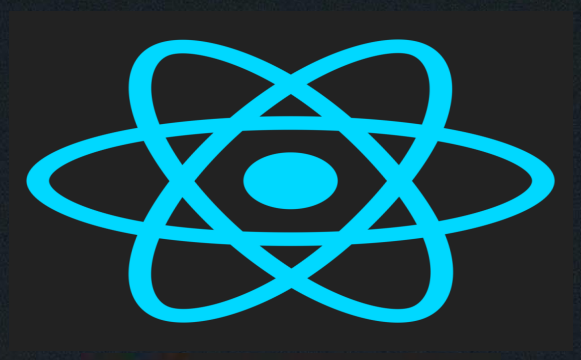
A Saga do Programador Destemido:

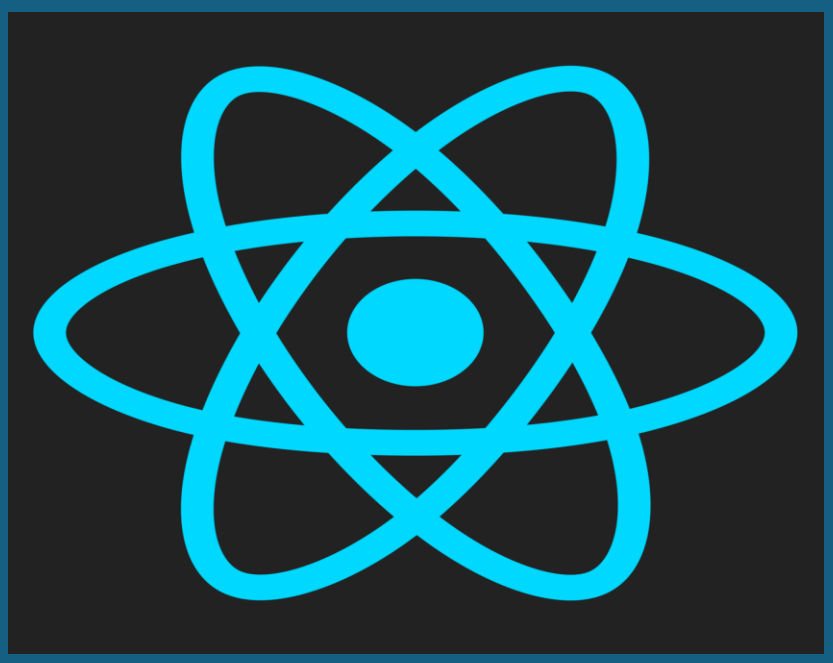
O Livro Proibido



Fernando F. Hipólito

Introdução

React é uma biblioteca JavaScript popular para construir interfaces de usuário interativas. Sua abordagem baseada em componentes facilita a criação de aplicativos escaláveis e reutilizáveis. Vamos explorar os principais componentes do React e entender como eles funcionam.



01

Componentes Funcionais e de Classe

Componentes Funcionais e de Classe

Em React, existem dois tipos principais de componentes: funcionais e de classe. Os componentes funcionais são simples funções JavaScript que retornam elementos React. Eles são mais simples e geralmente usados para componentes sem estado.

javascript Copy code

```javascript
// Exemplo de componente funcional
const Welcome = () => {
  return <h1>Olá, Mundo!</h1>;
};
```

Os componentes de classe são classes JavaScript que estendem React.Component. Eles têm um estado interno e podem ter métodos de ciclo de vida.

javascript Copy code

```javascript
// Exemplo de componente de classe
class Timer extends React.Component {
  constructor(props) {
    super(props);
    this.state = { seconds: 0 };
  }

  componentDidMount() {
    this.timerID = setInterval(
      () => this.tick(),
      1000
    );
  }

  componentWillUnmount() {
    clearInterval(this.timerID);
  }

  tick() {
    this.setState({
      seconds: this.state.seconds + 1
    });
  }

  render() {
    return (
      <div>
        Segundos: {this.state.seconds}
      </div>
    );
  }
}
```

02

Props: Propriedades dos Componentes

Props: Propriedades dos Componentes

As props são a principal maneira de passar dados entre componentes React. Elas são passadas para componentes como atributos e são acessíveis dentro do componente.

javascript Copy code

```javascript
// Exemplo de componente usando props
const Welcome = (props) => {
  return <h1>Olá, {props.name}!</h1>;
};

// Uso do componente
<Welcome name="João" />;
```

03

Estado: Gerenciando Dados Internos

Estado: Gerenciando Dados Internos

O estado é gerenciado internamente pelos componentes React. Quando o estado de um componente muda, o componente é renderizado novamente.

javascript Copy code

```javascript
// Exemplo de componente com estado
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  render() {
    return (
      <div>
        Contagem: {this.state.count}
        <button onClick={() => this.setState({ count: this.state.count + 1 })}>
          Incrementar
        </button>
      </div>
    );
  }
}
```

04

Ciclo de Vida dos Componentes

Ciclo de Vida dos Componentes

Os componentes React têm métodos de ciclo de vida que são chamados em diferentes estágios do ciclo de vida do componente.

javascript Copy code

```javascript
// Exemplo de ciclo de vida de um componente de classe
class ExampleComponent extends React.Component {
  componentDidMount() {
    // É chamado após o componente ser montado no DOM
  }

  componentDidUpdate(prevProps, prevState) {
    // É chamado após uma atualização do componente
  }

  componentWillUnmount() {
    // É chamado antes do componente ser desmontado e removido do DOM
  }

  render() {
    return <div>Exemplo de Componente</div>;
  }
}
```

Conclusão

Dominar os principais componentes do React é fundamental para se tornar um desenvolvedor React eficiente. Compreender como os componentes funcionam, como passar dados entre eles e como gerenciar estado e ciclo de vida é essencial para construir aplicativos React robustos e escaláveis. Experimente criar seus próprios componentes e explore ainda mais a poderosa biblioteca React!

Espero que este guia tenha sido útil para você começar sua jornada com React! Continue praticando e explorando para aprimorar suas habilidades de desenvolvimento web.

Continue codando, e que os elementos React estejam sempre a seu favor!