Google olabi

<React>

React

Criada em cima do conceito de componentização para reutilização de códigos, React é uma biblioteca JavaScript utilizada para facilitar a construção de interfaces de usuário (*user interface*, ou UI), um conjunto de códigos para desenvolvimento de aplicações que evita repetição deles. Surgiu de uma necessidade do Facebook em escalar aplicações diminuindo a necessidade de manutenção do código. O engenheiro Jordan Walke realizou um teste com tecnologia XHP (antes escrita em PHP) e levou-a para o navegador utilizando Javascript.

Componentes em React

Por ser uma biblioteca *front-end*, um dos objetivos da React é facilitar a conexão entre diferentes partes de uma página. Seu funcionamento acontece através de componentes - conjuntos isolados de lógica (Javascript), visualização (JSX/HTML) e estilização (CSS).

Considere a situação abaixo:

Na figura acima, temos um projeto escrito em React detalhando as informações de cada usuário listado. Muitos desses elementos seguem um padrão: os botões possuem o mesmo tamanho, cor e texto. Os espaços que contém os botões possuem os mesmos padrões de tamanho e cor, com poucas diferenças.

Esses são os componentes, que dividem a interface em pequenas partes de forma a permitir que sejam utilizadas várias vezes na aplicação - o que faz da React uma tecnologia flexível para solução de problemas e construção de interfaces reutilizáveis, já que cada componente pode ser manipulado de maneira distinta.

{ JSX

React utiliza uma extensão alternativa ao JavaScript chamada JSX, que descreve componentes e pode ser obtida por meio da união de HTML com JavaScript. A finalidade da JSX é ser transposta para JavaScript da maneira mais simples possível. Ela não é imprescindível para o uso de React, mas é amplamente utilizada.

Em React, a renderização é acoplada a outras lógicas de interface: ao invés de separar tecnologias artificialmente - colocando marcação (html) e lógica em arquivos distintos-, ela o faz por meio de conceitos com unidades pouco acopladas, chamadas “componentes”, que contém ambos.

{ React App

Quando criamos um projeto, alguns arquivos aparecem como padrão: App.js, App.test.js e App.css. Esses arquivos contém a lógica do componente App, seu teste e estilo, respectivamente. O arquivo App.js estende a classe *Component* (nativa de React), que faz da classe em questão um componente.

Render

Quando criamos um projeto, alguns arquivos aparecem como padrão: App.js, App.test.js e App.css. Esses arquivos contém a lógica do componente App, seu teste e estilo, respectivamente. O arquivo App.js estende a classe *Component* (nativa de React), que faz da classe em questão um componente.

Props

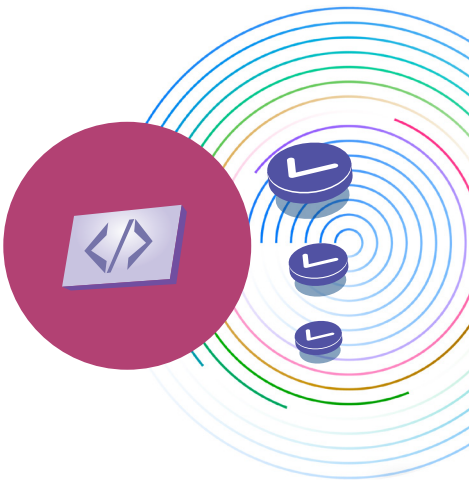
É a principal forma de passar informação para um componente. *Props* permite a passagem de vários dados e valores.

Controle de estados

Estado é um atributo de componente similar à *prop*, porém privado e controlado pelo próprio. Um estado pode conter informações de todos os tipos, assim como nas *props*, mas elas só podem ser controladas pelo próprio componente.

React router (roteamento)

Para chamar diferentes telas, utilizamos roteamento. Em React, ele é controlado pelo pacote *react-router*, que fornece componentes não visuais para controle das rotas do sistema.



{ Imagens em React

É importante separar a responsabilidade de cada arquivo em pastas diferentes para o exercício de boas práticas. Se quisermos usar imagens em React, precisamos seguir os passos abaixo:

• salvar a imagem em uma pasta *assets;*

• importar a imagem e guardá-la em uma variável (import Image from '../assets/minha-imagem.png);

• usar a imagem no componente ();

• dentro do src, usar a variável atribuída no *import*.

É obrigatório o uso do atributo alt informando um texto alternativo coerente, caso a imagem não carregue.

{ CSS em React

Em React, existem facilidades para criação de estilos de componentes e páginas. Bibliotecas como *styled components, saas, material-ui, bootstrap e ant design*, dentre outras.

Para CCS em React, basta seguir os passos abaixo:

• criar um arquivo com extensão .css;

• importar o arquivo (import from '../styles/style.css);

• para estilo, você pode atribuir *className* e id, ou mesmo uma tag como seletor css ();

• Dentro do arquivo CSS, usar os seletores e propriedades.

É importante atentar para não sobrescrever estilos de forma inesperada: Para zerar seu estilo de forma global na aplicação, você deve importá-lo nas camadas acima do componente, como *index* ou *app*.

<Exercícios>

1. Crie uma aplicação REACT WEB utilizando npx e depois, um componente "Título". Este componente deverá renderizar o corpo da *tag* <Título> dentro de um <h1>;

2. Crie um arquivo “Titulo.js” dentro da pasta src;

3. Crie uma aplicação REACT WEB utilizando npx, seguida dos componentes “Lista” e “Item”: o componente “Lista” precisa renderizar o conteúdo do corpo (*props.children*) dentro de uma <ul>; o componente “Item” precisa renderizar o conteúdo do corpo (*props.children*) dentro de uma <li>.

<Referências>

React Org: react
Simara Conceição: github

Professor: Júlio César | Monitora: Caroline Brum