Guia Rápido de React JS (React 18)

Índice

O que é React?

React é uma biblioteca JavaScript para construção de interfaces de usuário, especialmente para aplicações de página única (SPAs). Ele facilita a criação de componentes reutilizáveis e a gerência eficiente do DOM.

Configuração de Projeto

Para iniciar um novo projeto React, você pode usar `create-react-app`:

```
npx create-react-app meu-projeto
cd meu-projeto
npm start
```

Componentes

Os componentes são a base do React. Eles podem ser criados como **componentes de função** ou **componentes de classe** (embora os componentes funcionais com hooks sejam mais usados atualmente).

Exemplo de Componente de Função

```
function MeuComponente() {
  return <h1>Olá, mundo!</h1>;
}

export default MeuComponente;
```

Exemplo de Componente de Classe

```
import React, { Component } from 'react';

class MeuComponente extends Component {
  render() {
    return <h1>Olá, mundo!</h1>;
  }
}

export default MeuComponente;
```

JSX

JSX é uma extensão de sintaxe para JavaScript, que permite escrever HTML dentro de código JavaScript.

```jsx
const elemento = <h1>Olá, JSX!</h1>;
```

JSX se parece com HTML, mas com algumas diferenças:

- **class** se torna **className**
- Elementos devem ser encerrados corretamente (por exemplo, `<img />`)

State e Props

State

O **state** é um objeto que determina como o componente deve se comportar. Ele pode ser alterado dinamicamente.

```jsx
port { useState } from 'react';

function Contador() {
  const [contador, setContador] = useState(0);

  return (
    <div>
      <p>Você clicou {contador} vezes</p>
      <button onClick={() => setContador(contador + 1)}>
        Clique aqui
      </button>
    </div>
  );
}
```

Props

As **props** são valores passados para componentes para configurá-los externamente.

```jsx
function Saudacao(props) {
  return <h1>Olá, {props.nome}!</h1>;
```

```
}
```

Ciclo de Vida dos Componentes

Os componentes de classe têm métodos de ciclo de vida que você pode usar:

- `componentDidMount`: Invocado após o componente ser montado.
- `componentDidUpdate`: Chamado após uma atualização.
- `componentWillUnmount`: Chamado antes de o componente ser desmontado.

Nos componentes funcionais, você pode usar o `useEffect` para replicar esses comportamentos.

Hooks

Os hooks permitem que você use o state e outras funcionalidades do React sem escrever classes. Os principais hooks são:

`useState`

Permite adicionar state a um componente funcional.

```
const [state, setState] = useState(valorInicial);
```

`useEffect`

Permite lidar com efeitos colaterais, como chamadas de API, ou execução de código ao montar e desmontar componentes.

```
useEffect(() => {
  // Código a ser executado quando o componente monta
}, []); // Dependências
```

`useContext`

Permite acessar o contexto sem envolver os componentes intermediários com um Provider.

```jsx
const valor = useContext(MeuContexto);
```

`useRef`

Permite criar referências para elementos DOM ou armazenar valores persistentes entre renderizações.

```jsx
const minhaRef = useRef(null);
```

Context API

A **Context API** é usada para compartilhar dados entre componentes sem a necessidade de passar props manualmente por cada nível da árvore de componentes.

Exemplo:

```jsx
const MeuContexto = React.createContext();

function App() {
  return (
    <MeuContexto.Provider value={'Olá'}>
      <ComponenteFilho />
    </MeuContexto.Provider>
  );
}

function ComponenteFilho() {
  const valor = useContext(MeuContexto);
  return <h1>{valor}</h1>;
}
```

Renderização Condicional

Renderizar componentes com base em condições:

```jsx
function Saudacao({ isLoggedIn }) {
  return isLoggedIn ? <h1>Bem-vindo!</h1> : <h1>Por favor,
```

```
faça login.</h1>;
}
```

Renderização de Listas

Renderize listas em React usando o método `map`.

```
const itens = ['Item 1', 'Item 2', 'Item 3'];

return (
  <ul>
    {itens.map(item => (
      <li key={item}>{item}</li>
    ))}
  </ul>
);
```

React Router

React Router é usado para navegação em aplicações de página única.

Instalação:

```
npm install react-router-dom
```

Exemplo de uso:

```
import { BrowserRouter as Router, Route, Routes, Link } fro
m 'react-router-dom';

function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link>
        <Link to="/sobre">Sobre</Link>
      </nav>
      <Routes>
```

```
        <Route path="/" element={<Home />} />
        <Route path="/sobre" element={<Sobre />} />
      </Routes>
    </Router>
  );
}
```

Suspense e Lazy Loading

React 18 facilita a divisão de código e carregamento dinâmico com `React.lazy` e `Suspense`.

```
const Componente = React.lazy(() => import('./Component
e'));

function App() {
  return (
    <React.Suspense fallback={<div>Carregando...</div>}>
      <Componente />
    </React.Suspense>
  );
}
```

Concurrent Mode e Automatic Batching

Com React 18, o **Concurrent Mode** e o **Automatic Batching** melhoram a renderização, permitindo que várias atualizações de estado sejam agrupadas para reduzir re-renderizações.

```
function App() {
  const [count, setCount] = useState(0);
  const [text, setText] = useState('');

  function handleClick() {
    setCount(count + 1);
    setText('Atualizado!'); // Ambas as atualizações são ag
```

```
rupadas automaticamente
  }

  return (
    <div>
      <button onClick={handleClick}>Clique</button>
      <p>{count}</p>
      <p>{text}</p>
    </div>
  );
}
```

Referências Úteis

- Documentação Oficial do React
- React Router