DIFERENÇA ENTRE COMPONENTES DE FUNÇÃO E COMPONENTES DE CLASSE EM REACT

Um componente pode ser uma classe com um método render() e suas heranças do React.Component ou pode ser uma função. Em ambos os casos, recebem props como entrada de dados e retornam uma árvore de elementos como resultado.

Componente de Função

Chamamos esses componentes de "componentes de função" porque, na verdade, são funções JavaScript. Estas funções são um componente React válido porque aceitam um único parâmetro de objeto "props" (propriedades) com dados.

```
function ProfilePage(props) {
  const showMessage = () => {
    alert('Followed ' + props.user);
  };

  const handleClick = () => {
    setTimeout(showMessage, 3000);
  };

  return (
    <button onClick={handleClick}>Follow</button>
  );
}
```

Componentes de Classe

Os componentes de classe não são específicos do React, são implementações que aparecem no ES6. Classe é apenas uma função especial que pode ser definida usando a palavra-chave class. Sempre estende o Component da biblioteca React. A documentação diz que os componentes de

classe são usados apenas quando esse componente precisa controlar o estado, caso contrário, podemos usar componentes funcionais.

```
class ProfilePage extends React.Component {
  showMessage = () => {
    alert('Followed ' + this.props.user);
  };

  handleClick = () => {
    setTimeout(this.showMessage, 3000);
  };

  render() {
    return <button onClick={this.handleClick}>Follow</button>;
  }
}
```

Props vs States

Propriedades (props) são informações que podem ser passadas para um componente. Pode ser uma string, um número etc. Este valor pode então ser utilizado pelo componente que a recebe.

Estados (states) são propriedades do componente onde colocamos dados que, quando mudados, devem causar uma nova renderização. Diferente das propriedades, o estado não é repassado ao componente e sim configurado dentro dele.

Apesar de ambos guardarem informações que influenciam no resultado da renderização, eles são diferentes por uma razão importante: props são passados para o componente (como parâmetros de funções), enquanto state é gerido de dentro do componente (como variáveis declaradas dentro de uma função). (https://pt-pt.reactjs.org/docs/faq-state.html)

Métodos de estado e ciclo de vida

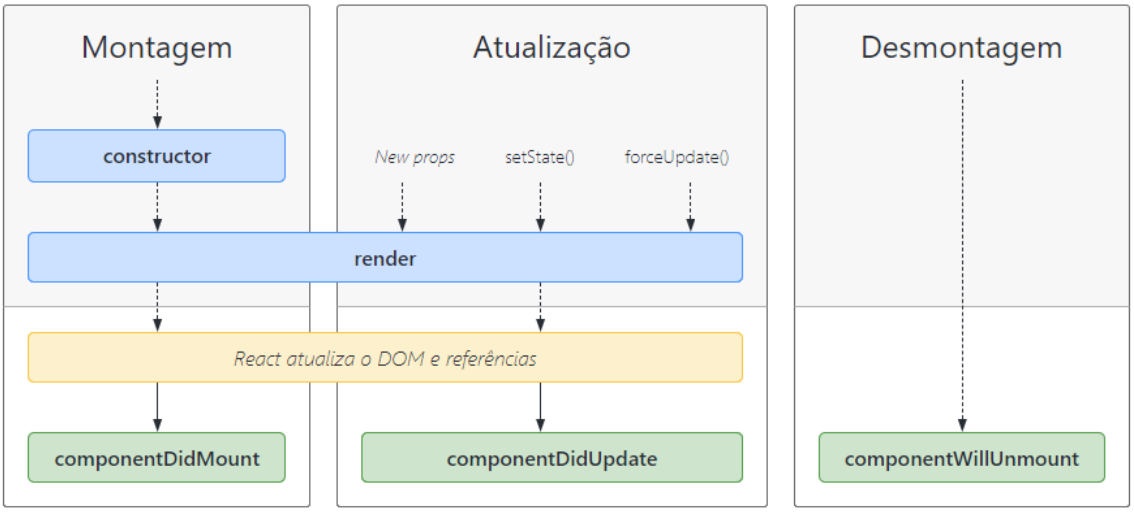

Figura 2 Ciclo de vida no React 16.4. Fonte: https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

Antes dos Hooks

Apenas os componentes de classe podiam lidar com estado. Os componentes funcionais não conseguiam lidar com métodos de estado e ciclo de vida. Atualmente, os Hooks podem ser usados para permitir que os programadores usem o estado em componentes funcionais também.

Ciclo de vida sem Hooks

Da forma antiga, utilizando class components o acesso ao ciclo de vida (momento em que o componente é carregado, por exemplo) é fornecido através dos métodos chamados *lifecycle methods*, cuja execução ocorre dado um determinado ciclo de vida do componente.

• **componentWillMount()**: chamado uma vez, antes da renderização inicial.

• **componentDidMount()**: chamado uma vez, após a renderização do componente.

• **componentWillReceiveProps()**: chamado quando um componente recebe uma nova propriedade. Usa-se este método para prever ações a serem tomadas antes do render().

• **shouldComponentUpdate()**: chamado antes de renderizar quando novas props ou states serem recebidas. Este método é utilizado para validar se o componente precisa ou não ser atualizado, retornando um resultado booleano.

• **componentWillUpdate()**: é chamado antes de renderizar quando novas props ou states são recebidas. Este método é usado para preparar os dados antes da renderização.

• **componentDidUpdate()**: é chamado quando a atualização do componente manipula o DOM. Este método é usado para manipular o DOM após a renderização.

• **componentWillMount()**: é chamado logo quando o componente é desmontado do DOM. Este método é usado para a limpeza de listeners, timers ou elementos criados no componentDidMount().

Ciclo de vida com Hooks

O React fornece a função *useState*, que retornará um array com dois elementos, onde o primeiro é a constante que armazena aquele estado e o segundo é uma função para substituir o valor daquele estado. Por convenção utilizamos atribuição por desestruturação dos itens desse array em constantes com os nomes no padrão *[state, setState]* para manter a clareza do que está sendo retornado.

Outro hook importante criado pelo time do React é o *useEffect*. Ele permite que o seu componente, em forma de função, tenha acesso aos métodos de ciclo de vida. Na prática, o *useEffect* nesse contexto é equivalente ao componentDidMount e ao componentDidUpdate. Ele invocará a função passada quando o componente é montado e quando é atualizado. O *useEffect* também te dá uma forma de fazer a limpeza de recursos – exatamente o que você usaria no componentWillUnmount. Para isso, basta retornar uma função de limpeza.

Opcionalmente, a função *useEffect* pode receber como parâmetro um array. Se este for vazio, a função de callback passada como parâmetro será executada quando o componente for montado (componentDidMount), mas não quando o mesmo for atualizado (componentDidUpdate).

Resumidamente, as principais diferenças são:

Componentes funcionais	Componentes de classe
Função que aceita props como argumento e retorna um elemento React	Um componente de classe exige que você estenda de React.Component()
Não existe método `render()` para exibir os elementos	É obrigatório que tenha o método `render` retornando HTML
Conhecido como *Componentes Stateless*, aceitam dados sem necessidade de ter lógica	Conhecido também como *Stateful Components*, visto que é implementado lógica e estados
São usados Hooks para seu ciclo de vida (como *useEffect* e *useState)*	São usados métodos para manipular o ciclo de vida