# Angular.

# Plano de Aula

## Módulo 1: Introdução ao Angular

* **Objetivos:**
  + Apresentar o Angular e seus principais conceitos.
  + Configurar o ambiente de desenvolvimento.
  + Criar um projeto Angular básico.
* **Tópicos:**
  + O que é Angular e suas vantagens.
  + Diferenças entre Angular e AngularJS.
  + Instalação do Node.js e Angular CLI.
  + Estrutura de um projeto Angular.
  + Criação de componentes básicos.

## Módulo 2: Componentes e Templates

* **Objetivos:**
  + Aprofundar o conhecimento sobre componentes.
  + Aprender a criar templates e utilizar data binding.
  + Explorar diretivas e pipes.
* **Tópicos:**
  + Criação de componentes reutilizáveis.
  + Data binding (interpolação, property binding, event binding, two way binding).
  + Diretivas (ngIf, ngFor, ngSwitch).
  + Pipes (formatar dados).

## Módulo 3: Serviços e Injeção de Dependência

* **Objetivos:**
  + Entender o conceito de serviços.
  + Aprender a utilizar injeção de dependência.
  + Realizar requisições HTTP.
* **Tópicos:**
  + Criação de serviços.
  + Injeção de dependência.
  + Comunicação com APIs (HttpClient).

## Módulo 4: Roteamento e Navegação

* **Objetivos:**
  + Implementar roteamento em um aplicativo Angular.
  + Criar rotas e navegação entre componentes.
  + Utilizar parâmetros de rota.
* **Tópicos:**
  + Configuração de rotas.
  + Navegação entre componentes.
  + Passagem de parâmetros via rota.

## Módulo 5: Formulários e Validação

* **Objetivos:**
  + Criar formulários em Angular.
  + Implementar validação de formulários.
  + Utilizar formulários reativos e template-driven.
* **Tópicos:**
  + Formulários template-driven.
  + Formulários reativos.
  + Validação de formulários.

## Módulo 6: Testes e Deploy

* **Objetivos:**
  + Escrever testes unitários e de integração.
  + Realizar o deploy de um aplicativo Angular.
* **Tópicos:**
  + Testes unitários com Jasmine e Karma.
  + Testes de integração.
  + Build e deploy para produção.
  + Apresentar o Angular e seus principais conceitos.

# Apresentar o Angular e seus principais conceitos.

O Angular é um framework de desenvolvimento web de código aberto, baseado em TypeScript, mantido pelo Google. Ele é usado para construir aplicativos web de página única (SPAs), que são aplicativos que carregam uma única página HTML e atualizam dinamicamente o conteúdo à medida que o usuário interage com o aplicativo.

## Principais conceitos do Angular:

### Componentes:

* + Os componentes são os blocos de construção fundamentais de um aplicativo Angular. Eles encapsulam a lógica de negócios e a apresentação em um único elemento, tornando-os reutilizáveis e fáceis de manter.
  + Cada componente é composto por três elementos: um template HTML, uma classe TypeScript e um arquivo de estilo CSS. 1

### Módulos:

* + Os módulos organizam os componentes, serviços e outros recursos relacionados em blocos funcionais.
  + O módulo raiz, AppModule, é o ponto de entrada do aplicativo.

### Templates:

* + Os templates definem a estrutura e a aparência da interface do usuário.
  + Eles usam HTML estendido com recursos do Angular, como data binding e diretivas.

### Data Binding:

* + O data binding permite sincronizar automaticamente os dados entre o modelo (a lógica do componente) e a visualização (o template).
  + O Angular oferece diferentes tipos de data binding, como interpolação, property binding e event binding.

### Diretivas:

* + As diretivas modificam o comportamento ou a aparência dos elementos DOM.
  + Existem diretivas internas, como ngIf e ngFor, e diretivas personalizadas que você pode criar.

### Serviços:

* + Os serviços são classes que encapsulam a lógica de negócios reutilizável.
  + Eles podem ser injetados em componentes e outros serviços usando injeção de dependência.

### Injeção de dependência:

* + A injeção de dependência é um padrão de design que permite fornecer dependências a um componente ou serviço sem que ele precise criá-las.
  + Isso torna o código mais modular e testável.

### Roteamento:

* + O roteamento permite navegar entre diferentes visualizações em um aplicativo de página única.
  + O Angular fornece um sistema de roteamento poderoso para definir rotas e navegar entre componentes.

## Outros pontos importantes:

### TypeScript:

* + O Angular é construído em TypeScript, uma linguagem que adiciona tipagem estática ao JavaScript.
  + O TypeScript ajuda a detectar erros durante o desenvolvimento e melhora a qualidade do código.

### Angular CLI:

* + O Angular CLI é uma ferramenta de linha de comando que simplifica o desenvolvimento de aplicativos Angular.
  + Ele permite criar projetos, componentes, serviços e outros recursos com facilidade.

# Configurar o ambiente de desenvolvimento.

Para configurar o ambiente de desenvolvimento Angular, você precisará instalar algumas ferramentas e configurar seu sistema. Aqui está um guia passo a passo:

## 1. Instale o Node.js e o npm:

* O Node.js é um ambiente de tempo de execução JavaScript que permite executar JavaScript fora de um navegador. O npm (Node Package Manager) é o gerenciador de pacotes padrão para Node.js.
* Vá para o site oficial do Node.js (nodejs.org) e baixe a versão LTS (Long Term Support) recomendada.
* Durante a instalação, certifique-se de que a opção "Adicionar ao PATH" esteja selecionada.
* Após a instalação, abra o terminal ou prompt de comando e execute os seguintes comandos para verificar se a instalação foi bem-sucedida:
  + node -v
  + npm -v
* Estes comandos devem exibir as versões instaladas do Node.js e do npm.

## 2. Instale o Angular CLI:

* O Angular CLI (Command Line Interface) é uma ferramenta de linha de comando que simplifica o desenvolvimento de aplicativos Angular.
* No terminal ou prompt de comando, execute o seguinte comando para instalar o Angular CLI globalmente:
  + npm install -g @angular/cli
* Após a instalação, execute o seguinte comando para verificar a versão do Angular CLI:
  + ng version

## 3. Instale um editor de código:

* Um editor de código é essencial para escrever e editar código Angular. Algumas opções populares incluem:
  + Visual Studio Code (recomendado)
  + Sublime Text
  + WebStorm
* O Visual Studio Code possui diversas extensões que facilitam o desenvolvimento angular.

## 4. Crie um novo projeto Angular:

* No terminal ou prompt de comando, navegue até o diretório onde você deseja criar o projeto.
* Execute o seguinte comando para criar um novo projeto Angular:
  + ng new nome-do-projeto
* O Angular CLI irá solicitar algumas informações sobre o projeto, como se você deseja adicionar roteamento e qual formato de folha de estilo usar.
* Após a criação do projeto, navegue até o diretório do projeto:
  + cd nome-do-projeto

## 5. Execute o aplicativo:

* No terminal ou prompt de comando, execute o seguinte comando para iniciar o servidor de desenvolvimento:
  + ng serve --open
* Este comando irá compilar o aplicativo e abri-lo em seu navegador padrão.

# Criar um projeto Angular básico.

Criar um projeto Angular básico é um processo simples usando o Angular CLI (Command Line Interface). Aqui estão os passos detalhados:

## Pré-requisitos:

* **Node.js e npm:** Certifique-se de que o Node.js e o npm estejam instalados em sua máquina. Você pode verificar as versões com os comandos node -v e npm -v no terminal.
* **Angular CLI:** Instale o Angular CLI globalmente com o comando npm install -g @angular/cli.

## Passos para criar um projeto Angular:

### Abra o terminal:

* + Abra o terminal ou prompt de comando em seu sistema operacional.

### Navegue até o diretório desejado:

* + Use o comando cd para navegar até o diretório onde você deseja criar o projeto. Por exemplo, cd Documentos/Projetos.

### Crie o projeto:

* + Execute o seguinte comando para criar um novo projeto Angular:
    - ng new nome-do-projeto
  + Substitua nome-do-projeto pelo nome que você deseja dar ao seu projeto.
  + O Angular CLI irá solicitar algumas informações:
    - **Would you like to add Angular routing? (Você gostaria de adicionar roteamento Angular?)** Responda y para sim ou n para não. O roteamento é útil para aplicativos com várias páginas.
    - **Which stylesheet format would you like to use? (Qual formato de folha de estilo você gostaria de usar?)** Escolha entre CSS, SCSS, Sass, Less ou Stylus. CSS é a opção padrão.

### Aguarde a instalação:

* + O Angular CLI irá criar a estrutura do projeto e instalar as dependências necessárias. Isso pode levar alguns minutos.

### Navegue até o diretório do projeto:

* + Use o comando cd nome-do-projeto para entrar no diretório do projeto que você acabou de criar.

### Execute o aplicativo:

* + Execute o comando ng serve --open para compilar e iniciar o servidor de desenvolvimento. O aplicativo será aberto automaticamente em seu navegador padrão.

## Estrutura básica do projeto:

Após a criação do projeto, você terá uma estrutura de diretórios semelhante a esta:

* **src:** Contém o código-fonte do seu aplicativo.
  + **app:** Contém os componentes, serviços e outros arquivos do seu aplicativo.
    - **app.component.ts:** O componente raiz do aplicativo.
    - **app.module.ts:** O módulo raiz do aplicativo.
  + **assets:** Contém arquivos estáticos, como imagens e arquivos de estilo.
  + **environments:** Contém arquivos de configuração para diferentes ambientes (desenvolvimento, produção).
  + **index.html:** O arquivo HTML principal do aplicativo.
* **angular.json:** Arquivo de configuração do Angular CLI.
* **package.json:** Arquivo de configuração do npm, contendo as dependências do projeto.

# O que são componentes em Angular.

Em Angular, componentes são os blocos de construção fundamentais de uma aplicação. Eles representam partes independentes e reutilizáveis da interface do usuário (UI), encapsulando lógica, dados e apresentação.

## Principais características dos componentes Angular:

* **Reusabilidade:**
  + Componentes podem ser reutilizados em diferentes partes da aplicação, promovendo um código mais organizado e eficiente.
* **Encapsulamento:**
  + Cada componente encapsula seu próprio HTML (template), CSS (estilos) e lógica (TypeScript), evitando conflitos e facilitando a manutenção.
* **Estrutura modular:**
  + Aplicações Angular são construídas combinando múltiplos componentes, criando uma estrutura modular e hierárquica.
* **Comunicação:**
  + Componentes podem se comunicar entre si através de entradas (inputs) e saídas (outputs), permitindo a troca de dados e a interação entre diferentes partes da UI.

## Estrutura de um componente Angular:

Um componente Angular é composto por três partes principais:

* **Template (HTML):**
  + Define a estrutura e a aparência do componente, utilizando HTML estendido com recursos do Angular, como data binding e diretivas.
* **Classe (TypeScript):**
  + Contém a lógica do componente, incluindo a definição de dados, métodos e interações com outros componentes e serviços.
* **Estilos (CSS):**
  + Define a aparência visual do componente, utilizando CSS para estilizar os elementos HTML do template.

## Por que usar componentes?

* **Organização:** Componentes ajudam a organizar o código em partes menores e mais gerenciáveis.
* **Manutenção:** Componentes facilitam a manutenção do código, pois as alterações em um componente não afetam outros componentes.
* **Testabilidade:** Componentes são mais fáceis de testar individualmente, o que melhora a qualidade do código.
* **Reutilização:** Componentes podem ser reutilizados em diferentes partes da aplicação, o que economiza tempo e esforço.

Em resumo, os componentes são essenciais para o desenvolvimento de aplicações Angular modernas e escaláveis. Eles permitem criar interfaces de usuário complexas e interativas de forma organizada, eficiente e reutilizável.

## Criar um componente em angular

Criar um componente em Angular é um processo simples, especialmente com o auxílio do Angular CLI (Command Line Interface). Aqui estão os passos detalhados para criar um componente:

### Usando o Angular CLI:

* A maneira mais rápida e recomendada de criar um componente é usando o Angular CLI.
* Abra o terminal e navegue até o diretório do seu projeto Angular.
* Execute o seguinte comando:
  + ng generate component nome-do-componente
  + Ou, de forma abreviada:
  + ng g c nome-do-componente
  + Substitua nome-do-componente pelo nome que você deseja dar ao seu componente. Por exemplo: ng g c minha-lista.
* O Angular CLI irá gerar os seguintes arquivos:
  + nome-do-componente.component.ts: Arquivo TypeScript que contém a lógica do componente.
  + nome-do-componente.component.html: Arquivo HTML que define o template do componente.
  + nome-do-componente.component.css: Arquivo CSS que contém os estilos do componente.
  + nome-do-componente.component.spec.ts: Arquivo de teste unitário para o componente.
* O Angular CLI também irá atualizar o arquivo app.module.ts para declarar o novo componente.

## Exemplo de um componente simples:

Aqui está um exemplo básico de um componente criado com o Angular CLI:

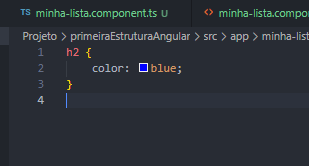
* **minha-lista.component.ts:**



* **minha-lista.component.html:**



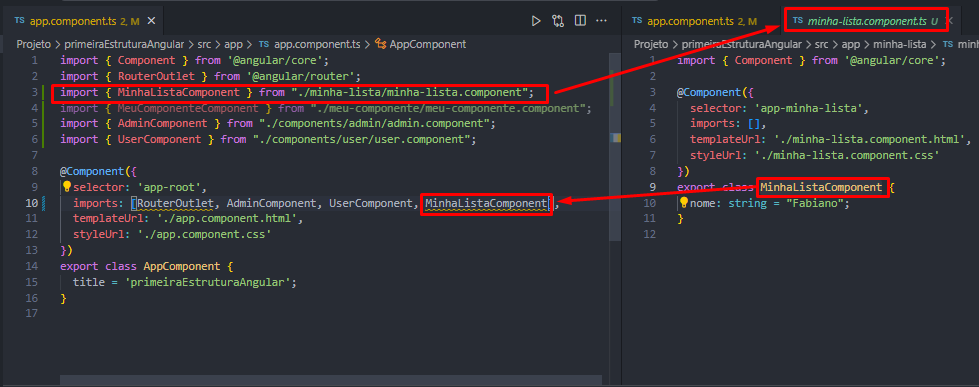
* **minha-lista.component.css:**



## Utilizando o componente:

Após criar o componente, você pode utilizá-lo em outros componentes ou no template do componente raiz (app.component.html).

Note que ao criar um componente novo, este é importado automaticamente para e colocado na propriedade imports do arquivo app.component.ts.



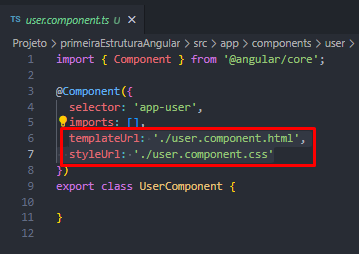
Para utilizar o componente criado, basta adicionar a tag do componente no template:

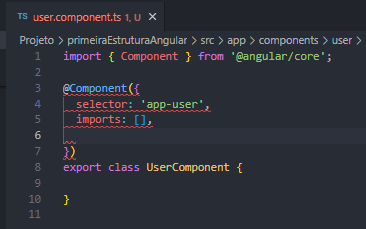
<app-minha-lista></app-minha-lista>



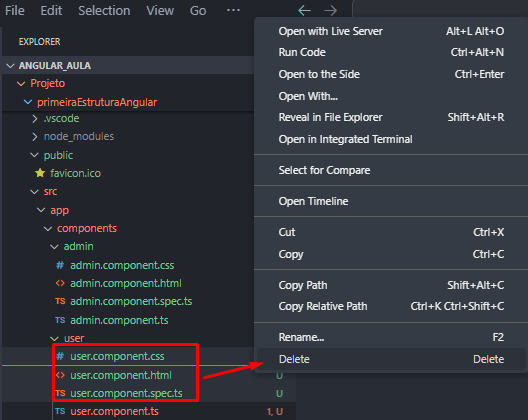
## Criar um arquivo simples de componente.

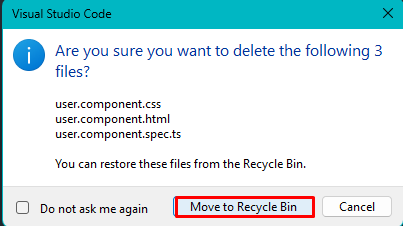
Apagar as linhas do user.component.ts:



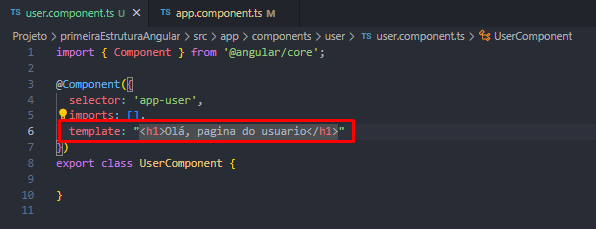


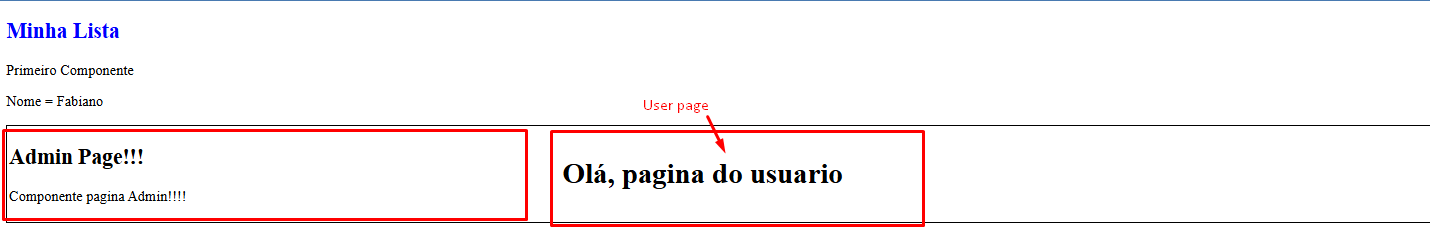
### Apagar os arquivos user.component.css, user.component.html, user.component.spec.ts.



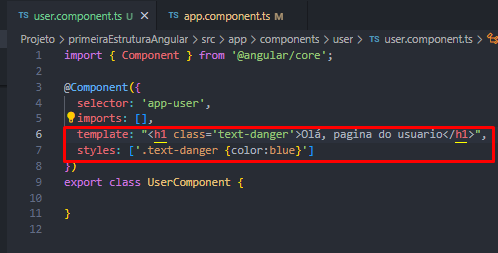


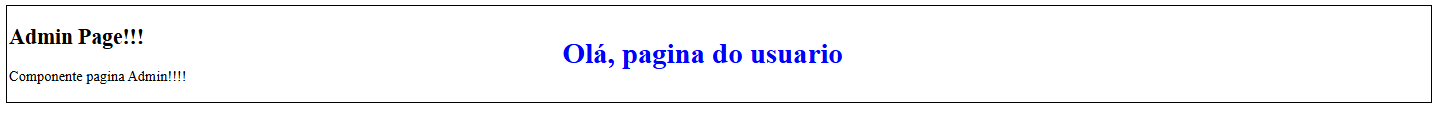
### Acrescentar no user.component.ts no template:





### Criar um estilo para componente:





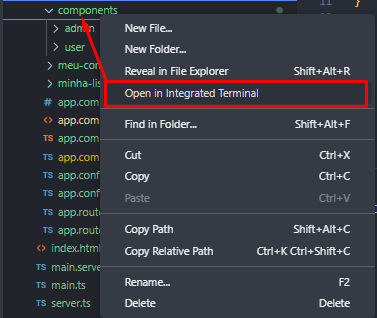
## Exercício:

### Criar uma pasta componentes dentro da pasta app.



### Abrir o terminal na pasta componentes criada.

Clicar com o botão direito apontando para a pasta componente, selecionar, Open in Integrated Terminal.



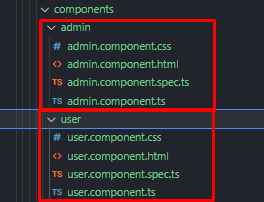
### Criar um componente admin:



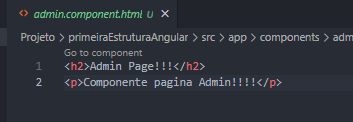
### Criar um componente user. Note que o comando pode ser abreviado:



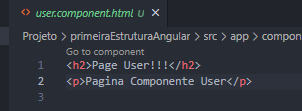
### Após criar os novos componentes a estrutura na pasta componentes deve ser:



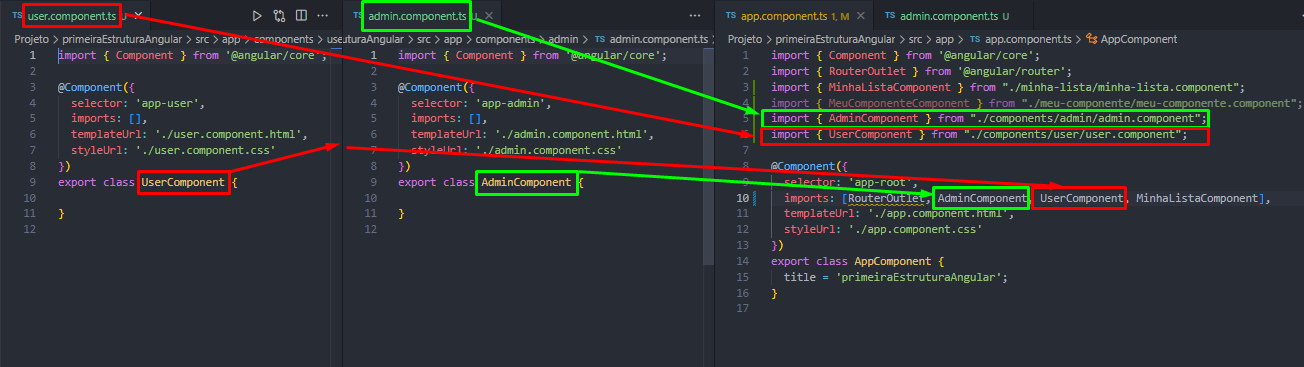
### O Arquivo admin.component.html deve ser alterado:



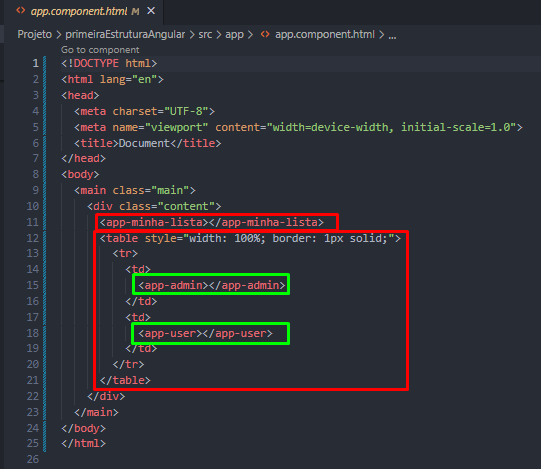
### O arquivo user.component.html deve ser alterado:



### Verificar a importação dos arquivos no app.component.ts.



### Chamar o componente no app.component.html



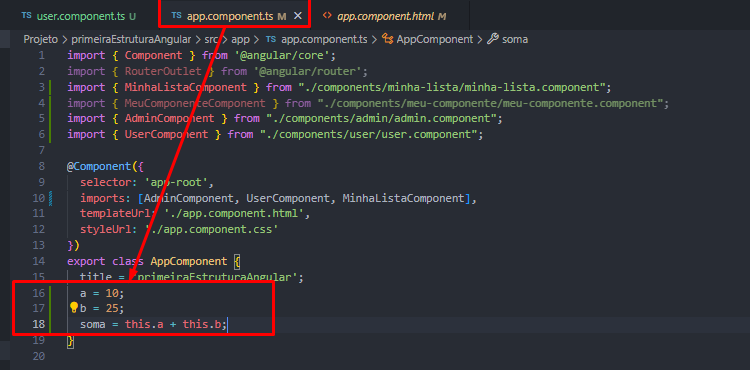
## **Interpolação em Angular**

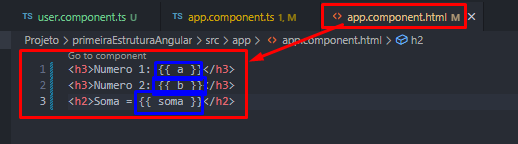
A interpolação no Angular é uma técnica poderosa que permite incorporar valores dinâmicos de variáveis e expressões TypeScript diretamente no seu template HTML. É uma forma de comunicação unidirecional de dados, onde os dados do componente são exibidos na visualização.

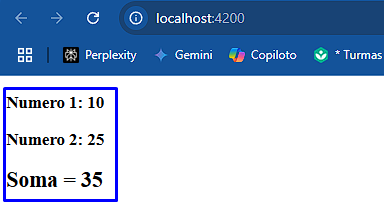
### **Como funciona:**

* **Sintaxe**: A interpolação utiliza a sintaxe de chaves duplas **{{ expressão }}**. O Angular avalia a expressão dentro das chaves e substitui o resultado no HTML.
* **Expressões**: Dentro das chaves, você pode usar variáveis do componente, propriedades de objetos, chamadas de funções e até mesmo expressões matemáticas simples.
* **Exibição**: O resultado da expressão é convertido em uma string e exibido no local onde a interpolação foi usada.
* **Atualização dinâmica**: O Angular monitora as variáveis usadas na interpolação. Se um valor mudar no componente, a visualização é atualizada automaticamente.

### **Exemplo: Interpolação da soma de sois valores.**







## **DataBind**

Em Angular, o databinding é um mecanismo que permite a sincronização automática de dados entre o modelo (componente TypeScript) e a visualização (template HTML). Isso significa que as alterações feitas no modelo são refletidas automaticamente na visualização, e vice-versa.

### Como funciona:

O **databinding** no Angular é baseado em um sistema de detecção de mudanças que monitora as propriedades do modelo. Quando uma propriedade é alterada, o Angular atualiza automaticamente a visualização para refletir a nova alteração.

### Tipos de databinding:

Existem quatro tipos principais de databinding no Angular:

#### Interpolação:

Exibe valores do modelo na visualização.

**Sintaxe: {{ valor }}.**

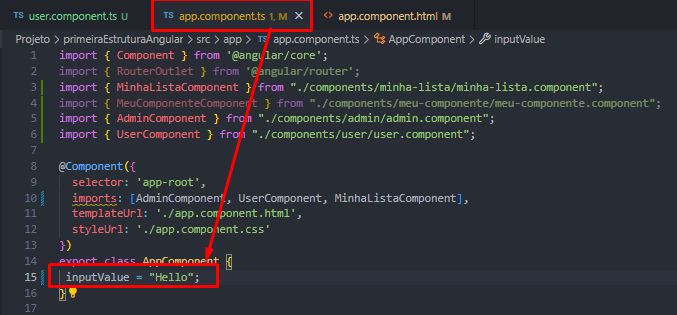
**Exemplo**: **<h1>{{ titulo }}</h1>**

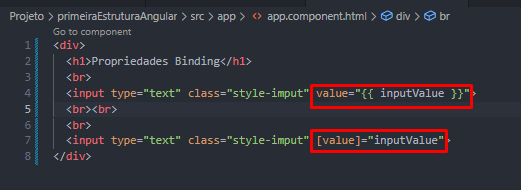
#### Property binding:

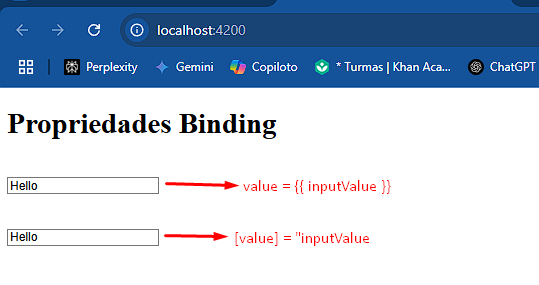
Define valores de propriedades de elementos HTML.

**Sintaxe: [propriedade]="valor"**

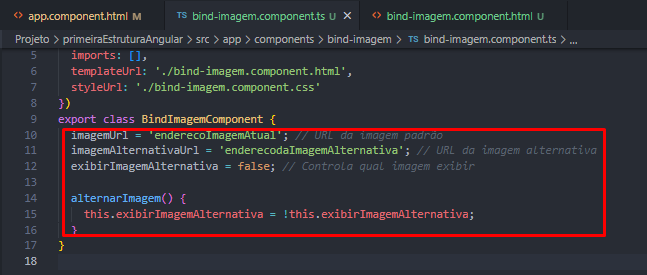
**Exemplo: <img [src]="imagemUrl">**

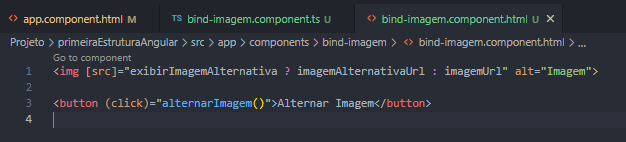


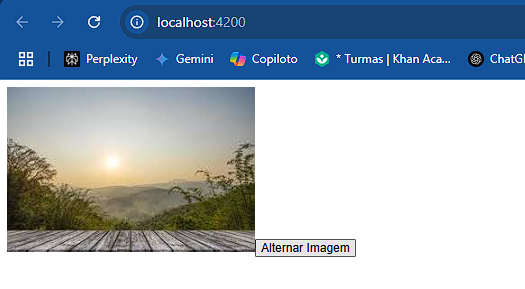




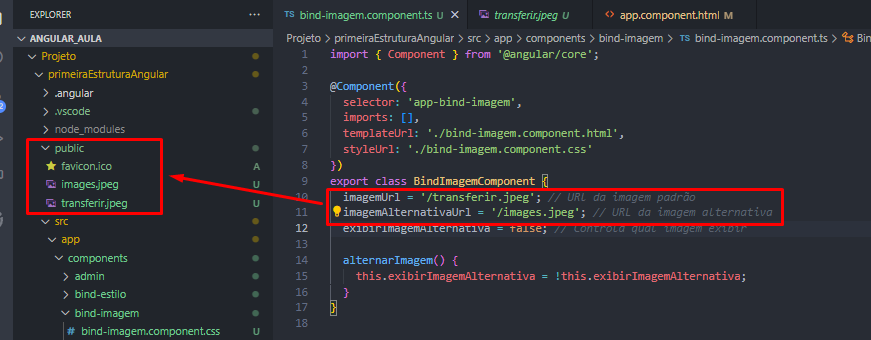
##### Property bind com imagens:





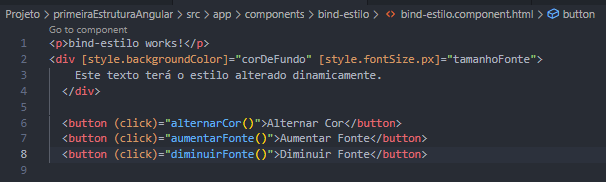


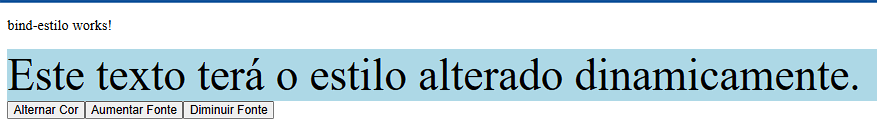
###### Imagem na pasta Public.



##### Property Bind Estilo





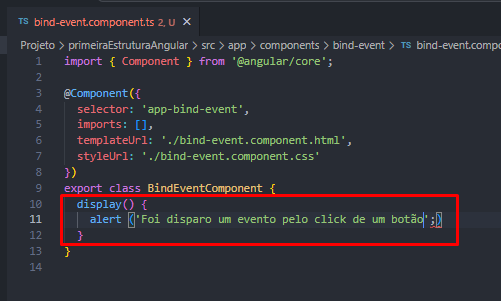


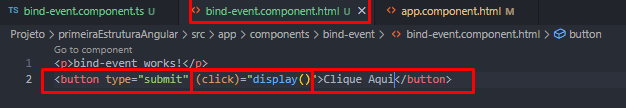
#### Event binding:

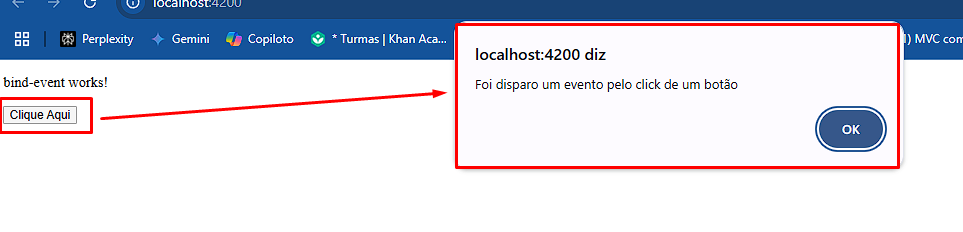
Responde a eventos do DOM, como cliques e entradas de teclado.

**Sintaxe: (evento)="expressão"**

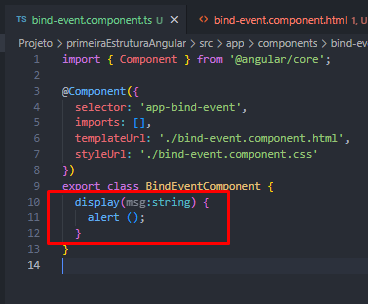
**Exemplo: <button (click)="onClick()">Clique aqui</button>**

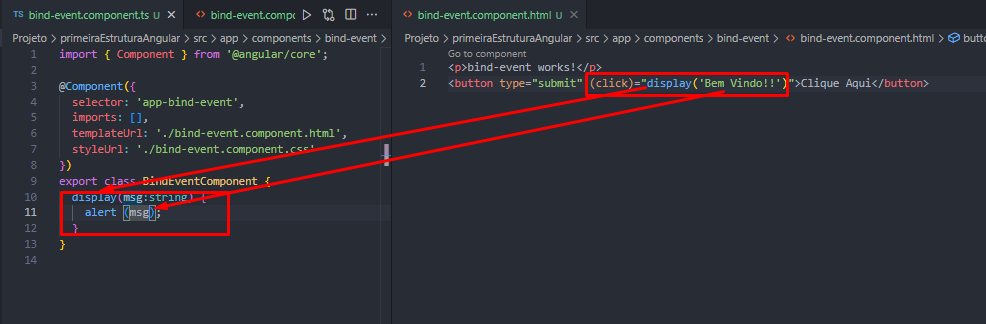


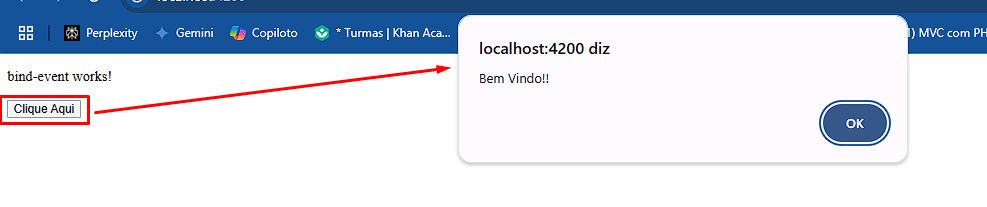




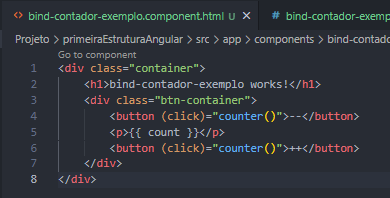
##### Passagem de Parâmetro na Função.

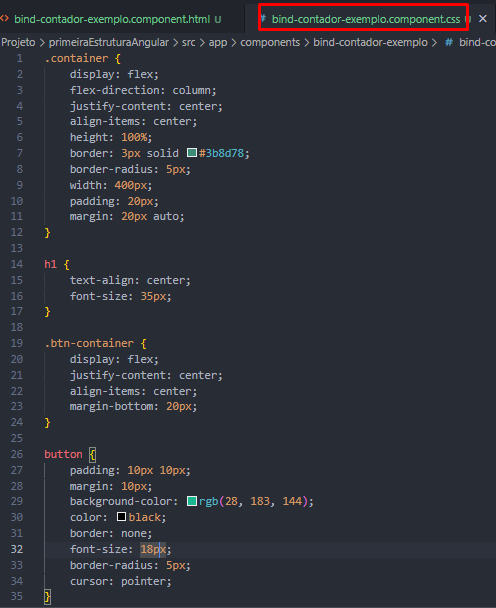




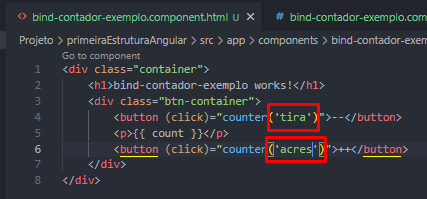


##### Exemplo de Contador:











##### Eventos de teclado.

**keydown:**

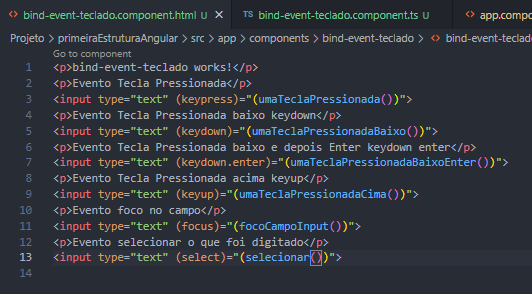
* Este evento é disparado quando uma tecla é pressionada.
* É útil para capturar ações imediatas, como comandos de atalho.
* Fornece informações sobre a tecla pressionada, incluindo códigos e modificadores.

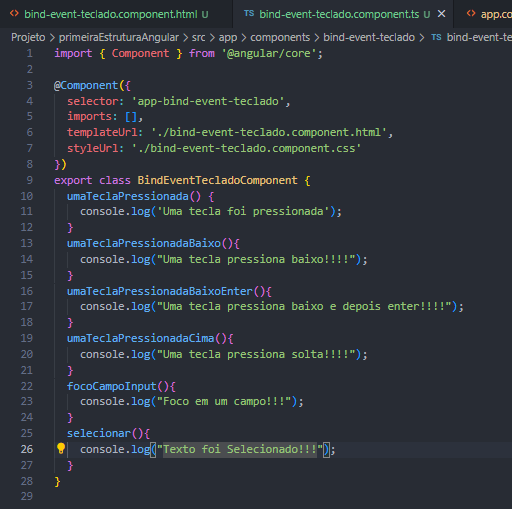
**keypress:**

* Este evento é acionado quando uma tecla que produz um caractere é pressionada.
* É geralmente usado para capturar a entrada de texto.
* Pode ser menos confiável em alguns navegadores e para teclas não caracteres.

**keyup:**

* Este evento ocorre quando uma tecla é liberada.
* É útil para ações que devem ser executadas após a conclusão da entrada, como validar um campo de formulário.
* Para uma melhor utilização dentro do angular, use junto com o keydown, para pegar o valor da key, e usar o keyup para executar uma função após soltar a tecla.





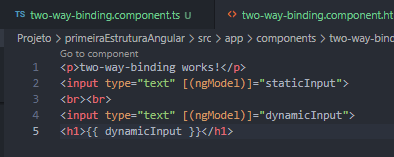
#### Two-way binding:

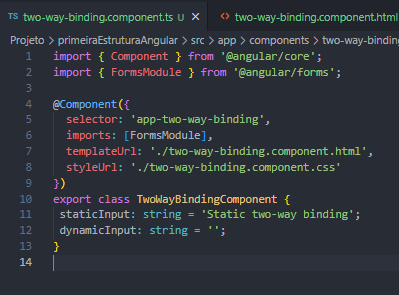
Sincroniza valores entre o modelo e a visualização em ambas as direções.

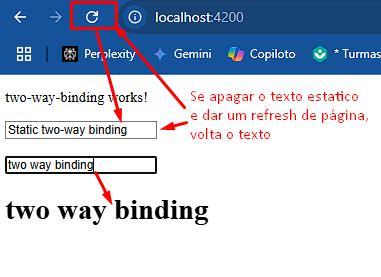
**Sintaxe: [(ngModel)]="valor"**

**Exemplo: <input [(ngModel)]="nome">**

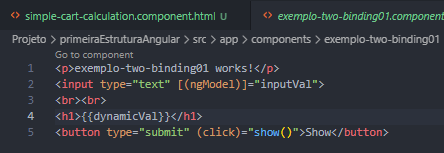
##### Pegando valor digitado em um campo mostrando na tela



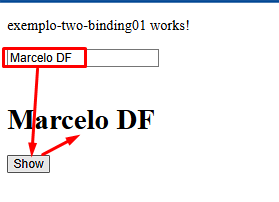




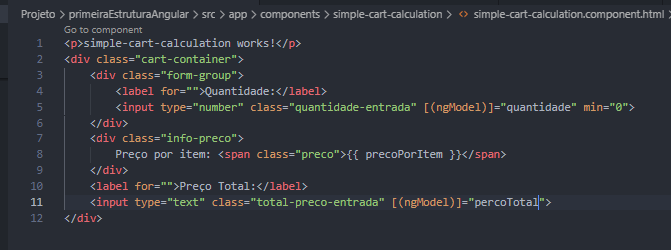
##### Exemplo: Ao clicar em um botão pegar o texto do input e mostrar na tela

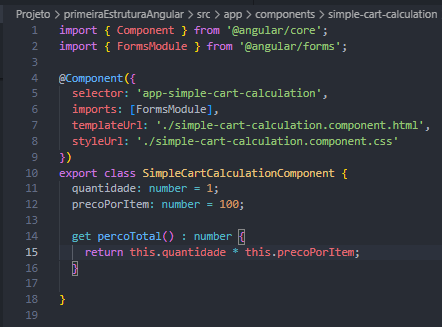


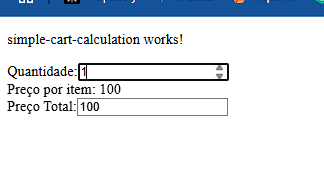


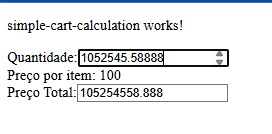


##### Calculo Simples









### Diferença entre Interpolação e DataBind.

#### Interpolação:

**Foco**:

* A interpolação é especificamente para exibir valores de variáveis e expressões do componente diretamente no template HTML.
* É uma forma simples de inserir dados dinâmicos no texto de um elemento HTML.

**Direção**:

* É unidirecional: os dados fluem do componente para o template.

**Sintaxe**:

* Utiliza chaves duplas {{ expressão }}.

**Conversão**:

* Sempre converte os valores para strings antes de exibi-los.

**Uso**:

* Ideal para exibir texto, valores simples e resultados de expressões.

#### Databinding:

**Foco**:

* O databinding é um conceito mais amplo que abrange várias técnicas para sincronizar dados entre o componente e o template.
* Permite definir valores de propriedades de elementos HTML, responder a eventos e realizar sincronização bidirecional.

**Direção**:

* Pode ser unidirecional (do componente para o template ou vice-versa) ou bidirecional (em ambas as direções).

**Sintaxe**:

* Utiliza diferentes sintaxes, como:
* [propriedade]="valor" (property binding),
* (evento)="expressão" (event binding)
* [(ngModel)]="valor" (two-way binding).

**Tipos de Dados**:

Mantem o tipo de dados original, sem converter tudo para string.

**Uso**:

É mais versátil e adequado para cenários mais complexos, como definir atributos de elementos, manipular eventos e criar formulários interativos.

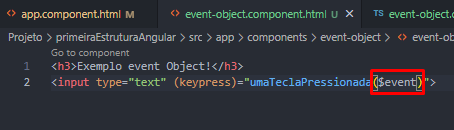
## **Event Object em Angular.**

Em Angular, o objeto $event é uma referência ao objeto de evento DOM que é passado para um manipulador de eventos. Ele contém informações sobre o evento que ocorreu, como o elemento que disparou o evento, o tipo de evento e quaisquer dados adicionais associados ao evento.

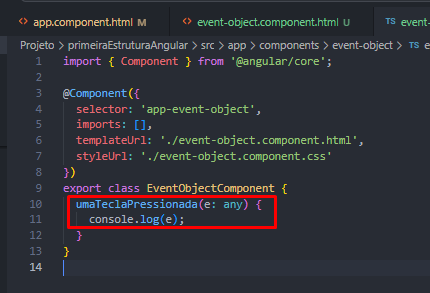
### Como usar o objeto $event

O objeto $event é normalmente usado em templates Angular para acessar informações sobre eventos que foram disparados por elementos HTML. Por exemplo, você pode usar o objeto $event para obter o valor de um campo de entrada quando um usuário digita nele ou para obter as coordenadas do mouse quando um usuário clica em um elemento.

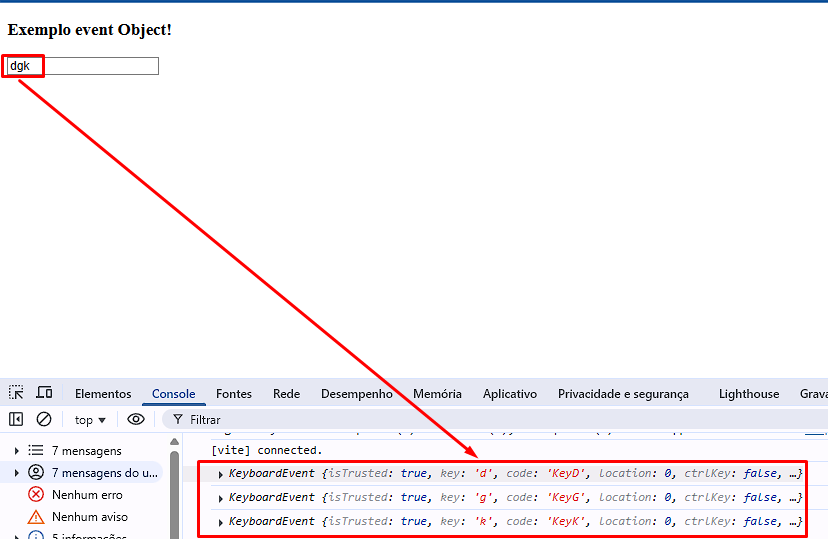
### Exemplo de uso em campo input:

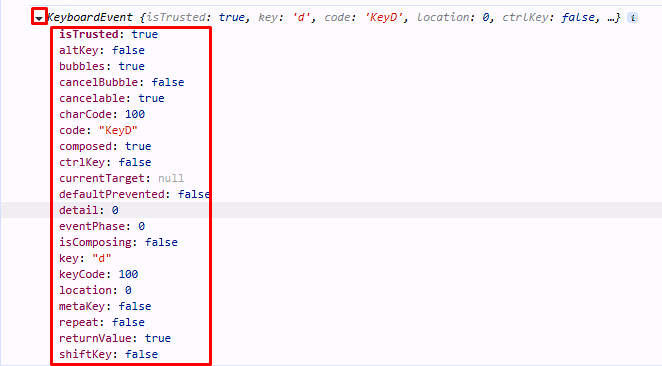


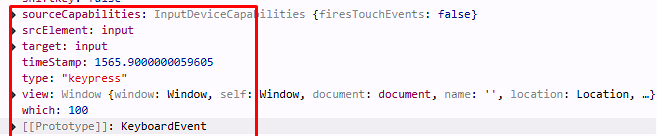
Neste exemplo, o manipulador de eventos umaTeclaPressionada() é chamado sempre que o valor do campo de entrada é alterado. O objeto $event é passado para o manipulador de eventos e pode ser usado para acessar o valor do campo de entrada.



O manipulador de eventos umaTeclaPressionada() é chamado sempre que o valor do campo de entrada é alterado. O objeto $event é passado para o manipulador de eventos e é usado para acessar o valor do campo de entrada. O valor do campo de entrada é então exibido no console.





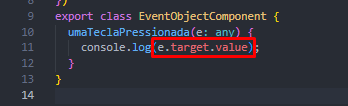


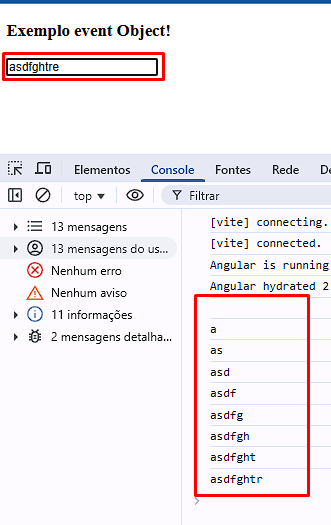
### Propriedades do objeto $event

O objeto $event tem várias propriedades que podem ser usadas para acessar informações sobre o evento. Algumas das propriedades mais comuns incluem:

* target: O elemento que disparou o evento.
* type: O tipo de evento (por exemplo, click, input, submit).
* timeStamp: O tempo em que o evento ocorreu.
* preventDefault(): Um método que pode ser usado para impedir o comportamento padrão do evento.

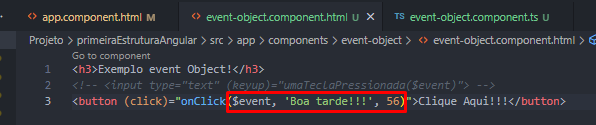
### Utilizando as propriedades do evento. Taregt.value

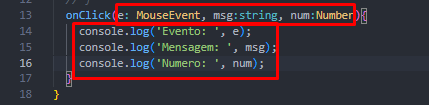




### Passando mais de um evento:

Pode-se passar mais de um argumento dentro de um evento:

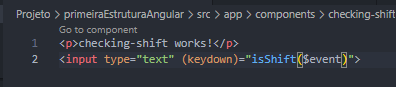


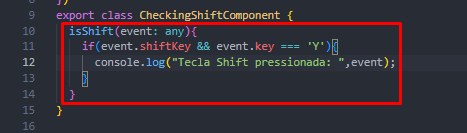




### Exemplos:

#### Verificando se a tecla Shift foi pressionada:





#### Incrementar e decrementar quando as teclas UP e DOWN forem pressionadas.

