

FIAP
GRADUAÇÃO

TDS

Front-End Design Engineering

Prof. Alexandre Carlos profalexandre.jesus@fiap.com.br

Prof. Luís Carlos lsilva@fiap.com.br

useState React





useState

45697056
■ ■ ■

useState é um **hook** fundamental da biblioteca **React**. Ele permite a criação e manipulação de **estados** em componentes funcionais.

O estado dos componentes é algo muito importante no React, pois é ele que indica a hora de atualizar os dados do componente na tela. De outra forma, o react não sente a necessidade de atualizar.

Vamos fazer alguns exemplos...





Usando State

45697056
■ ■ ■

Quando temos valores em nossos componentes que queremos que sejam atualizados na tela juntamente com os valores internamente, usamos os states ao invés de uma variável simples. Crie um componente chamado **TesteState.tsx** e faça o código.

```
import { useState } from "react"
```

```
export default function TesteState(){
```

```
  const [aluno, setAluno] = useState("João")
```

```
  return(
```

```
    <>
```

```
    <p>Aluno: {aluno}</p>
```

```
    <button onClick={() => setAluno("Maria")} >Mudar</button>
```

```
    </>
```

```
  )
```

```
}
```

Chamando o useState

Forma de declarar o useState

Valor sempre será alterado pela sua função



React

State vs Variáveis

Para entendermos melhor a diferença entre o state e uma variável simples, vamos fazer mais um exemplo. Crie um componente chamado `TesteState2.tsx` e insira o código ao lado:

Ao testar, repare que apenas o parágrafo de `valorState` atualiza na tela.

OBS. Não esqueça de chamar o nosso componente em App.

```
import { useState } from "react"

export default function TesteState2(){

  const [valorState, setValorState] = useState(5)
  let valorVariavel = 5

  let aumentar = ()=>{
    setValorState(valorState + 5)
    valorVariavel += 5
  }

  return(
    <>
      <p>Valor State: {valorState}</p>
      <p>Valor Variável: {valorVariavel}</p>
      <button onClick={aumentar}>Aumentar</button>
    </>
  )
}
```



Recebendo dados do Usuário

45697056
■ ■ ■

Vamos ver agora como receber valores do usuário. Podemos atualizar os dados da tela com os que o usuário digitar. Crie um componente chamado `TesteState3.tsx`, nele vamos atualizar o texto com o valor digitado pelo usuário.

```
import { useState } from "react"
```

```
export default function UseState1(){
```

```
  const [nome, setNome] = useState('');
```

```
  return(
```

```
    <>
```

```
    <p>O nome do usuário é: {nome}</p>
```

```
    <input type="text" name={nome} placeholder="Digite o nome do usuário"
      onChange={(e)=>setNome(e.target.value)}/>
```

```
    </>
```

```
  )
```

```
}
```

Atualiza dos dados no input

Representa o evento

Pega o valor digitado



Atualizando booleanos

Conseguimos trabalhar com valores booleanos. Neste exemplo vamos fazer um componente ser montado e desmontado. Crie o `TesteState4.tsx`, faça conforme abaixo.

Monta e desmonta o componente

Atualiza o texto do botão

```
import { useState } from "react"
import Filho from "../Filho";

export default function TesteState4(){

  const [filho, setFilho] = useState(false);

  return(
    <>
      <h1>Teste State 4</h1>
      {filho ? <Filho/> : ''}
      <button onClick={()=>setFilho(!filho)}>
        {filho ? 'Excluir Filho':'Criar Filho'}
      </button>
    </>
  )
}
```




Atualizando arrays

45697056
■ ■ ■

Também podemos trabalhar com arrays. Crie o `TesteState5.tsx`, faça conforme abaixo.

```
import { useState } from "react"

export default function TesteState5(){

  const [carros, setCarros] = useState(['Onix','Polo','HB20']);
  const [carro, setCarro] = useState('');

  return(
    <>
    <p>Carros: {carros.map(car=> <span>{car} </span>)}</p>
    <input type="text" name={carro} onChange={(e)=>setCarro(e.target.value)}>
    <button onClick={()=>setCarros([...carros,carro])}>Adicionar</button>
    </>
  )
}
```



Tipando o useState

45697056
■ ■ ■

Podemos padronizar nossos projetos, isso também cabe a tipagem dos nossos useStates. Veja alguns exemplo de como tipar eles:

```
import { useState } from "react"

type Pet = {nome: string; idade: number;}

export default function TesteState5(){

  const [count, setCount] = useState<number>(0) //tipando um number
  const [casado, setCasado] = useState<boolean>(true) //tipando um boolean
  const [nome, setNome] = useState<string>('Luís') //tipando uma string
  const [carros, setCarros] = useState<string[]>(['Onix', 'Polo', 'HB20']); //tipando um array
  const [pet, setPet] = useState<Pet>({nome: 'Spark', idade: 18 }); //tipando um objeto
  const [pets, setPets] = useState<Pet[]>([
    {nome: 'Spark', idade: 18 },
    {nome: 'Licica', idade: 15 }
  ]); //tipando um array de objetos
```



Passando State para o filho

45697056
■ ■ ■

Podemos passar o valor do state que está no pai para um componente filho.

```
import { useState } from "react"
import Filho from "../Filho"

export default function TesteState6(){

  const [count, setCount] = useState<number>(0)
  const aumentar = ()=>{setCount(count + 1)}

  return(
    <>
    <h1>Teste State 6</h1>
    <p>Contador: {count}</p>
    <Filho count={count} aumentar={aumentar}/>
    </>
  )
}
```

```
type CountProps = { count: number; aumentar: ()=>void }

export default function Filho({count,aumentar}:CountProps){

  return(
    <div>
      <h2>Filho de Teste State 4</h2>
      <p>Contagem: {count}</p>
      <button onClick={aumentar}>Aumentar</button>
    </div>
  )
}
```



Exercício

45697056
■ ■ ■

1. Crie uma aplicação chamada exercício-state;
2. Em App.tsx crie um useState chamado “count” , uma função para aumentar e outra para diminuir seu valor. Agora passe eles por props para um componente chamado Contagem.tsx;
3. Em App.tsx crie um useState chamado “frutas” começando com 2 frutas e uma função para inserir novas frutas. Agora crie uma lista não ordenada usando map para mostrar as frutas;
4. Em App.tsx crie um useState chamado “contagem” com o valor ‘false’. Agora crie um ternário e um botão que façam o componente Contagem aparecer e sumir da tela;




DUVIDAS

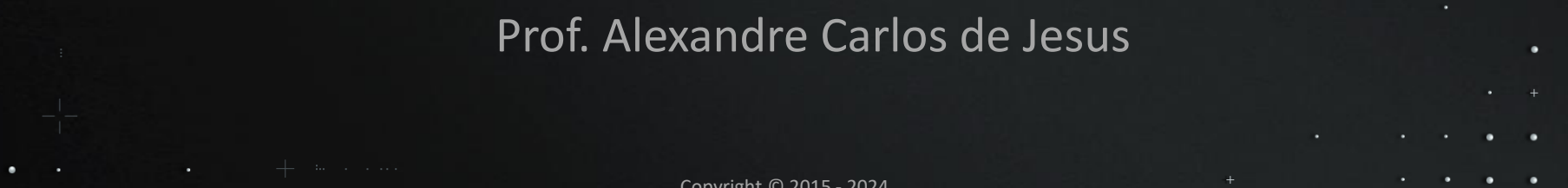




FLAP



Prof. Luís Carlos S. Silva
Prof. Alexandre Carlos de Jesus



Copyright © 2015 - 2024

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).