React: Props, Children e Fragment

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Fragment
- Props
- Children

FRAGMENT

Fragment

Sempre que precisamos **renderizar mais de um elemento**
em um mesmo componente, precisamos colocá-los dentro de um **componente pai**.

Isso acontece porque, como já vimos, quando usamos JSX
para criar nossos componentes,
o React executa internamente a função ***React.createElement***.

Fragment

Imagine o cenário de 2 elementos parágrafos sendo criados em paralelo:

```
function App() {
  return (
    <p>Paragrafo um </p>
    <p>Parágrafo 2</p>
  );
}
```

Ao transpilar o código são criadas duas funções paralelas.
Não é possível retornar ambas as funções de uma só vez:

```
// Por baixo dos panos
function App() {
    return React.createElement("p", null, "Parágrafo um")
    // Todo código abaixo do return é inacessível
    React.createElement("p", null, "Parágrafo dois")
}
```

Fragment

Há 2 maneiras de resolver esse problema:

- A **<u>criação de um elemento pai</u>** (*div, section, article,...*) que compreende todos os outros elementos
  Porém, isso necessariamente ***cria um elemento HTML adicional*** no nosso código final, e às vezes não precisamos ou não queremos esse elemento extra.

```
function App() {
  return (
    <div>
      <p>Paragrafo 1</p>
      <p>Parágrafo 2</p>
    </div>
  );
}
```

Fragment

- Uso do **Fragment**, um componente criado pelo React que serve como contentor (container) de outros elementos, mas que não cria um HTML desnecessário na tela

```
function App() {
  return (
    <>
      <p>Paragrafo 1</p>
      <p>Parágrafo 2</p>
    </>
  );
}
```

```
function App() {
  return (
    <React.Fragment>
      <p>Paragrafo 1</p>
      <p>Parágrafo 2</p>
    </React.Fragment>
  );
}
```

Fragment

O que acontece “por baixo dos panos” é que as funções React.createElement passam a ser chamadas uma dentro da outra, criando uma **cadeia de funções** que, no fim, criará um ***único elemento renderizado na tela***!

```
function App() {
  return React.createElement(
    React.Fragment,
    null,
    React.createElement("p", null, " Parágrafo 1"),
    React.createElement("p", null, " Parágrafo 2")
  );
}
```

PROPS

PROPS

Quando usamos JSX para criar nossos componentes,
o React executa a seguinte **função** internamente:

```
React.createElement(component, props, ...children)
```

O que são props?

Prop é a abreviação de *property*.
Por debaixo dos panos são **propriedades de um objeto**.
Em diversas situações precisamos compartilhar dados entre componentes,
e fazemos isso através das props.

Props na prática

Vamos criar um componente que irá transformar qualquer texto passado para ele em caixa alta (*letras maiúsculas*): <CapsLock />

Sem props

```
function App() {
  return (
    <div>
      <h1>Meu primeiro componente com props</h1>
      <CapsLock />
    </div>
  );
}

export default App;

function CapsLock() {
  const texto = "Texto que ficará com letras maiúsculas"
  const textoEmLetrasMaiusculas = texto.toUpperCase();
  return <div>{textoEmLetrasMaiusculas}</div>;
}
```

Com props

```
function App() {
  return (
    <div>
      <h1>Meu primeiro componente com props</h1>
      <CapsLock textoEnviado="Texto que ficará com letras maiúsculas"/>
      <CapsLock textoEnviado="Outro texto que ficará com letras maiúsculas"/>
    </div>
  );
}

export default App;

function CapsLock(props) {
  const texto = props.textoEnviado
  const textoEmLetrasMaiusculas = texto.toUpperCase();
  return <div>{textoEmLetrasMaiusculas}</div>;
}
```

Props na prática

Por ser um objeto, podemos usar o método de **destructuring** para acessar os valores das nossas props, que é feita usando chaves **{}**

```
function CapsLock(props) {
  const { textoEnviado } = props;
  const textoEmLetrasMaiusculas = textoEnviado.toUpperCase();
  return <div>{textoEmLetrasMaiusculas}</div>;
}
```

ou

```
function CapsLock({ textoEnviado }) {
  const textoEmLetrasMaiusculas = textoEnviado.toUpperCase();
  return <div>{textoEmLetrasMaiusculas}</div>;
}
```

Props na prática

Quando passamos **apenas o nome** da prop sem informar um valor, o React apropria um tipo de dado **booleano** e o valor true para ela.

```
function App() {
  return (
    <div>
      <h2>True</h2>
      <EstudarReact bom/>

      <h2>False</h2>
      <EstudarReact/>
    </div>
  );
}

function EstudarReact({ bom }) {
  const resposta = bom === true ? 'Bom' : 'Ruim';
  return (
    <div>Estudar React é muito {resposta}</div>
  );
}
```

True

Estudar React é muito Bom

False

Estudar React é muito Ruim

EXERCÍCIO

Crie um componente de **soma** que recebe
duas propriedades (números),
e que retorna uma frase: "***A soma dos números é*** __"

EXERCÍCIO

Crie um componente de **botão** com as props
corDeFundo (cor de fundo do botão) e texto (texto do botão).

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40

CHILDREN

Children

Vimos que quando usamos JSX para criar nossos componentes,
o React executa a seguinte **função** internamente:

```
React.createElement(component, props, ...children)
```

O que é children?

Children em português significa ***filhos***.
É uma propriedade especial do React que contém
qualquer elemento filho definido no componente.

Children na prática

Relembrando nosso componente<CapsLock />, ao invés de enviarmos um *prop textoEnviado*, podemos enviar o texto como *elemento filho*.

```
function App() {
  return (
    <div>
      <h1>Meu primeiro componente com props</h1>
      <CapsLock>Texto que ficará com letras maiúsculas</CapsLock>
    </div>
  );
}

function CapsLock(props) {
  const texto = props.children;
  const textoEmLetrasMaiusculas = texto.toUpperCase();
  return <div>{textoEmLetrasMaiusculas}</div>;
}
```

Children

Tudo pode ser um **componente filho**: parágrafo, texto, div, botão...

```
function App() {
  return (
    <div className="container">
      <CartaoDePresente titulo="Parabéns Arthur">
        <CapsLock />
        <div>
          <p>Seu aniversário é muito importante pra mim.</p>
        </div>
        <p>Eu gosto de te chamar de Arthur.</p>
        <button>Clique aqui para ganhar seu presente</button>
      </CartaoDePresente>;
    </div>
  );
}
```

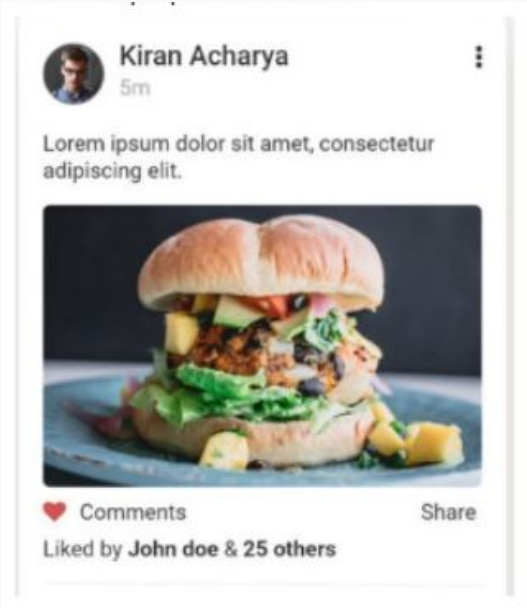
```
export default function CartaoDePresente({ titulo, children}){
  return (
    <div>
      <h3>{titulo}</h3>
      {children}
    </div>
  )
}
```

EXERCÍCIO

Crie um componente que espera receber um filho.
Feito isso, envie como filho o resultado de uma soma.

Exercício

Identifique quantas e quais são as props do componente abaixo.
Logo após, crie um componente **Post** passando todas as props identificadas.



Material complementar

- Como personalizar os componentes React com props | DigitalOcean
- Components and Props – React

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>