

Introdução aos Hooks e useState

Labenu_



O que vamos ver hoje?

- Revisão de componentes funcionais
- Hooks
- useState
- Regras dos Hooks



Revisão

Labenu_



Componentes Funcionais

- Tipo mais "simples" de componente
- Declarado somente como uma **função**
- Recebe **props como argumento** da função
- Declaramos **funções auxiliares** no próprio **corpo da função**



Componentes Funcionais

```
1 import React from 'react'
2
3 export function MeuComponente(props) {
4
5   const onClickButton = () => {
6     alert('Você clicou!')
7   }
8
9   return <div>
10     <p>{props.texto}</p>
11     <button onClick={onClickButton}>Clique aqui!</button>
12   </div>
13 }
```



Componentes Funcionais

- Não possui **estado**, nem métodos de **ciclo de vida**... até agora!
- **Hooks** possibilitam a inclusão dessas e outras funcionalidades antes restritas aos **componentes de classe**



Hooks - Introdução

Labenu_



Hooks

- "Nova" funcionalidade do React (fev/2019)
- Grupo de **funções** que adicionam **capacidades extras** aos **componentes funcionais**
- Substituem o uso de componentes de classe
- Ainda em **progressiva adoção** no mercado



Hooks - Motivação

- Dois motivos principais:
 - **Reutilização de lógicas de estado e lifecycle**
 - Classes são difíceis (this tem uso muito confuso)
- Componentes de classe **não serão removidos** e podem continuar sendo usados normalmente
- É apenas uma **nova forma** de fazer as mesmas coisas!



useState

Labenu_



useState

- Hook mais comum, é análogo ao **state** nos componentes de classe
- Cria uma **variável de estado** em um componente funcional e permite **atualizá-la** por meio de uma função
- **Exemplo:** contador

Vamos ver na prática! 



Declarando uma variável de estado



- Para declarar uma variável de estado, usamos a função **useState**
- Ela deve ser importada do React, entre {chaves}

```
import React, {useState} from 'react'
```

Adicionar isso ao topo do
arquivo quando quisermos usar
o estado



Declarando uma variável de estado



```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>0 valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- Essa função recebe o valor inicial do estado
- Ela retorna duas variáveis entre [colchetes]:
 - Estado Atual
 - Função que atualiza o estado (e avisa o React)



Declarando uma variável de estado



```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>O valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- **Essa função recebe o valor inicial do estado**
- Ela retorna duas variáveis entre [colchetes]:
 - Estado Atual
 - Função que atualiza o estado (e avisa o React)



Declarando uma variável de estado



```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>O valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- Essa função recebe o valor inicial do estado
- **Ela retorna duas variáveis entre [colchetes]:**
 - Estado Atual
 - Função que atualiza o estado (e avisa o React)



Declarando uma variável de estado



```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>0 valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- Essa função recebe o valor inicial do estado
- Ela retorna duas variáveis entre [colchetes]:
 - Estado Atual
 - Função que atualiza o estado (e avisa o React)



Declarando uma variável de estado



```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>O valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- Essa função recebe o valor inicial do estado
- Ela retorna duas variáveis entre [colchetes]:
 - Estado Atual
 - Função que atualiza o estado (e avisa o React)



Usando uma variável de estado

```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>O valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- A variável de estado é usada como qualquer outra variável (não precisa de **this.state**)
- Podemos usar ela no meio do JSX, passar como props para outros componentes, etc...



Atualizando o estado



```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>O valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- Devemos chamar o segundo argumento retornado pelo useState (substitui **this.setState**)
- Ele recebe como parâmetro o novo valor do estado (direto o novo valor, não mais um objeto)



Atualizando o estado



```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>O valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- Devemos chamar o **segundo argumento retornado pelo useState** (substitui **this.setState**)
- Ele recebe como parâmetro o novo valor do estado (direto o novo valor, não mais um objeto)



Atualizando o estado



```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const [valorContador, setValorContador] = useState(0)
5
6   const somaContador = () => {
7     setValorContador(valorContador + 1)
8   }
9
10  return <div>
11    <p>O valor do contador é: {valorContador}</p>
12    <button onClick={somaContador}>Soma</button>
13  </div>
14 }
```

- Devemos chamar o segundo argumento retornado pelo useState (substitui **this.setState**)
- **Ele recebe como parâmetro o novo valor do estado (direto o novo valor, não mais um objeto)**



Diferença

- O estado **não é mais um único objeto** por componente
- O useState representa **uma** variável de estado
- É possível ter vários useState por componente, ou seja, **várias variáveis de estado**
- Cada variável possui a sua **função de atualização**



```

1 export class UserForm extends React.Component {
2   state = {
3     nameValue: '',
4     emailValue: ''
5   }
6
7   onChangeName = (event) => {
8     this.setState({nameValue: event.target.value})
9   }
10
11  onChangeEmail = (event) => {
12    this.setState({emailValue: event.target.value})
13  }
14
15  render() {
16    return (
17      <div>
18        <input
19          value={this.state.nameValue}
20          onChange={this.onChangeName}
21        />
22        <input
23          value={this.state.emailValue}
24          onChange={this.onChangeEmail}
25        />
26      </div>
27    )
28  }
29 }

```

```

1 export function UserForm(props) {
2   const [nameValue, setNameValue] = useState('')
3   const [emailValue, setEmailValue] = useState('')
4
5   const onChangeName = (event) => {
6     setNameValue(event.target.value)
7   }
8
9   const onChangeEmail = (event) => {
10     setEmailValue(event.target.value)
11   }
12
13   return <div>
14     <input
15       value={nameValue}
16       onChange={onChangeName}
17     />
18     <input
19       value={emailValue}
20       onChange={onChangeEmail}
21     />
22   </div>
23 }

```



```

1 export class UserForm extends React.Component {
2   state = {
3     nameValue: '',
4     emailValue: ''
5   }
6
7   onChangeName = (event) => {
8     this.setState({nameValue: event.target.value})
9   }
10
11  onChangeEmail = (event) => {
12    this.setState({emailValue: event.target.value})
13  }
14
15  render() {
16    return (
17      <div>
18        <input
19          value={this.state.nameValue}
20          onChange={this.onChangeName}
21        />
22        <input
23          value={this.state.emailValue}
24          onChange={this.onChangeEmail}
25        />
26      </div>
27    )
28  }
29 }

```

```

1 export function UserForm(props) {
2   const [nameValue, setNameValue] = useState('')
3   const [emailValue, setEmailValue] = useState('')
4
5   const onChangeName = (event) => {
6     setNameValue(event.target.value)
7   }
8
9   const onChangeEmail = (event) => {
10    setEmailValue(event.target.value)
11  }
12
13  return <div>
14    <input
15      value={nameValue}
16      onChange={onChangeName}
17    />
18    <input
19      value={emailValue}
20      onChange={onChangeEmail}
21    />
22  </div>
23 }

```



Declaração do estado



Atualização do estado



Passando função onChange



Usando
o estado



E o resto? 🤔

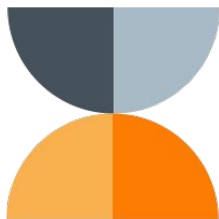
- Todas as **outras propriedades** continuam iguais!
- O estado **não pode ser alterado diretamente**, somente através da função de atualização
- Regras de **arrays** e **objetos** são as mesmas
- Componente é **renderizado** quando o estado é atualizado



E o jeito de pensar? 🤔

- A lógica por trás do estado **é a mesma!**
- Só muda o jeito de declarar, usar e atualizar
- É necessário se acostumar com a nova sintaxe e, no início, é **normal** que confusões aconteçam





Pausa para relaxar 🧘

10 min

- Hooks dão novas funcionalidades aos componentes **funcionais**
- **useState** permite usar o estado em componentes funcionais
- **Muda a sintaxe**, mas a lógica e a forma de pensar são os mesmos



Hooks

Labenu_



useState

- Usamos o **useState** para adicionar **estado** aos nossos componentes funcionais
 - Função que adiciona **funcionalidade** ao componente
- Como ela, existem algumas funções que adicionam coisas extras aos **componentes funcionais**
- Essas funções são o que chamamos de **Hooks**



Hooks

- Alguns exemplos de outros hooks:
 - **useEffect:** adiciona side-effects aos componentes
 - **useCallback** e **useMemo:** otimizações de performance
 - **useReducer:** gerenciamento de estados complexos
 - **useContext:** consumir informações de componentes acima na árvore
- Também podemos criar nossos próprios Hooks!



Regras dos Hooks


- O uso de qualquer hook deve seguir 2 regras:
 1. Apenas utilizar hooks no **nível superior** do componente
 2. Apenas utilizar hooks dentro de **componentes funcionais** do React




1. Apenas no nível superior

- Não use Hooks dentro de loops, condicionais ou funções aninhadas, **apenas no nível superior do componente**

```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   const somaContador = () => {
5     const [valorContador, setValorContador] = useState(0)
6     setValorContador(valorContador + 1)
7   }
8
9   return <div>
10     <p>0 valor do contador é: {valorContador}</p>
11     <button onClick={somaContador}>Soma</button>
12   </div>
13 }
```




```
1 import React, {useState} from 'react'
2
3 export function Contador() {
4   if(condicao) {
5     const [valorContador, setValorContador] = useState(0)
6   }
7
8   const somaContador = () => {
9     setValorContador(valorContador + 1)
10   }
11
12   return <div>
13     <p>0 valor do contador é: {valorContador}</p>
14     <button onClick={somaContador}>Soma</button>
15   </div>
16 }
```




2. Apenas Componentes Funcionais

- Não use Hooks dentro de qualquer outra função que não seja um componente funcional ou Custom Hook

```
1 import {useState} from 'react'
2
3 export function soma(num1, num2) {
4   const [valorContador, setValorContador] = useState(0)
5
6   return num1 + num2
7 }
```



```
1 import React from 'react'
2
3 export class Componente extends React.Component {
4   render() {
5     const [valorContador, setValorContador] = useState(0)
6
7     return <div>{valorContador}</div>
8   }
9 }
```





Exercício 1

- Crie um **input controlado** por estado usando useState
- Crie um **botão** que adiciona o texto digitado a uma lista
- Mostre a **lista de itens** adicionados na tela
- Tudo deve ser feito em um componente funcional!



Resumo

Labenu_



Resumo

- Hooks dão novas funcionalidades aos componentes **funcionais**
- **useState** permite usar o estado em componentes funcionais
- **Muda a sintaxe**, mas a lógica e a forma de pensar são os mesmos



```

1 export class UserForm extends React.Component {
2   state = {
3     nameValue: '',
4     emailValue: ''
5   }
6
7   onChangeName = (event) => {
8     this.setState({nameValue: event.target.value})
9   }
10
11  onChangeEmail = (event) => {
12    this.setState({emailValue: event.target.value})
13  }
14
15  render() {
16    return (
17      <div>
18        <input
19          value={this.state.nameValue}
20          onChange={this.onChangeName}
21        />
22        <input
23          value={this.state.emailValue}
24          onChange={this.onChangeEmail}
25        />
26      </div>
27    )
28  }
29 }

```

```

1 export function UserForm(props) {
2   const [nameValue, setNameValue] = useState('')
3   const [emailValue, setEmailValue] = useState('')
4
5   const onChangeName = (event) => {
6     setNameValue(event.target.value)
7   }
8
9   const onChangeEmail = (event) => {
10    setEmailValue(event.target.value)
11  }
12
13  return <div>
14    <input
15      value={nameValue}
16      onChange={onChangeName}
17    />
18    <input
19      value={emailValue}
20      onChange={onChangeEmail}
21    />
22  </div>
23 }

```



Declaração do estado



Atualização do estado



Passando função onChange



Usando
o estado



Resumo

Regras dos Hooks:

- 1) Só usar hooks no **nível superior** da função
 - Não pode estar dentro de ifs, whiles ou funções
- 2) Só usar hooks em **componentes funcionais** do React



Dúvidas? 🧐

Labenu_





Obrigado(a)!