INSTITUTO FEDERAL
Catarinense
Campus Blumenau

REACT JS

Base para aprender React JS

Jessica Mayumi Schuhmacher Kogake

101 Info | 2022

Web Design

REACT JS

Há algum tempo já vem sendo criados programas, plugins e ferramentas diferentes para facilitar a criação de códigos, tanto para desenvolvedores back-end quanto desenvolvedores front-end.

Dentre elas existem as bibliotecas, que na área da programação, são subprogramas, num geral públicos, onde dentro deles existem códigos que auxiliam na rapidez e em uma menor taxa de erros dos desenvolvedores de programas.

O React é uma Biblioteca JavaScript criada pelo Facebook, para facilitar a criação e UI (User Interface ou interfaces de usuários).

Em 2011, quando o Facebook percebeu que havia uma grande dificuldade de manter o Live Feed constante, isto é, manter diversas atividades completamente diferentes entre si sempre atualizadas (chats, notificações, notícias, ...), eles criaram o React para facilitar e otimizar a união do HTML e CSS com a parte lógica do código (JavaScript). Mas em 2013, o React se tornou aberto à comunidade, e em pouco tempo conseguindo ultrapassar outras ferramentas como jQuery ou Angular, que tinham o mesmo propósito de unir as 3 principais tecnologias já utilizadas pelos desenvolvedores front-end (HTML, CSS e JavaScript).

Atualmente diversas empresas famosas já utilizam dessa tecnologia dominante no mercado front-end como Tesla, Instagram, Discord, Netflix.

O React trabalha de forma declarativa, tendo como sua principal característica a praticidade de poder criar layouts complexos, apenas unindo pequenos fragmentos de códigos, conhecidos também como componentes.

- Componentes

Os componentes são itens individuais que, podem ou não conter outros itens. Eles são como peças de quebra-cabeça que, quando combinados com outras peças, conseguem gerar algo mais complexo, ou de maneira mais bruta, "componentes são como funções JavaScript".

- JSX (JavaScript XML)

O JSX é uma extensão criada pela equipe de desenvolvimento React para simplificar o desenvolvimento de novos códigos, pois ele consegue unir a parte lógica do JavaScript com o HTML. Porém, os navegadores não conseguem interpretá-lo, sendo necessário por meio de transpiladores. O transpilador mais conhecido atualmente é o Babel.

- Props (propriedades)

Dentro desses componentes existem os props (propriedades) que recebem a função de retornar algo, podendo ser números ou strings.

- Props são valores estáticos.

- States (estados)

Já os states (estados) podem ser criados diretamente dentro dos componentes e, eles têm funções semelhantes a de uma variável. Armazenam um valor (número ou string) que pode ser passado adiante através das props.

- States são valores dinâmicos.

```
// Exemplo do State e Props na prática

class Filmes extends React.Component {
  constructor(props) {
    super(props)

    this.state = {
    nome: 'Homem Aranha'
   };

  }

  render () {
    return (
      <p>Nome do Filme: { this.state.nome } </ p>
  )

}
```

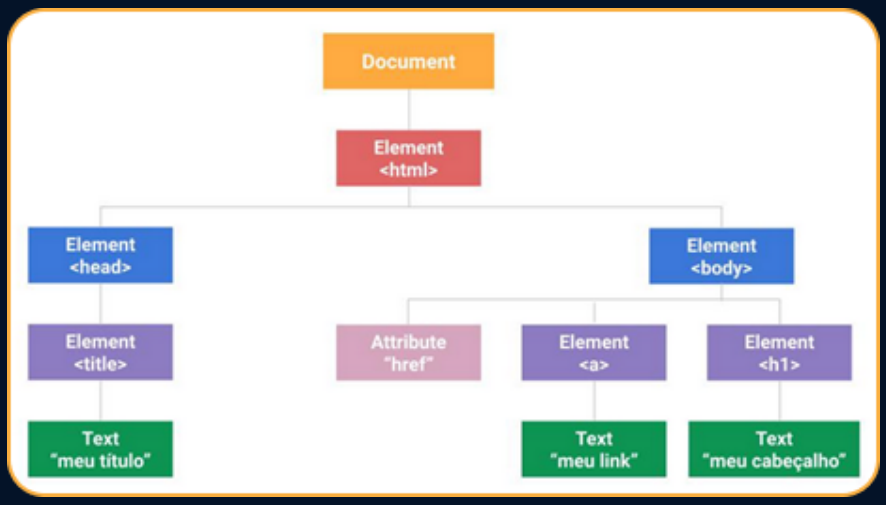
https://sujeitoprogramador.com/qual-a-diferenca-entre-props-e-states-em-react/

DOM

Criada em 1994, por Tim Berners-Lee, o mesmo criador da World Wide Web Consortium, DOM significa Document Object Model ou Modelo de Documento por Objetos. Ele é uma interface padronizada (Padrões da W3C) já embutida nos navegadores que tem o poder de gerar estruturas que representam, em uma espécie de "árvore", o documento HTML ou XML da sua página. Ele geralmente é utilizado por quem deseja construir ou apenas atualizar uma UI avançada.

O DOM permite que manipulações sejam feitas na página de maneira muito fácil e prática, pois com ele, não só o desenvolvedor, mas como também o usuário pode alterar o layout da página apenas movendo, arrastando ou removendo elementos.

Exemplo de como a "árvore DOM" geralmente é estruturada



https://www.hostinger.com.br/tutoriais/dom-o-que-e

VIRTUAL DOM

O Virtual DOM ou VDOM é uma técnica utilizada pelo React para uma melhor otimização do site ou app.

Basicamente ele guarda a estrutura UI em memória e sincroniza em tempo real com o DOM "real" através de bibliotecas em um processo conhecido como Reconciliação. O Virtual DOM sempre tenta ver o que pode ser reutilizado, para que sejam necessários o menor número de passos possíveis, para mais uma vez, otimizar cada vez mais a transmissão de dados para a tela.

Além disso, o Virtual DOM permite a "tradução" de estruturas, ou seja, é possível criar uma estrutura que será utilizada tanto por VR(Virtual Reality) quanto por usuários de celular. Gerando assim, um "leque" de infinitas possibilidades para a adição de novas plataformas ao React.

HOOKS

Os Hooks é uma função recentemente adicionada no React. Até então, na versão 16.7 para baixo, algumas funções só poderiam ser acessadas por meio de classes. Mas nem sempre conseguiam extrair o máximo do React, foi aí que lançaram a versão 16.8, que, juntamente dela vieram os hooks. Esses Hooks vieram com a intenção de ajudar a reutilização e organização da lógica dentro dos componentes. Então com eles é possível simplificarmos o código, tornando-o mais claro.

Exemplo de como era antes de criarem os hook:

```
// Exemplo antes da invenção dos Hooks, onde se usavam propriedades e classes ao invés de States.

class Example extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0
    };
  }

  render() {
    return (
      <div>
        <p>Você clicou {this.state.count} vezes.</p>
        <button onClick={() => this.setState({ count: this.state.count + 1 })}>
          Clique Aqui
        </button>
      </div>
    );
  }
}
```

Exemplo de depois de criarem os hooks:

```
// Exemplos depois da criação dos Hooks, neste caso o useState

import React, { useState } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Você clicou {count} vezes.</p>
      <button onClick={() => setCount(count + 1)}>
        Clique Aqui
      </button>
    </div>
  );
}
```

- Hook useState

O hook useState é utilizado para dar um estado para os componentes React.

Exemplos de useState

```jsx
// useState

import { useState } from "react";
import ReactDOM from "react-dom/client";
// Primeiro é importado o Hook useState do React

function FavoriteColor() {
  const [color, setColor] = useState("vermelho");
// Se a constante acima fosse uma porta, o 'color' seria a porta e o 'setColor' a maçaneta.
// Então porta (color) seria a maneira na qual chamamos a constante e, maçaneta (setColor)
// a maneira com que atualizamos/mechemos na constante.

  return <h1>Minha cor favorita é {color}!</h1> // Exemplo chamando a constante.
}

//const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<FavoriteColor />);
// Aqui apenas estamos declarando que quando utilizarmos algo com o id='root' no HTML,
// será rendenizado/retornado o função FavoriteColor.
```

```jsx
// useState, exemplo atualizando o setColor

import { useState } from "react";
import ReactDOM from "react-dom/client";

function FavoriteColor() {
  const [color, setColor] = useState("vermelho");

  return (
    <>
      <h1>Minha cor favorita é {color}!</h1>
      <button
        type="button"
        onClick={() => setColor("azul")}
      >Azul</button>
    </>
  )
}

// No h1 está chamada a constante através do 'color'
// Já na tag button está falando que, quando o botão for clicado, a cor
// se tronará azul, pois chamamos a constante para atualizar seu valor com
// o setColor.

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<FavoriteColor />);
```

Exemplo de useState criando vários Hooks

```jsx
// useState, criando mais de um hook dentro de uma mesma função

import { useState } from "react";
import ReactDOM from "react-dom/client";

function Car() {
  const [marca, setMarca] = useState("Ford"); // Hook #1
  const [modelo, setModelo] = useState("Mustang"); // Hook #2
  const [ano, setAno] = useState("1964"); // Hook #3
  const [cor, setCor] = useState("vermelho"); // Hook #4

  return (
    <>
      <h1>Meu {marca}</h1>
      <p>
        Ele é {cor} {modelo} do ano de {ano}.
      </p>
    </>
  )
}

// Neste exemplo criamos diversos hook que podem ter seus valores
// alterados de maneira separada.

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

Exemplo de useState atribuindo vários valores a um mesmo Hook

```jsx
// useState, criando apenas um hook com vários valores

import { useState } from "react";
import ReactDOM from "react-dom/client";

function Car() {
  const [car, setCar] = useState({
    marca: "Ford", // valor #1
    modelo: "Mustang", // valor #2
    ano: "1964", // valor #3
    cor: "vermelho" //valor #4
  });

  return (
    <>
      <h1>Meu {car.marca}</h1>
      <p>
        Ele é {car.cor} {car.modelo} do ano de {car.ano}.
      </p>
    </>
  )
}

// Já neste exemplo criamos uma função que armazena apenas um hook, mas com 4
// valores diferentes entre si.
// Para chamarmos ele é preciso colocar o nome que utilizaríamos para chamá-lo,
// neste caso o 'car', em seguida colocamos o '.' ( ponto ) e o nome do objeto.

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

Exemplo de useState utilizando o previousState

O previousState é utilizado quando queremos alterar um valor de um objeto quando temos vários valores dentro de um mesmo Hook.

```jsx
// useState, alterando apenas um valor do hook

import { useState } from "react";
import ReactDOM from "react-dom/client";

function Car() {
  const [car, setCar] = useState({
    marca: "Ford",
    modelo: "Mustang",
    ano: "1964",
    cor: "vermelho"
  });

// Queremos mudar apenas a cor, mas se apenas colocar-mos
// {cor: "azul"} estaríamos tirando todo os outros valores
// do hook e deixando apenas o {cor: "azul"} dentro de toda
// a constante. Para isso não ocorrer, utilizamos o 'previousState',
// na qual ele carrega todos os outros valores da constante, em
// seguida colocamos a ',' ( vírgula ) e o objeto que queremos
// alterar.

  const updateColor = () => {
    setCar(previousState => {
      return { ...previousState, cor: "azul" }
    });
  }

  return (
    <>
      <h1>Meu {car.marca}</h1>
      <p>
        Ele é {car.cor} {car.modelo} do ano de {car.ano}.
      </p>
      <button
        type="button"
        onClick={updateColor}
      >Azul</button>
    </>
  )
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

Todos estes exemplos foram tirados da página de useStates do site da w3school, mas explicados de uma maneira mais simplificada para uma melhor compreensão.

https://www.w3schools.com/react/react_usestate.asp

CONCLUSÃO

Após tudo isso concluímos que o React JS consegue contribuir muito para o desenvolvimento dos códigos, facilitando a criação, o entendimento e futuras manutenções que podem ser necessárias.
Atualmente diversas empresas famosas utilizam desta tecnologia que, cada vez mais vem se tornando algo essencial para se aprender para quem pretende trabalhar na área.

LINKS DE REFERÊNCIA

https://www.w3schools.com/react/react_hooks.asp

https://www.alura.com.br/artigos/react-hooks

https://reactjs.org/docs/hooks-state.html

https://pt-br.reactjs.org/docs/hooks-intro.html

https://pt-br.reactjs.org/

https://pedro-mello.netlify.app/react-part-2/

https://pt-br.reactjs.org/docs/faq-internals.html

https://www.google.com/search?client=firefox-b-d&q=Virtual+Dom

https://www.hostgator.com.br/blog/o-que-e-dom-na-programacao/

https://www.hostinger.com.br/tutoriais/dom-o-que-e

https://pt-br.reactjs.org/docs/faq-state.html

https://imasters.com.br/front-end/entendendo-estado-de-componentes-com-react-na-pratica

https://pt-br.reactjs.org/docs/hooks-state.html

https://tipscode.com.br/state-e-props-no-react-entenda-de-uma-vez-as-diferencas

https://sujeitoprogramador.com/qual-a-diferenca-entre-props-e-states-em-react/

https://pboverflow.programadoresbrasil.com.br/t/qual-a-diferenca-entre-state-e-props-no-react/1660

https://lucasbaradel.medium.com/diferen%C3%A7a-entre-props-e-state-no-react-js-50b3db57a8e3

https://www.horadecodar.com.br/2020/08/14/react-qual-a-diferenca-entre-props-e-state/

https://pt-br.reactjs.org/docs/dom-elements.html

https://blog.cronapp.io/framework-react/

https://www.idtech.com/blog/what-are-libraries-in-coding

https://www.treinaweb.com.br/blog/conheca-o-react-biblioteca-para-desenvolvimento-web

https://www.w3schools.com/react/react_render.asp

https://celke.com.br/artigo/o-que-e-componente-no-react

https://pt.wikipedia.org/wiki/React_(JavaScript)

https://kenzie.com.br/blog/react/