |
|  | **constructor() {}** primeiro metodo que é chamado. This is invoked when |
|  | Angular creates a component or directive by calling new on the class. |
|  |  |
|  | **ngOnChanges(changes: SimpleChanges) { }** |
|  |  |
|  | **ngOnInit () { }** chamado após o change. Invocado quando determinado |
|  | Component foi inicializado. Este hook é chamado apenas uma vez apos o primeiro |
| Ch | change |
|  |  |
| X |  |
|  |  |
|  |  |
|  | **HOOKS FOR THE COMPONENT’S CHILDREN** |
|  | **ngAfterContentInit:** chamado depois que o conteúdo externo é inserido no component |
|  | por exemplo vindo por meio do ng-content |
|  |  |
|  | **ngAfterViewInit::** chamado logo após o conteúdo do próprio component e de seus |
|  | Filhos ser inicializados |
|  |  |
|  | **ngOnDestroy:** chamdo antes do angular destruir o componente |
| **SUMMARY** | **ngDoCheck:** chamado a cada ciclo de deteccao de alteracao(processo que pecorre |
|  | o component atrás de mudança). É invocado quando o detector de alteracoes do |
| X | component é chamado. Use este ao invés do change, e outra não deve usa-lo com o |
|  | Docheck juntamente. |
|  | **ngAfterContentChecked:** invocado ap´so a verificacao do conteudo externo |
|  | **ngAfterViewChecked::** chamado cada vez que o conteudo do component é verificado |
|  | pelo mecanismo de alteraç~eos do angular. |

**Dependency Injection DI**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | Capacidade de resolver dependecias de instancias de objetos. |
|  | Cria-se uma nova instancia ou fornece-se uma ja criada |
|  |  |
|  | **constructor(private dependencia: Dependencia){}** |
|  |  |
|  |  |
|  |  |
|  | **@Injectable({providedIn: ‘local’})** |
|  | **export class nome {}** |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Serviços**

**NOTES WRITTEN**

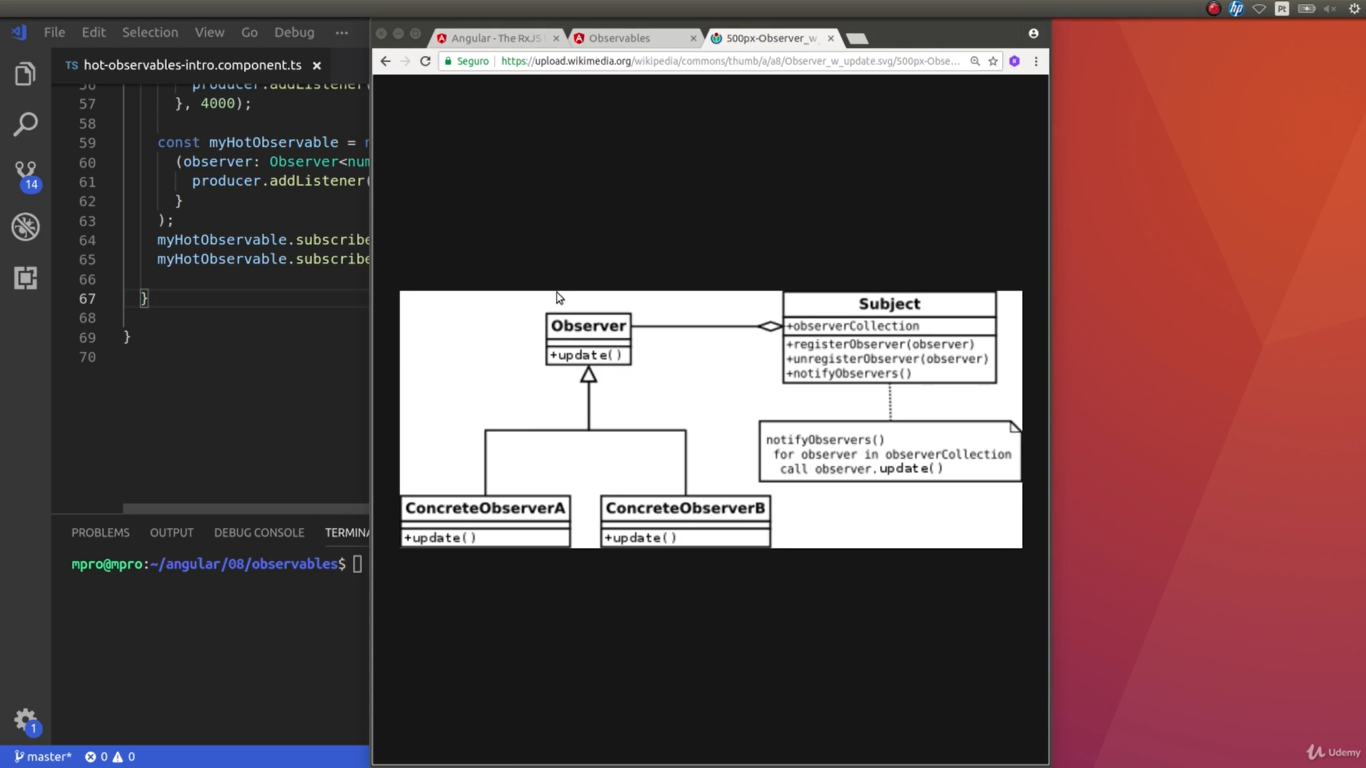
|  |  |
| --- | --- |
|  | Classes que fornecem serviços que podem ser compartilhados entre vários modulos |
|  | Ou components e que possuem a caracteristica de serem injetaveis. |
|  |  |
|  |  |
|  |  |
|  | **constructor(private service: MyService) {}** |
|  |  |
|  | Como sera uma class que sera injetada dentro de outra. Utiliza um decorator |
|  | **@Injectable()** - com isto o angular vai entender que a class podera ser injetada dentro |
|  | Dos modulos, pode ser criado automaticamente pela parte de gerenciamento de depen |
|  | Dencia do angular |
|  |  |
| X | **Provider**: é um cara que vai fornecer o service. providers:[] |
|  |  |
|  | **providedIn:** vc pode indicar onde o service sera fornecido a partir do proprio service, |
|  | Assim fica duas possibilidade, ou vc vai no modulo e indica o provider ou a partir do |
|  | Proprio service vc indica o provider, de onde sera fornecido |
|  |  |
|  | **@Injectable({providedIn: ‘local’})** |
|  | **export class nome {}** |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X | **Subject:** é um subtipo de estrutura do observable |
|  |  |
|  |  |
|  |  |
|  |  |

**RXJS – Observables e subjects**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

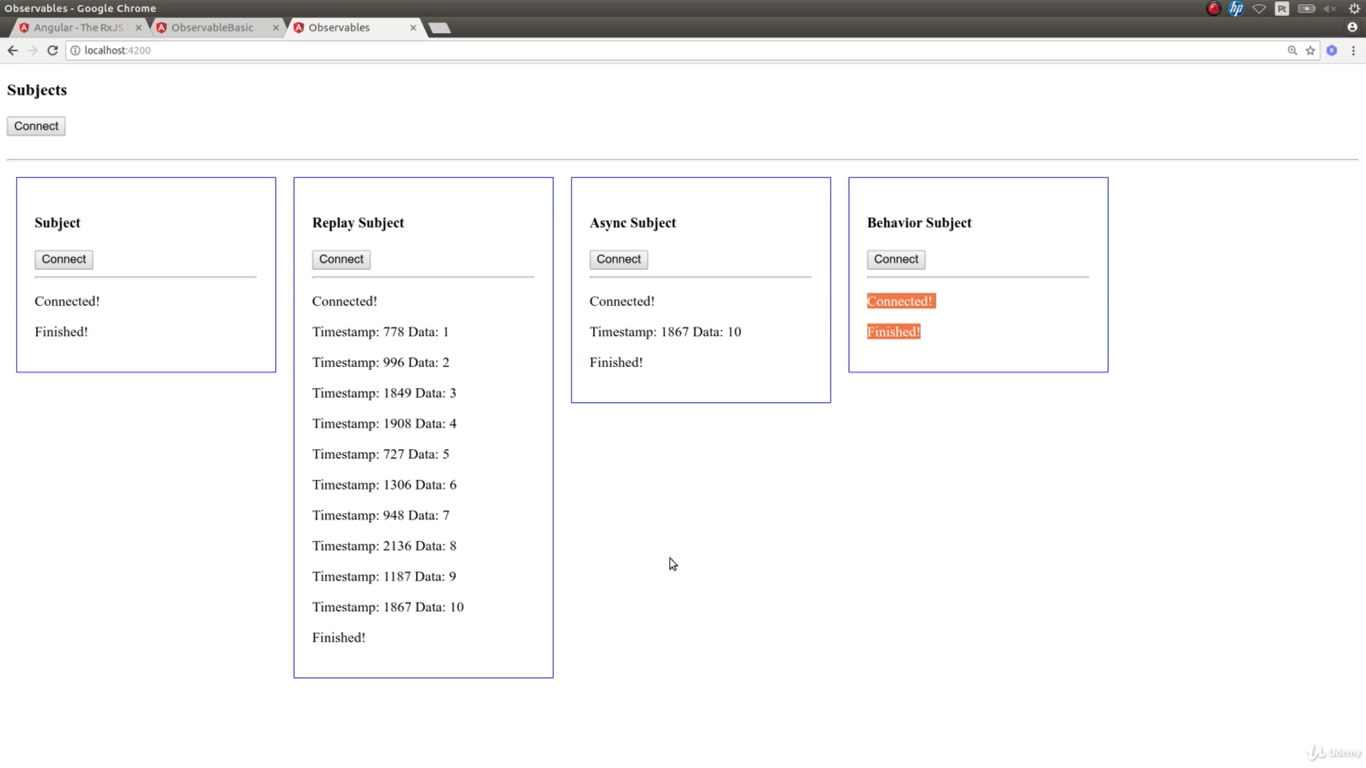
|  |  |
| --- | --- |
|  | **Pacote rxjs:** contem todas as class operadores funcoes para trabalharmos com a parte |
|  | Assincrona da programacao reativa. Ja vem por padrao |
|  |  |
|  | **Observable:** vai observer a geracao de determinado dado Observer. |
|  | Observer é generic <> precisa especificar o tipo que sera observado. |
|  |  |
|  | **next():** metodo do Observer que é utilizado para gerar os dados . |
|  | **complete()** – metodo do Observer que diz que o Observable está completo |
|  | **subscribe()** – metodo que chama o observable criado |
|  | **erro()-** metodo do Observer que indica um erro |
|  |  |
|  |  |
| X | **Cold Observables:** usando o mesmo observable conseguimos obter fonte gerador de |
|  | Dados distintos. Ou seja, a cada subscribe é criado uma nova fonte produtora. |
|  |  |
|  | **Hot Observables:** quando o produtor de dados dentro do observable não é criado a |
|  | Cada subscribe. |
|  |  |
|  | **Subject:** permite fazer um subscribe e um publish, ou seja, ele nos permite enviar e |
|  | Receber dados. |
|  |  |
|  | **pipe() –** funcao do observable que conseguimos aplicar alguns tipos de operadores a |
|  | Esses dados. Podemos filtrar os dados que recebemos algo do tipo. Gera um subject |
| **SUMMARY** | **refCount():** faz com que o subject seja conectado ao observable |
|  | **publish():** retorna um subject |
| X | **Share:** serve para a mesma coisa do publish e refCount, mas mesmo se o produtor |
|  | Observable deu complete com o share se alguem subscrever novamente |
|  | Então o produtor começa novamente |
|  |  |
|  |  |



**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | **Subject :** ele tem um observador que conseguimos capturer os dados de uma |
|  | Observable se subscrevermos o subject a um Observable bem como diretamente nele |
|  | Podemos gerar dados. LOGO ele tem dois lados |
|  |  |
|  | **ReplaySubject:** pega o primeiro dado gerado |
|  |  |
|  | **BehaviorSubject:** quase semelhante ao Subject, porém ele começa no dado gerado |
|  | Anterior a subscricao enquanto q o Subject pega o dado gerado após a subscricao |
|  |  |
|  | **Async Subject:** pega apenas o ultimo evento ou dado. |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |



**Operadores RXJS**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | from([]) – criador de observable |
|  | of()– criador de observable |
|  | interval(milisegundos) – criador de observable |
|  | Time(milisegundos) - como o timeout, depois disso ele aciona – criador de observable |
|  | Time(3000, 1000) – depois de 3 s conte de 1 s – criador de observable |
|  | fromEvent(element do dom, ‘evento’) – criador de observable |
|  |  |
|  |  |
|  | * Entenda os operadores como canos conectados ao cano principal de agua |
|  | E cada cano fará alguma tarefa como retirada de agua, colocar mais cloro e assim vai |
|  |  |
|  | Map() – precisa adicionar o pipe |
| X |  |
|  | Delay(milissegundos) |
|  |  |
|  | Filter() |
|  |  |
|  | Tap() – vc nao toca o valor, a ideia é apenas logar o valor |
|  |  |
|  | Take() |
|  | First() |
|  | Last() |
|  |  |
| **SUMMARY** | debounceTime(milisegudos) |
|  |  |
| X | takeWhile() |
|  | takeUntil() |
|  | Retry() ou tbm retryWhen |
|  | toArray() |
|  | observable | async tbm (observable | async)?.propriedade |

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | mergeAll() – colocando isso no pipe nao precisaremos subscrever os observable. Ele |
|  | Cuida dessa parte automaticamente. |
|  |  |
|  | mergeMap() – faz filtros e subscrabe internamente |
|  |  |
|  | switchAll() – faz a mesma coisa que o mergeAll, porém ele descarta as requisições |
|  | Anteriores e se atenta apenas as novas. A medida que surge novas req, ele troca |
|  | Pelas velhas. |
|  |  |
|  | switchMap() |
|  |  |
|  | * Usar o debounceTime e o switchMap conjuntamente é o pica |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**REQUISIÇÕES HTTP**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | | async -> ele pode subscrever as observable automaticament e gerar os dados |
|  | Bem como requisitar, ou seja tbm é uma forma de vc fazer um subscribe no template. |
|  |  |
|  | snackBar: MatSnackBar |
|  |  |
|  | HttpError: dois tipos de erros: o erro que nao consegue converser com o server e o |
|  | Erro que consegue converser com o server |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**FORMS**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | <mat-hint> verificacao de input (classes: ng-untouched: nao foi clicado; pristine: nao |
|  | alterou nada no campo ainda; invalid: validacao do campo; touched: foi tocado; |
|  | dirty: alterou o valor |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**React Forms**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  | ReactiveFormsModule |
|  |  |
|  | FORMCONTROL -> trabalhamos com ele quando nao temos por exemplo formulario |
|  | E queremos controlar o valor de algum input, entao podemos usar essa técnica ou tbm |
|  | Podemos usar o ngModel como fizemos |
|  |  |
|  | FORMGROUP -> agrupa forms. Usamos para atrelarmos um grupo de campos |
|  |  |
|  | FORMBUILDER -> um jeito melhor de tartar o form retornando um objeto literal num |
|  | Instante. Possibilita vc trabalhar tbm com array, formGroup, formControl. E OBS |
|  | É uma injeção de depencia de service |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Cornell Notes Template**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Cornell Notes Template**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Cornell Notes Template**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Cornell Notes Template**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Cornell Notes Template**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Cornell Notes Template**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Cornell Notes Template**

**NOTES WRITTEN**

**CUE WORDS or QUESTIONS**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **SUMMARY** |  |
|  |  |
| X |  |
|  |  |
|  |  |
|  |  |
|  |  |