Téc em Desenvolvimento
de Sistemas Bilíngue

Desenvolver Código

React Native

Componentes

React Native

Componentes são elementos reutilizáveis que representam partes da interface do usuário. Eles podem ser comparados a funções em programação: você os define uma vez e pode utilizá-los em diversos lugares. No React Native, os componentes podem ser de dois tipos principais: funcionais e de classe.

Componentes Funcionais

React Native

Componentes funcionais são funções que retornam elementos React. Eles são simples e eficazes, especialmente para componentes que não possuem estado interno complexo.

React Native

Componente Funcional

```
//a função MeuComponente é um componente funcional
function MeuComponente() {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Este é um componente funcional</Text>
    </View>
  );
}
```

React Native

Componente Funcional (arrow function)

```
//a função MeuComponente é um componente funcional (agora em arrow function)
const MeuComponente = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Este é um componente funcional</Text>
    </View>
  );
};
```

Componentes de Classe

React Native

Em React, os componentes de classe são formas de definir partes reutilizáveis da sua interface de usuário usando a sintaxe de classes. Eles permitem criar interfaces dinâmicas e interativas.

React Native

Um componente de classe é uma classe do JavaScript que estende React.Component. Isso permite que ele tenha funcionalidades especiais, como gerenciar estado interno e responder a eventos. O que isso significa? Que ele herda funcionalidades específicas de React.

React Native

```
class MeuComponente extends Component {
```

React Native

Componentes de classe podem ter um estado interno (state) que armazena dados dinâmicos. Esse estado pode ser atualizado ao longo do tempo, fazendo com que o componente se re-renderize para refletir as mudanças.

React Native

Componentes de classe podem ter um estado interno (state) que armazena dados dinâmicos que podem mudar ao longo do tempo, como dados de formulários, contadores, ou qualquer outra informação que precisa ser mantida e atualizada durante a interação do usuário.

React Native

Em componentes de classe, o estado é inicializado no construtor da classe usando this.state = { /* dados iniciais */ }..

React Native

```
constructor(props) {
    super(props);
    this.state = {contador: 0};
}
```

React Native

Para atualizar o estado em um componente de classe, você deve usar o método setState().

React Native

```
this.setState({ contador: this.state.contador + 1 });
```

React Native

O método render() é obrigatório em um componente de classe. Ele retorna o que deve ser exibido na interface com base no estado atual e nas propriedades recebidas.

React Native

```
render() {
  return (
    <View>
      <Text>Contador: {this.state.contador}</Text>
      <Button title="Incrementar" onPress={this.incrementarContador} />
    </View>
  );
}
```

Hooks

React Native

Hooks são uma adição poderosa ao React que permitem que você use estado e outras funcionalidades do React em componentes funcionais, sem a necessidade de escrever uma classe.

React Native

Com a introdução dos hooks, muitos desenvolvedores preferem usar componentes funcionais. Antes dos hooks, apenas os componentes de classe podiam ter estado e outras funcionalidades do React, como ciclos de vida.

React Native

1. useState

O useState é usado para adicionar estado a componentes funcionais. Ele retorna um par de valores: o estado atual e uma função para atualizar esse estado. É ideal para gerenciar dados locais dentro de um componente.

React Native

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

  function handleClick() {
    setContador(contador + 1);
  }

  return (
    <View style={styles.container}>
      <Text style={styles.text}>Você clicou {contador} vezes</Text>
      <Button title="Clique aqui" onPress={handleClick} />
    </View>
  );
}
```

React Native

1. useEffect

O useEffect é usado para realizar efeitos secundários em componentes funcionais. Ele serve para lidar com operações que não têm impacto direto na renderização, como buscar dados de uma API, alterar o título da página, adicionar eventos, etc.

React Native

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

  function handleClick() {
    setContador(contador + 1);
  }

  function updateDocumentTitle() {
    document.title = `Você clicou ${contador} vezes`;
  }

  useEffect(function() {
    updateDocumentTitle();
  }, [contador]);

  return (
    <View style={styles.container}>
      <Text style={styles.text}>Você clicou {contador} vezes</Text>
      <Button title="Clique aqui" onPress={handleClick} />
    </View>
  );
}
```

React Native

- Componentes são os blocos de construção da interface do usuário no React Native.
- Funcionais: Usados para componentes simples e reutilizáveis.
- De Classe: Utilizados para componentes que necessitam de mais funcionalidades de ciclo de vida e gerenciamento de estado.
- Hooks: Facilitam o uso de estado e outros recursos em componentes funcionais.

Props

React Native

Props (Propriedades)

As props são objetos que contêm dados passados para componentes React Native. Elas são usadas para enviar dados de um componente pai para um componente filho.

React Native

```jsx
// Componente filho que recebe uma propriedade 'nome'
const Saudacao = (props) => {
  return (
    <View style={styles.container}>
      <Text>Olá, {props.nome}!</Text>
    </View>
  );
};

// Componente pai que renderiza o componente 'Saudacao' passando a propriedade 'nome'
const App = () => {
  return (
    <View style={styles.container}>
      <Saudacao nome="João" />
      <Saudacao nome="Maria" />
    </View>
  );
};
```

React Native

Props permitem que você envie informações de um componente pai para um componente filho durante a sua criação e uso.

São imutáveis, o que significa que um componente filho não pode modificar diretamente os valores recebidos através das props.