Como usar PropTypes no React

Caso hipotético

- Imagine o seguinte: você acabou de ser contratado, está ansioso para aplicar tudo o que aprendeu, pronto para codar, e então... você é atribuído a um projeto legado. E não apenas isso, mas o projeto está escrito em JavaScript puro, sem TypeScript à vista!

ropTypes

- PropTypes são uma boa defesa de primeira linha quando se trata de depurar seus aplicativos.
- PropTypes são simplesmente um mecanismo que garante que o valor passado seja do tipo de dados correto. Isso garante que não recebamos um erro no final do nosso aplicativo pelo console, o que pode não ser fácil de resolver.
- Não recomendo usá-los em aplicativos pequenas, como projetos de pratica, mas isso depende totalmente de você. Para projetos maiores, como para um cliente, muitas vezes é uma escolha sábia e uma boa prática utilizá-los.
- Existem muitos tipos diferentes de PropTypes e todos eles têm suas classes ES6 exclusivas que podemos usar. Discutiremos cada tipo neste artigo.

O que são *props*?

Props é uma forma reduzida de "propriedades" em React.

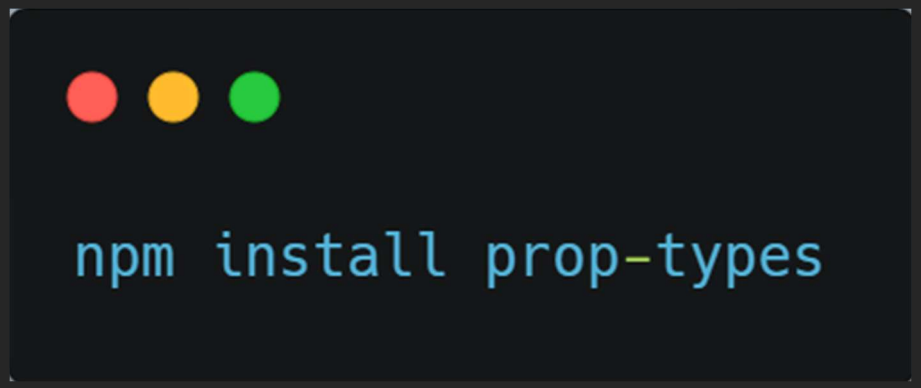
São usados para passar dados e funções entre o componente pai e o filho.

As props são imutáveis, significando que os componentes filhos não podem modificar os valores das props recebidas.

Contribuem para o fluxo de dados unidirecional do React, tornando a passagem de dados previsível e consistente.

As props tornam os componentes reutilizáveis e dinâmicos, adaptando-se aos dados fornecidos.

São principalmente passadas do componente pai para o filho, embora a passagem inversa seja tecnicamente possível, não é recomendada para manter a clareza e a previsibilidade do fluxo de dados.



Como usar PropTypes

- Antes do lançamento do React 15.5.0, PropTypes estavam disponíveis no pacote React, mas agora temos que adicionar a biblioteca prop-types em nosso projeto.
- Podemos fazer isso executando o seguinte comando em nosso terminal:

Como usar o PropTypes

- Podemos usar PropTypes para validar quaisquer dados que recebemos de props. Mas antes de usá-lo teremos que importá-lo como sempre em nosso projeto:

```jsx
import React from 'react';
import { PropTypes } from "prop-types";

const Count = (props) => {
  return (
    <>
      // sua lógica aqui
    </>
  )
};

Count.propTypes = {
// key é o nome da prop e
// value é o PropType
}
export default Count;
```

Como usar PropTypes

- Eles são frequentemente usados após o término do componente e começam com o nome do componente, conforme mostrado:

Estrutura de um PropTypes

- PropTypes também são objetos com uma chave e um par de valores onde a ‘chave’ é o nome da propriedade enquanto o valor representa o tipo ou classe pela qual eles são definidos.

```
Count.propTypes = {
// Put props here
}
```

Tipos básicos de PropTypes

A maneira mais básica de verificar o tipo de uma propriedade é verificando se ela se enquadra na categoria de tipos primitivos em JavaScript, como booleano, string, objeto e assim por diante.

Type	Class	Example
String	PropType.string	“helllo”
Object	PropType.object	{name: “Dev em Dobro”}
Number	PropType.number	10
Boolean	PropType.bool	true/false
Function	PropType.func	const say = {console.log(“O mundo!”)}
Symbol	PropType.symbol	Symbol(“m”)

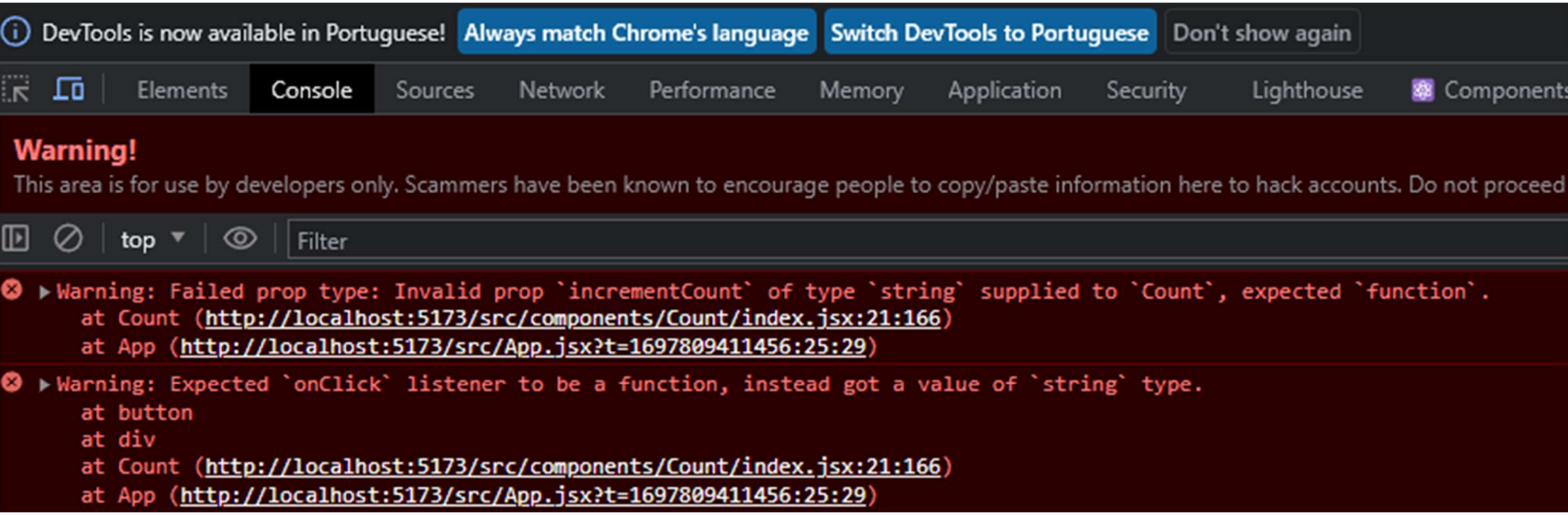
Tipos básicos de PropTypes

Ao lado está um exemplo que nos mostra como usar esses PropTypes para verificação de tipo em nosso aplicativo.

```
Count.propTypes = {
  name: PropTypes.string,
  age: PropTypes.number,
  address: PropTypes.object,
  friends: PropTypes.array,
};
```

Como erros aparecem no dev tools

- Se qualquer valor diferente deste tiver sido usado para as mesmas propriedades de um valor, será exibido um erro no console como este:



Tipos básicos de PropTypes

Podemos encadear qualquer um dos itens acima com isRequired para garantir que um aviso seja mostrado se o suporte não for fornecido.

```
Count.propTypes = {
  basicObject: PropTypes.object,
  numbers: PropTypes.objectOf(PropTypes.numbers),
  messages: PropTypes.instanceOf(Message),
  contactList: PropTypes.shape({
    name: PropTypes.string.isRequired,
    phone: PropTypes.string.isRequired,
  }),
};
```

Tipos básicos de PropTypes

Array Types

```
Count.propTypes = {
  counts: PropTypes.array,
  users: PropTypes.arrayOf(PropTypes.obje
  alarmColor: PropTypes.oneOf(["red", "bl
  description: PropTypes.oneOfType([
    PropTypes.string,
    PropTypes.instanceOf(Title),
  ]),
};
```

Type	Class	Exam
Array	PropTypes.array	[]
Array of numbers	PropTypes.arrayOf([type])	[1,2,
Array of string	PropTypes.oneOf([arr])	[“Red”, “
Array of objects	PropTypes.oneOfType([types])	PropTypes PropTypes.insta

Tipos básicos de PropTypes

Object Types

```
Count.propTypes = {
  basicObject: PropTypes.object,
  numbers: PropTypes.objectOf(PropTypes.nu
  messages: PropTypes.instanceOf(Message),
  contactList: PropTypes.shape({
    name: PropTypes.string,
    phone: PropTypes.string,
  }),
};
```

Type	Class	Exa
Object	PropTypes.object	{name
Number Object	PropTypes.objectOf()	{ag
Object Shape	PropTypes.shape()	{n PropTyp ph PropTyp
Instance	PropTypes.objectOf()	New m

Verificação avançada de tipo

```
Count.propTypes = {
  displayElement: PropTypes.ele
};
```

Como verificar um componente React

- Se você quiser apenas verificar se um prop é um componente React, você pode usar PropTypes.element. Isso é útil para garantir que um componente tenha apenas um componente filho.

Type	Class	Ex
Element	PropTypes.element	<T

Verificação avançada de tipo

- Como verificar o nome de um componente React
  - Podemos verificar se o prop é o nome de um componente React usando PropTypes.elementType.

Tipos personalizados

ype	Class	Example
stom	function(props, propName, componentName) {}	“Olá, Mundo!”
stom rray	PropTypes.arrayOf(function(props, propName, componentName) {})	[“Olá, Mundo!”]

- Também podemos ter um validador personalizado ou verificação de tipo para props, mas isso precisamos passar um objeto de erro se a validação falhar.
- Você pode usar isso para arrays e objetos, mas o objeto de erro será chamado para cada chave no array o objeto. Os dois primeiros argumento do validador são o array ou objeto e si e a chave do item atual.

Tipos personalizados

```
Count.propTypes = { // array function
  customArrayProp: PropTypes.arrayOf(function (
    propValue,
    key,
    componentName,
    location,
    propFullName
  ) {
    if (!/matchme/.test(propValue[key])) {
      return new Error(
        "Invalid prop `" +
          propFullName +
          "` supplied to" +
          " `" +
          componentName +
          "`. Validation failed."
      );
    }
  }),
};
```

```
Count.propTypes = {  // normal functionn
  customProp: function (props, propName, componentNa
    if (!/matchme/.test(props[propName])) {
      return new Error(
        "Invalid prop `" +
          propName +
          "` supplied to" +
          " `" +
          componentName +
          "`. Validation failed."
      );
    }
  },
};
```

Default PropTypes

- Às vezes, queremos poder definir um valor padrão para uma propriedade. Por exemplo, nosso componente pai pode não exigir a passagem de um título. Mas ainda queremos que um título seja renderizado.
- Em casos como este, podemos definir um valor padrão para nosso título que será renderizado automaticamente se o título não tiver sido passado como suporte de nosso componente pai.

Referências

- FreeCodeCamp: https://www.freecodecamp.org/news/how-to-use-proptypes-in-react/
- Documentação oficial (legada): https://legacy.reactjs.org/docs/typechecking-with-proptypes.html
- O que são props: https://tekolio.com/what-are-props-in-react-and-how-to-use-them/

HORA DE COZINHAR!