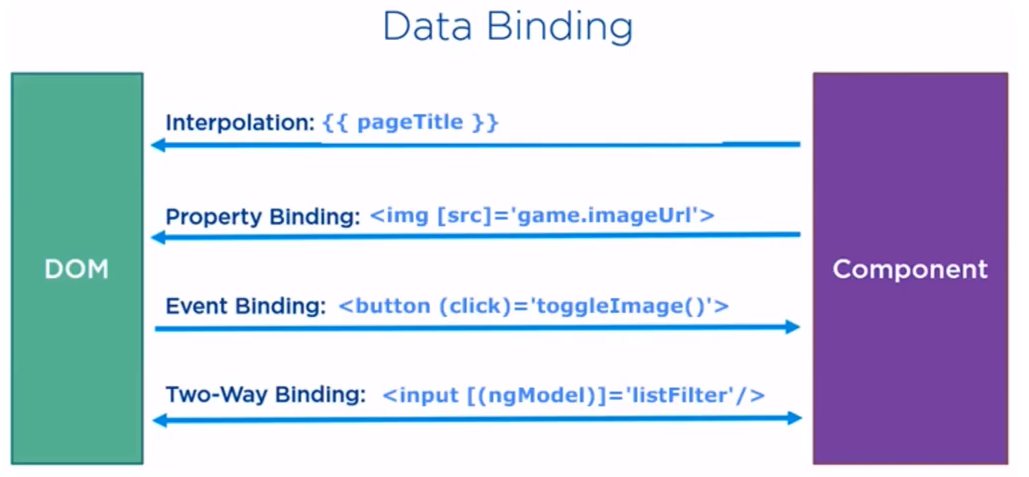
**Data Biding**

* quando quero que o dado venha do componente pra minha DOM (view), posso fazer através de interpolação, property binding



* **interpolação:** {{ valor }}. associa informação do componente para o template (HTML)
* **property binding:** [ propriedade ]=””valor”. associa informação do componente para o template (HTML)
* **event binding:** (evento)=”handler”. associa informação do template (HTML) para o componente
* **two-way data binding:** [(ngModel)]=”propriedade”. associa informação entre ambos, ou seja, mantém ambos atualizados (componente e template (HTML)).

**property binding**

<img *[src]*= "imageUrl" [*alt]*="imageDesc">

**interpolação**

<h1> o nome é {{ retornaNome() }}</h1>

<h1 *[class]*="textRed" *class*="{{textRed}}"> o nome é {{ retornaNome() }}</h1>

**event binding**

<button *(click)*="clicou">Clique</button>

<input *#input* *(keyup)*="clicou(input.value)">

**Mudando o style através do angular**

no arquivo HTML:

<img *[src]*= "imageUrl" *[alt]*="imageDesc" *[style.width.px]*="widthImg">

<button *[attr.aria-Label]*="buttonText" *[style.backgroundColor]*="bgColor" *[style.color]*="fontColor"> {{buttonText}} </button>

no arquivo componente.ts

  bgColor = 'black';

  fontColor = 'white';

  widthImg = 600;

**two way data binding**

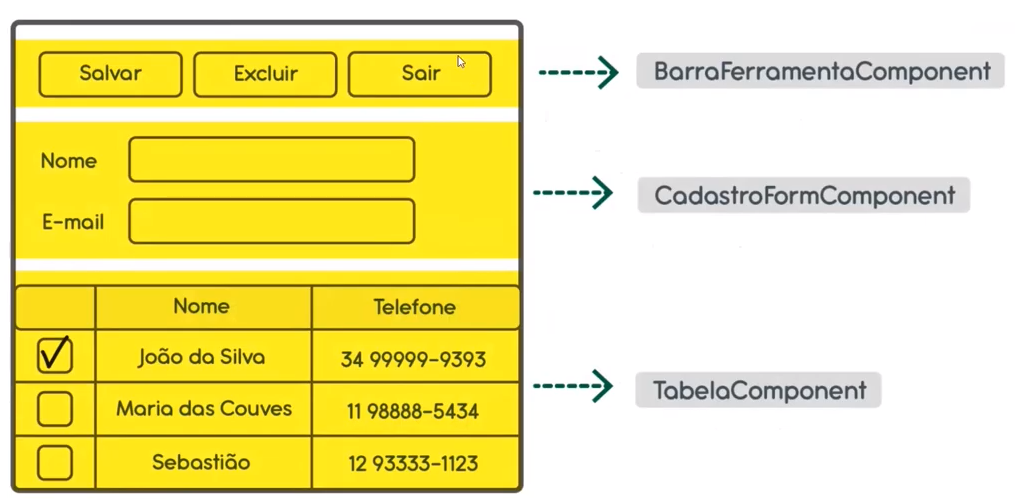
<input *type*="text" *class*="form-control" *[(ngModel)]*="textInput">

<p *class*="text-default">  {{text}}  </p>

**Componentes**

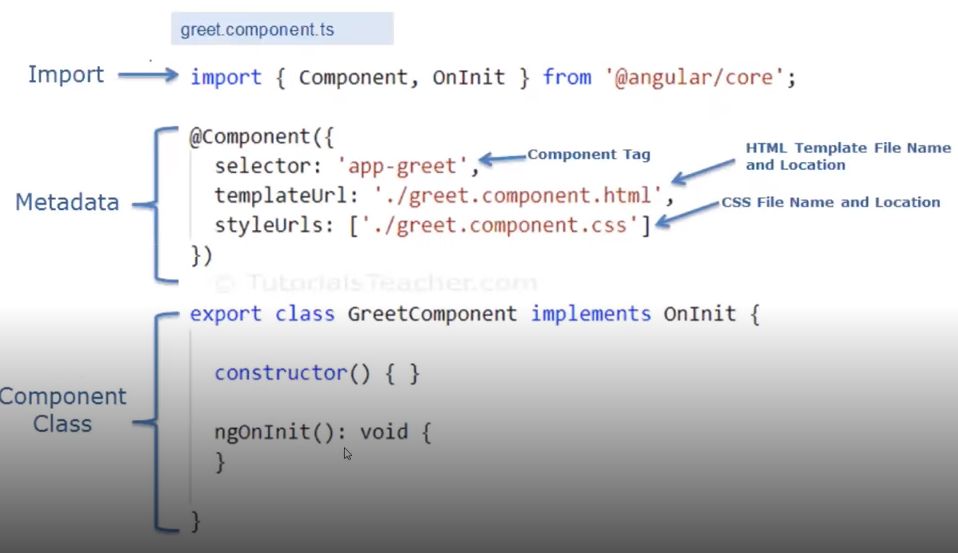
Os **componentes** são blocos que sozinhos tem um sentido. Eles podem ser únicos, ou um conjunto de vários outros componentes.

Quando utilizamos da componentização, temos muitos ganhos, dos quais podemos citar: **a reutilização de trechos de código; isolamento de contexto; legibilidade do código; redução de código** e a **padronização do projeto.**



**Componentes Angular**

* **metadata**: são definições que iremos informar ao Angular que a nossa classe é um Component por meio de decorator, nesse caso @Component
* **class**: assim como em qualquer linguagem de programação que utiliza o paradigma da orientação a objeto, possui suas propriedades e métodos

****

**Pipes**

Pipes são funções simples que você pode usar em expressões nos Templates para aceitar um valor de entrada e retornar um valor transformado