



Introdução aos Componentes Angular: `@Input()` e `@Output()`

September 25, 2023

Agenda

- Introdução ao Angular
- Compreendendo os Componentes Angular
- @Input() in Angular
- Exemplo de uso do @Input()
- @Output() in Angular
- Exemplo de uso do @Output()
- Dicas para usar @Input() e @Output()
- Como @Input() e @Output() trabalham juntos
- Erros comuns ao usar @Input() e @Output()
- Resumo e próximos passos

Introdução ao Angular

01

O Angular é um framework de desenvolvimento de aplicações web mantido pela Google.

02

Os componentes são a base do Angular e são utilizados para criar elementos reutilizáveis e interativos.

03

Algumas das principais diretivas de componentes no Angular são @Input() e @Output().



Plus tip:

Compreendendo os Componentes Angular

@Input()

- A diretiva @Input() é usada para receber dados de um componente pai para um componente filho.
- Permite que os componentes se comuniquem enviando dados de um componente pai para um componente filho.
- Os dados são passados através de propriedades decoradas com o decorator @Input().

@Output()

- A diretiva @Output() é usada para emitir eventos de um componente filho para um componente pai.
- Permite que os componentes se comuniquem enviando eventos de um componente filho para um componente pai.
- Os eventos são emitidos através de propriedades decoradas com o decorator @Output() e podem ser capturados pelo componente pai.



Plus tip:

@Input() in Angular

Definition



O decorato `@Input()` é usado para receber dados de um componente pai para um componente filho. Ele permite que dados sejam passados de um componente pai para um componente filho por meio de uma propriedade.

Usage



Suponha que você tenha um componente filho 'ChildComponent' é um componente pai 'ParentComponent'. Para passar dados do pai para o filho, você pode usar o decorador `@Input()` no filho e definir uma propriedade com o mesmo nome do decorador. No pai, você pode vincular o valor desejado a esta propriedade do filho usando a sintaxe de ligação de propriedade no modelo HTML.

Code Example



Próximo Slide

Exemplo

ParentComponent:

```
@Component({
  template: `<app-child [data]="parentData"></app-child>`
})
export class ParentComponent {
  parentData: string = 'Parent data';
}
```

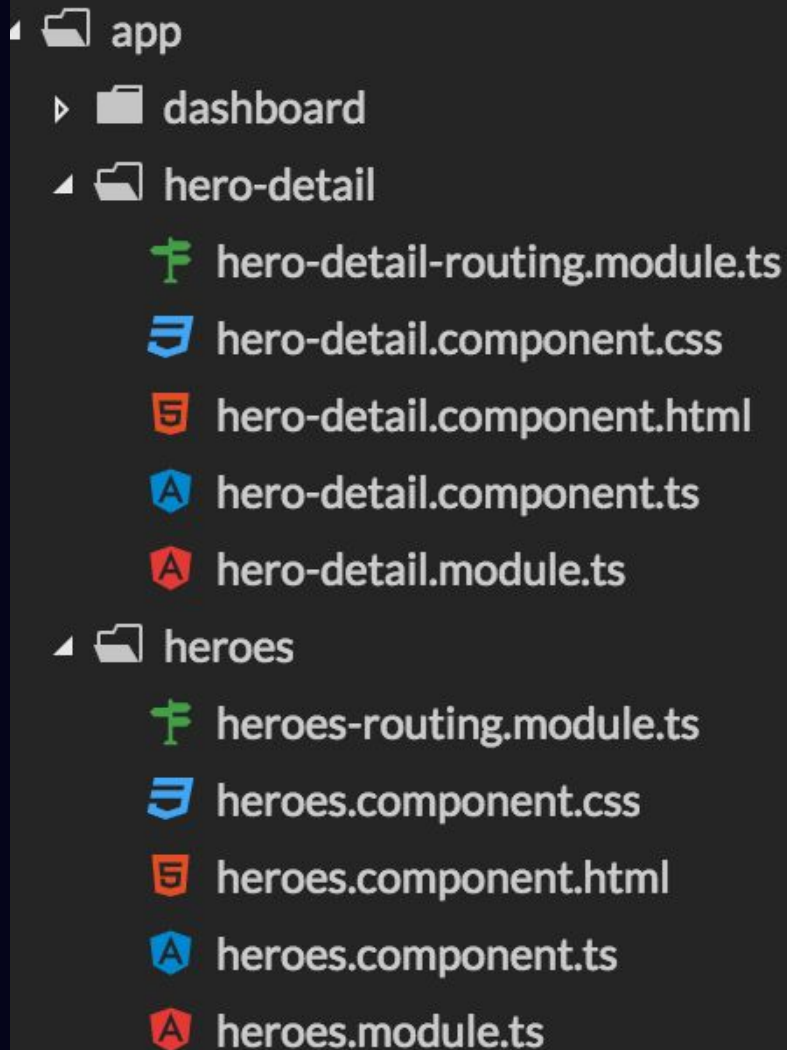
ChildComponent:

```
@Component({
  template: `<h2>Child component data: {{ data }}</h2>`
})
export class ChildComponent {
  @Input() data: string;
}
```

Exemplo de uso do @Input()

@Input()

- O @Input() é um decorator do Angular que permite a passagem de dados de um componente pai para um componente filho.
- Ele é usado para receber valores externos e atualizar a propriedade correspondente no componente filho.
- Isso permite a comunicação entre componentes e facilita a reutilização de código.
- O @Input() é definido na propriedade do componente filho que deseja receber o valor do componente pai.



@Output() in Angular

Definition

@Output() is a decorator in Angular that allows communication from a child component to a parent component. It's used to emit custom events and pass data from the child component to the parent component.

Usage

To use @Output(), define a custom event in the child component using EventEmitter. The custom event can then be emitted using the emit() method. The parent component can subscribe to the custom event using event binding syntax in the child component's template.

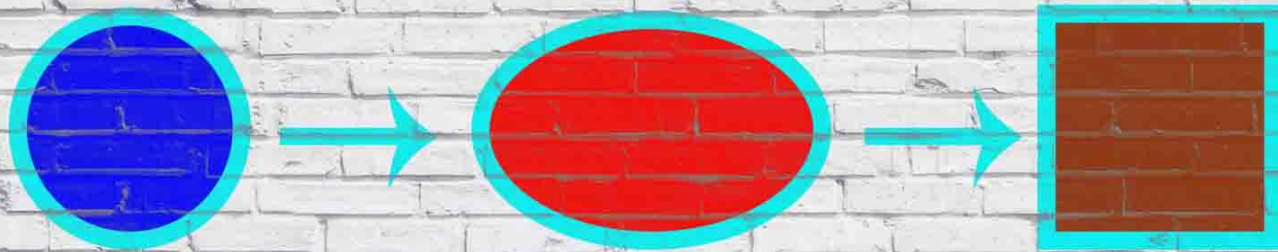
Example

For instance, if you have a parent component that displays a list of items and a child component that allows adding new items to the list. @Output() can be used to emit an event when a new item is added in the child component and the parent component can subscribe to this event to update the item list.

Exemplo de uso do @Output()


Exemplo de uso do @Output()

- O decorador @Output() é usado para criar eventos personalizados em um componente Angular.
- Um evento personalizado pode ser emitido por um componente filho e capturado por um componente pai.
- Isso permite a comunicação entre componentes em diferentes níveis da hierarquia do Angular.
- A sintaxe do @Output() envolve a criação de um EventEmitter e a emissão do evento usando o método emit().




Getting started with Angular 2 Animation


Dicas para usar @Input() e @Output()



Utilize @Input() para receber dados de um componente pai para um componente filho.



Utilize @Output() para emitir eventos de um componente filho para um componente pai.



Ao utilizar @Input() e @Output(), tenha em mente a hierarquia de componentes e a comunicação entre eles.

Como @Input() e @Output() trabalham juntos

@Input()

- O decorador @Input() é usado para passar dados de um componente pai para um componente filho.
- Ele permite que um valor seja passado para uma propriedade do componente filho através da propriedade de entrada correspondente.
- Isso permite que o componente filho receba dados do componente pai e os utilize para fins de exibição ou cálculos internos.
- Para usar o decorador @Input(), é necessário importá-lo do pacote @angular/core e declará-lo acima da propriedade do componente filho que receberá os dados.

@Output()

- O decorador @Output() é usado para emitir eventos de um componente filho para um componente pai.
- Ele permite que um evento seja emitido a partir do componente filho e capturado pelo componente pai.
- Isso permite que o componente filho notifique o componente pai sobre a ocorrência de uma ação ou evento específico.
- Para usar o decorador @Output(), é necessário importá-lo do pacote @angular/core e declará-lo acima do evento que será emitido pelo componente filho.

Erros comuns ao usar @Input() e @Output()



Não importar corretamente as classes dos decoradores



Não definir corretamente os tipos de dados dos inputs e outputs



Não atribuir valores e emitir eventos corretamente



Plus tip:

Resumo e próximos passos



Resumo

Nesta apresentação, você aprendeu sobre os componentes Angular e como usá-los em seu projeto. Os componentes são blocos de construção fundamentais do Angular e permitem criar aplicativos web dinâmicos e interativos.

Próximos passos

Agora que você entende os conceitos básicos dos componentes Angular, é hora de praticar e explorar mais recursos. Aqui estão alguns próximos passos para você:

1. Crie seus próprios componentes

Experimente criar seus próprios componentes em um projeto Angular. Comece com componentes simples e, em seguida, adicione recursos mais avançados à medida que ganha confiança.