React: PropTypes e valores padrão

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- PropTypes
- Valores padrão: destructuring e defaultProps

PROPTYPES

PropTypes: por que usar?

Quando um componente recebe propriedades (props), espera-se que **cada propriedade seja enviada com o "tipo" correto**.

Por exemplo, o componente Soma espera receber propriedades com o formato de número.

```
const Soma = ({ numero1, numero2}) => {
  const resultado = numero1 + numero2;
  return <p>A soma dos números é {resultado}</p>
}

export default Soma;
```

Se as propriedades forem enviadas num formato não esperado, o resultado será incorreto.

```
{/* Envio de props: maneira esperada */}
<Soma numero1={10} numero2={20} />

{/* Envio de props: maneira não esperada */}
<Soma numero1="10" numero2="20" />
```

PropTypes: instalação da biblioteca

Para lidar com esse problema usamos a **biblioteca prop-types**, que nos permitirá controlar o tipo de propriedade que nosso componente irá receber.

Para instalar a biblioteca no projeto:

```
npm install prop-types
```

PropTypes

Para usar iremos importar para o nosso componente o **objeto PropTypes**, que conterá as *regras de tipagem* das propriedades.

```
import PropTypes from 'prop-types';
```

E adicionamos essas regras em outro **objeto propTypes**, que será uma *propriedade especial do nosso componente* (do próprio React)

```
Soma.propTypes = {}
```

As **chaves** desse objeto são as props do nosso componente.

```
Soma.propTypes = {
  numero1: PropTypes.number,
  numero2: PropTypes.number,
}
```

PropTypes

Por exemplo, o nosso componente Soma ficaria com a seguinte estrutura:

```
import PropTypes from 'prop-types';

export default function Soma({ numero1, numero2}) {
  const resultado = numero1 + numero2;
  return <p>A soma dos números é {resultado}</p>
}

Soma.propTypes = {
  numero1: PropTypes.number,
  numero2: PropTypes.number,
}
```

PropTypes

Exemplos de validações:

- PropsTypes.**string**
- PropTypes.**number**
- PropTypes.**bool**
- PropTypes.**func**
- PropsTypes.**object**
- PropsTypes.**shape**({
  color: PropTypes.string
  fontSize: PropsTypes.number
  })
- PropTypes.**array**
- PropsTypes.**oneOf**([“maçã”, “pera”])

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

PropTypes

Se tentarmos passar props com **tipagem diferente** do estabelecido...

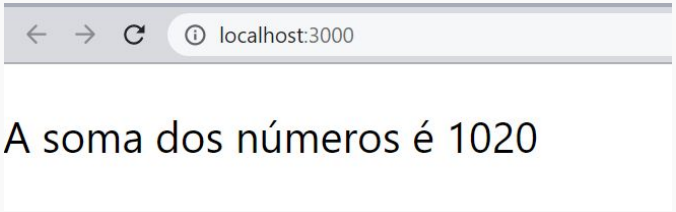
```
{/* Envio de props: maneira não esperada */}
<Soma numero1="10" numero2="20" />
```

receberemos um **warn** no console

```
▶Warning: Failed prop type: Invalid prop `numero1` of type `string` supplied to `Soma`, expected `number`.
    at Soma (http://localhost:3000/static/js/bundle.js:112:5)
    at App
```

react-jsx-dev-runtime.development.js:87

Mas a aplicação, apesar de incorreta, **continuará funcionando**.



PropTypes

E se **não enviarmos alguma propriedade**,
nenhum alerta será informado no console

```
{/* Não enviada uma prop (numero2) */}
<Soma numero1={10} />
```

Apenas afetando a aplicação.

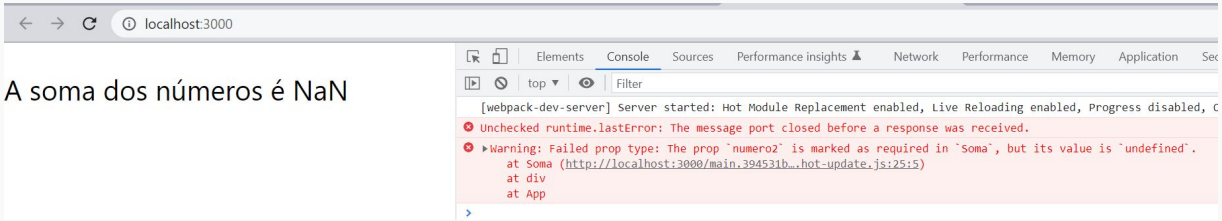
OBS: Isso poderá **dificultar muito a manutenção do código**,
pois teremos que "adivinhar" de onde vem o erro, demandando mais tempo e esforço.

PropTypes

Podemos tornar **obrigatório** que seja enviado o tipo especificado de prop utilizando **.isRequired**

```
Soma.propTypes = {
  numero1: PropTypes.number,
  numero2: PropTypes.number.isRequired,
}
```

e assim receberemos um **warn** no console, além dele deixar de funcionar corretamente.



O que **facilitará a manutenção**.

VALORES PADRÃO

Valores padrão

É possível atribuir **valores padrões** para as props dos componentes.

Isso “protege” nossa aplicação, evitando que erros possam acontecer caso alguma prop obrigatória não seja passada.

Há duas formas de atribuir valores padrão:

- destructuring
- defaultProps

Valores padrão: destructuring

Como vimos anteriormente, é possível usar **destructuring** dentro dos parênteses da função que retorna nosso componente.

Para usar valores padrão para cada props, basta **atribuir o valor** desejado quando declaramos cada propriedade.

```
export default function Soma({ numero1 = 0, numero2 = 0}) {
  const resultado = numero1 + numero2;
  return <p>A soma dos números é {resultado}</p>
}
```

Valores padrão: defaultProps

Outra maneira de fazer isso é utilizar o **objeto defaultProps**,
uma propriedade especial do React presente em qualquer componente.

```
Soma.defaultProps = {}
```

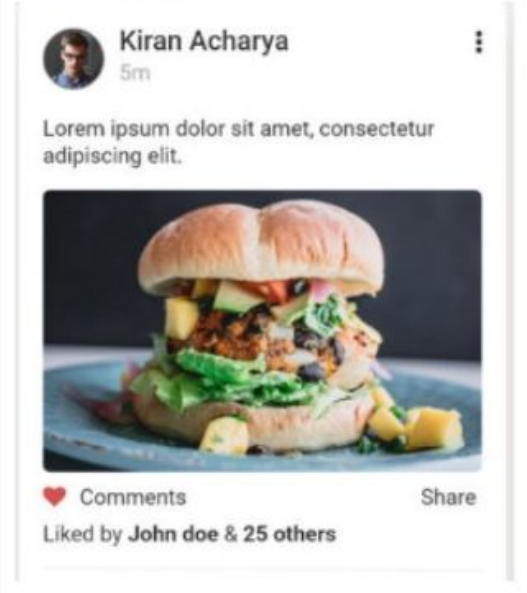
Cada chave deste objeto corresponde a uma prop do componente.

```
export default function Soma({ numero1, numero2}) {
  const resultado = numero1 + numero2;
  return <p>A soma dos números é {resultado}</p>
}

Soma.defaultProps = {
  numero1: 0,
  numero2: 0
}
```

Exercício

Utilizando componente Post criado na aula anterior, crie outros **componentes** recebidos por ele, controlando a **tipagem das props** de cada um e atribuindo **valores padrão** quando necessário



Material complementar

- Checagem de tipos (Typechecking) com PropTypes – React
- Typechecking With PropTypes – React
- Destructuring assignment

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>