FIAP

FRONT-END DESIGN ENGINEERING

REACT - USEEFFECT

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

Prof. Luís Carlos lsilva@fiap.com.br

Prof. Wellington Cidade profwellington.tenorio@fiap.com.br

USEEFFECT

CICLO DE VIDA DOS COMPONENTES

Em uma aplicação precisamos organizar e integrar as ações de forma que as atividades fluam e cada tarefa seja executada no momento e forma correta.

- Nas aplicações do React utilizamos um hook que controla e escuta o ciclo de vida destes componentes, é o useEffect.

USEEFFECT

UTILIZANDO O USEEFFECT NA APLICAÇÃO

Vamos fazer alguns exemplos para entender que fazes do ciclo de vida dos componentes podemos manipular.

Após criar uma nova aplicação React utilizando o Vite, limpe o componente App.tsx desta forma:

```tsx
import { useState } from 'react'
import './App.css'

function App() {
  const [valor, setValor] = useState<number>(0)

  function aumentar(){setValor(valor + 1)}

  return (
    <div>
      <h1>Componente App</h1>
    </div>
  )
}
export default App
```

Vamos criar um state para fazermos algumas interações.

USEEFFECT

UTILIZANDO O USEEFFECT NA APLICAÇÃO

Dentro de **src**, crie a pasta **components** e nela um componente chamado **ExemploEffect.tsx**. Vamos utilizar nosso state valor dentro dele.

```tsx
type ValorProps ={ valor: number; aumentar: ()=>void;}

export default function ExemploEffect({valor,aumentar}:ValorProps){

    return(
        <div>
            <h2>Exemplo Effect</h2>
            <p>Valor do State: {valor}</p>
            <button onClick={aumentar}>Incrementar</button>
        </div>
    )
}
```

Temos aqui uma contagem simples, acrescentando 1 ao valor sempre que for clicado no botão..

USEEFFECT

UTILIZANDO O USEEFFECT NA APLICAÇÃO

Não se esqueça de chamar o **ExemploEffect** dentro do componente **App** e passar o **valor** e **setValor** por props. Assim o componente filho poderá utilizar o state armazenado no componente pai.

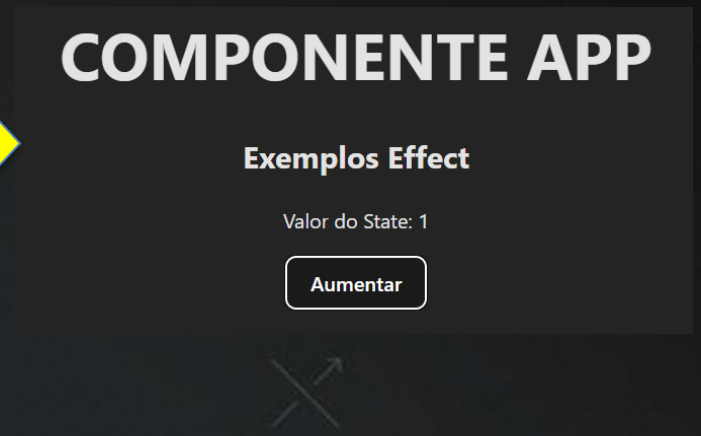
```
import { useState } from 'react'
import './App.css'
import ExemploEffect from './components/ExemploEffect'

function App() {
  const [valor, setValor] = useState<number>(0)

  function aumentar(){setValor(valor + 1)}

  return (
    <div>
      <h1>Componente App</h1>
      <ExemploEffect valor={valor} aumentar={aumentar} />
    </div>
  )
}
export default App
```

Agora já conseguimos utilizar o state valor a partir do componente ExemploEffect, certo?



USEEFFECT

CHAMANDO A AÇÃO EM QUALQUER EVENTO DO COMPONENTE

A partir de agora podemos testar vários eventos no ciclo de vida de nosso componentes.
Eventos como **criação**, **alteração**, **alteração de um valor específico** e até a **exclusão** dele na página.
Para controlar estes eventos utilizaremos o Hook **useEffect** no componente **"ExemploEffect"**, vamos começar utilizando ele para disparar uma mensagem sempre que ele tiver um evento, seja qual for.

```
1  import { useEffect } from "react";
2
3  type ValorProps ={ valor: number; aumentar: ()=>void;}
4
5  export default function ExemploEffect({valor,aumentar}:ValorProps){
6
7      useEffect(()=>{
8          console.log('Em todas as alterações eu sou chamado!!!');
9      })
10
11     return(
```

O método do useEffect espera receber uma função anônima, que apresentada desta forma, é chamada sempre que temos alguma mudança de estado em nosso componente.

Abra o inspetor de código do navegador para vermos a mensagem sendo exibida no console.

USEEFFECT

OBSERVAÇÃO IMPORTANTE

A partir da versão 18 do react, sempre que precisamos trabalhar com o useEffect você vai perceber que se você realizar alguma impressão de saída de dados, ela será duplicada, por causa da dupla verificação.

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

React.StrictMode na utilização de useEffect ativa dupla verificação.

USEEFFECT

CHAMANDO A AÇÃO SOMENTE NA CRIAÇÃO DO COMPONENTE

Aqui temos o useEffect somente executando a função quando o componente é criado. Repare que, o que mudou no código foi que, no final da função, colocamos uma virgula e um par de colchetes vazios. Nestes colchetes ele espera um array de valores que ele deve observar, como está vazio, ele só executa quando o componente é criado.

```
10
11      //Exemplo de chamada na criação
12      useEffect(()=>{
13          console.log('Sou chamado só quando o comp. é criado!!!');
14      },[])
```

Como o array está vazio, ele só executa a primeira vez, quando criado, depois não tem mais ninguém para acompanhar.

USEEFFECT

CHAMANDO A AÇÃO QUANDO UM VALOR É ALTERADO

Como já começamos falar no exemplo anterior, também podemos controlar a chamada de quando um valor específico é alterado. Para fazer isso, devemos indicá-lo dentro do array no final da função.

```
//Exemplo de chamada na mudança de um valor específico
useEffect(()=>{
    console.log(`Sou chamado só quando o ${valor} muda!!!`);
},[valor])
```

Aqui ele será chamado sempre que valor sofrer alteração.

USEEFFECT

CHAMANDO A AÇÃO QUANDO O COMPONENTE É EXCLUÍDO

Por fim vamos ver como fazer para, se o componente for destruído, ou desmontado, retirado da tela, podemos também pegar este evento. Vamos ter que passar a ação como uma função sendo retornada da função anônima que temos nele.

```
21      //Exemplo de chamada quando o elemento é excluído
22      useEffect(()=>{
23          return ()=> console.log(`Ops, me apagaram!`);
24      },[])
```

Retornamos a função que queremos executar.

USEEFFECT

CHAMANDO A AÇÃO QUANDO O COMPONENTE É EXCLUÍDO

Mas para vermos funcionando, vamos ao componente App. Aqui criaremos um state chamado filho para controlarmos sua criação e exclusão.
A lógica que vamos usar é bem simples, se filho for igual a true mandamos inserir o componente, se for false vamos substituí-lo por uma string vazia. Para isso vamos usar o ternário.

Criação do state filho.

Criação da lógica como falamos acima.

```
5  function App() {
6    const [valor, setValor] = useState<number>(0)
7    const [filho, setFilho] = useState<boolean>(true)
8
9    function aumentar(){setValor(valor + 1)}
10
11   return (
12     <div>
13       <h1>Componente App</h1>
14       <button onClick={()=>setFilho(!filho)}>{filho ? "Apagar" : "Criar"}</button>
15       {filho ? <ExemploEffect valor={valor} aumentar={aumentar} /> : ''}
16     </div>
```

EXERCÍCIO

Crie um novo projeto chamado exercicio-useeffect.

Crie uma pasta chamada components e dentro um arquivo chamado Aviao.tsx, ele deve ter um h2 o identificando e um state chamado altura, que deve começar em “0” e apresentar a altura em um parágrafo.

Você deve criar um botão que quando clicado aumente a altura em 100.

Após a primeira parte apresente no console, usando o useEffect, mensagens avisando que:

“o Avião está ligado!!!” sempre que sofrem uma alteração.

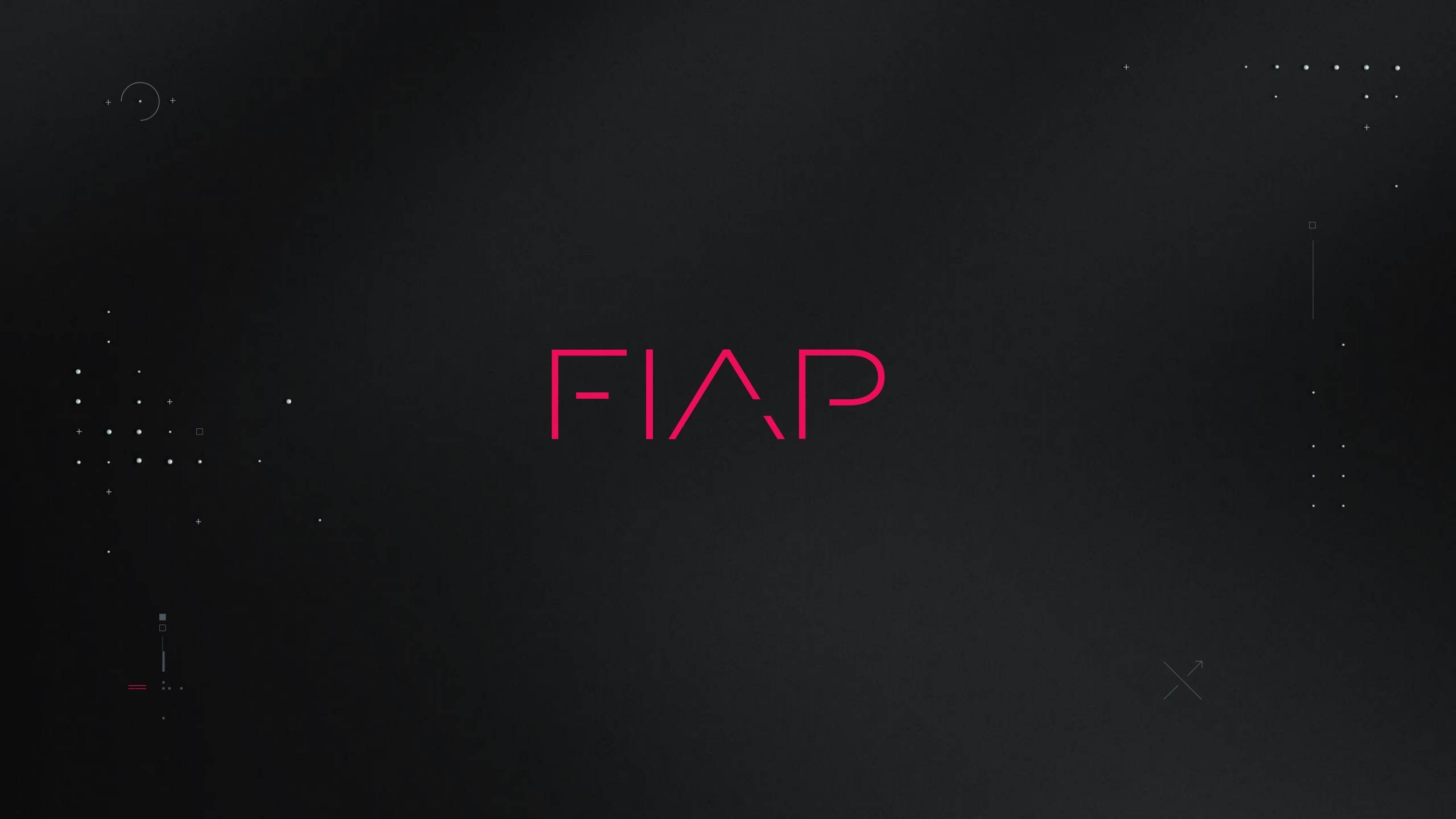
“o avião decolou” (quando o componente for criado),

“O avião está em XXX pés”(quando ele subir através do botão) e

“O avião foi derrubado”(quando o componente for removido).

OBRIGADO

FIAP


FIAP