

FIAP

GRADUAÇÃO

45697056



TDS

Responsive Web Development

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

Prof. Luís Carlos lsilva@fiap.com.br

Prof. Renato Bortolin

45697056



INTRODUÇÃO



React





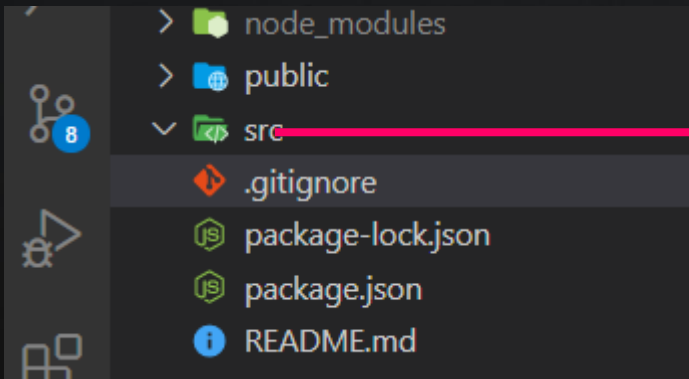
React

Novo projeto c/ pasta src do zero

45697056

■ ■ ■

Para iniciarmos nossos projeto da aula de hoje, vamos criar um novo projeto chamado `react-aula2` e vamos **apagar** todos os arquivos da pasta `src`.



Apagamos todos
os arquivos da
pasta src



React

Novo projeto c/ pasta src do zero

O próximo passo será recriar o arquivo index.js na pasta src. Insira o código abaixo para que os componentes que vamos criar a seguir possam ser renderizados no arquivo index.html.

JS index.js M X

src > JS index.js

```
1 import React from 'react'
2 import ReactDOM from 'react-dom'
3
4 ReactDOM.render(
5   <h1>Conteúdo de Index.js</h1>,
6   document.getElementById('root')
7 )
8
```

Nos permite usar o JSX

Nos permite usarmos o VDOM

Método para renderizar os componentes na tela

Conteúdo que será renderizado

Local onde será renderizado

Repare em nossa página no navegador, agora só temos a tag h1.



React

Novo projeto c/ pasta src do zero

Por boa prática, vamos criar um novo componente chamado `App.js`, ele será o nosso componente principal. A princípio vamos apenas colocar um `h1` dentro, repita o código abaixo:

```
src > JS App.js > ...
1 | import React from 'react'
2 |
3 | export default () =>{
4 |
5 |     return(
6 |         <h1>Conteúdo de App.js</h1>
7 |     )
8 | }
```

Podemos usar uma arrow function para deixar o código mais leve.



React

Novo projeto c/ pasta src do zero

45697056
■ ■ ■

Para o nosso componente App ser reproduzido na tela agora temos que inserir ele no index.js.

JS index.js M X

JS App.js M

src > JS index.js

```
1 import React from 'react'
2 import ReactDOM from 'react-dom'
3 import App from './App'
4
5 ReactDOM.render(<App/>, document.getElementById('root'))
6
```

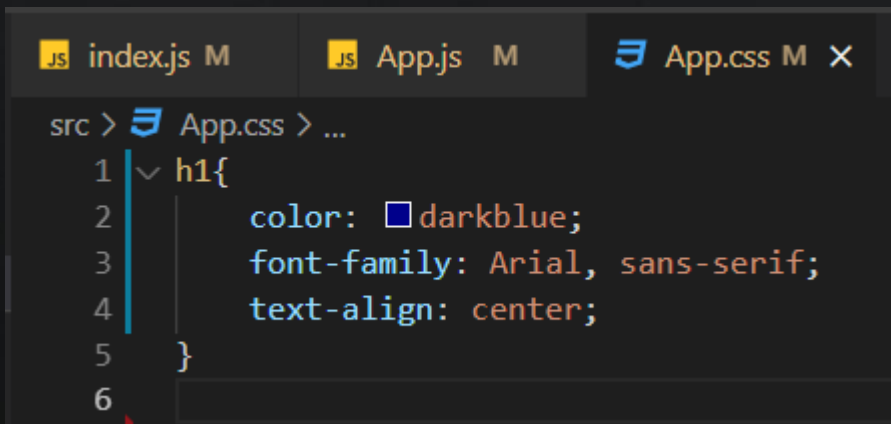
Repare que agora conseguimos até deixar em apenas uma linha.



CSS em Componentes

45697056
■ ■ ■

A forma de estilização de componentes é muito parecida com a que utilizamos nos projetos sem o React. Podemos ter arquivos de estilização CSS dedicados a um ou vários componentes. Dentro da pasta src, crie um arquivo chamado App.css e insira o código abaixo:

A screenshot of a code editor with a dark theme. The top bar shows three tabs: 'index.js M', 'App.js M', and 'App.css M X'. The 'App.css' tab is active. The editor content shows the file path 'src > App.css > ...' and a CSS rule for 'h1' with properties for color, font-family, and text-align. Line numbers 1 through 6 are visible on the left.

```
1  h1{  
2      color: darkblue;  
3      font-family: Arial, sans-serif;  
4      text-align: center;  
5  }  
6
```

OBS. Por boa prática, colocamos os nomes dos arquivos CSS iguais aos do componente.



CSS em Componentes

45697056
■ ■ ■

Criado o arquivo CSS, agora vamos importar ele dentro do arquivo App.js. Faça conforme abaixo:

```
JS index.js M JS App.js M X App.css M
src > JS App.js > ...
1 | import React from 'react'
2 | import './App.css'
3 |
4 | export default () =>{
5 |   return(
6 |     <h1>Conteúdo de App.js</h1>
7 |   )
8 | }
9 |
```

No caso do CSS o import é mais simples, não precisamos atribuir nome a ele.



React

CSS em Componentes

45697056

■ ■ ■

Para inserirmos valores de forma inline no elemento devemos nos atentar a pequenas diferenças:

```
src > JS index.js M JS App.js M X App.css M
1 | import React from "react";
2 | import "../App.css";
3 |
4 | export default () => {
5 |   return (
6 |     <h1 style={{fontSize:'4em', color:'red'}}>Conteúdo de App.js</h1>
7 |   );
8 |
9 | };
```

Ao invés de aspas devemos usar chaves duplas para inserir as propriedades no atributo style

Como é js devemos usar camel case em propriedades de nome composto

Para separar as propriedades devemos usar a virgula.



CSS em Componentes

45697056

■ ■ ■

Quando temos muitas propriedades para passar de forma inline, podemos criar um objeto, usando as propriedades como atributos:

```
4 export default () => {  
5  
6   let paragr={  
7     textAlign: 'justijy',  
8     color: 'darkgreen',  
9     textIndent: '50px',  
10    fontSize: '2em'  
11  }  
12  
13  return (  
14    <>  
15      <h1 style={{fontSize: '4em', color: 'red'}}>CSS em Componentes</h1>  
16      <p style={paragr}>Formas de inserir CSS em um componente</p>  
17    </>  
18  );  
19  
20 };
```

Criando um objeto chamado paragr e inserindo as propriedades.

Desta vez para inserir usamos chaves simples.



React

CSS em Componentes

45697056



Outro detalhe importante é que, para inserir uma classe no elemento devemos usar `className` ou invés de `class`:

```
return (  
  <>  
    <h1 style={{fontSize:'4em', color:'red'}}>CSS em Componentes</h1>  
    <p style={paragr}>Formas de inserir CSS em um componente</p>  
    <p className="exemplo">Aqui um exemplo do "className".</p>  
  </>  
);  
};
```

Usando o atributo `className`
com o valor `exemplo`

```
src > App.css > ...  
1 | h1{  
2 |   color: darkblue;  
3 |   font-family: Arial, sans-serif;  
4 |   text-align: center;  
5 | }  
6 |  
7 | .exemplo{  
8 |   color: orange;  
9 |   font-size: 2em;  
10| }
```

Criando formatação da classe
`exemplo`.



React

CSS em Componentes

45697056

■ ■ ■

O componente também pode receber a estilização quando for inserido no componente pai através de seu arquivo CSS. Crie uma pasta no src chamada **componentes** e nela crie um arquivo chamado **ComponenteTeste.js** e insira o código abaixo:

JS index.js M

JS App.js M

App.css M

JS ComponenteTeste.js U X

src > componentes > JS ComponenteTeste.js > ...

```
1 import React from 'react'
```

```
2
```

```
3 export default ()=> <p className="cTeste">Conteúdo do ComponenteTeste</p>
```

```
4
```

Repare que neste exemplo, como não temos código js na função podemos usar desta forma.



CSS em Componentes

45697056
■ ■ ■

No arquivo App.css, vamos criar as propriedades de estilização da classe `cTeste`:

```
src > App.css > ...
1 | h1{
2 |   color: darkblue;
3 |   font-family: Arial, sans-serif;
4 |   text-align: center;
5 | }
6 |
7 | .exemplo{
8 |   color: orange;
9 |   font-size: 2em;
10 | }
11 |
12 | .cTeste{
13 |   color: red;
14 |   font-size: 2em;
15 | }
16 |
```



CSS em Componentes

45697056



Agora é só chamar o componente no App.js:

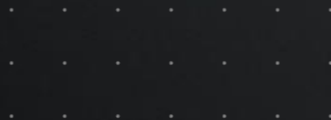
```
src > js App.js > ...
1  import React from "react";
2  import "../App.css";
3  import CompTeste from '../componentes/ComponenteTeste'
4
5  export default () => {
6
7      let paragr={
8          textAlign:'justijy',
9          color: 'darkgreen',
10         textIndent: '50px',
11         fontSize: '2em'
12     }
13
14     return (
15         <>
16         <h1 style={{fontSize:'4em', color:'red'}}>CSS em Componentes</h1>
17         <p style={paragr}>Formas de inserir CSS em um componente</p>
18         <p className="exemplo">Aqui um exemplo do "className".</p>
19         <CompTeste/>
20         </>
21     );
22 };
```



Exercício

45697056
■ ■ ■

1. Crie uma nova aplicação chamada exercício2;
2. Limpe o conteúdo da pasta src e comece a estrutura da pasta do zero, criando o index.js e o App.js como seu componente principal;
3. Crie uma pasta chamada componentes em src;
4. Crie um componente chamado Header.js, dentro uma tag header com um título h1 e um parágrafo;
5. Crie um componente chamado Corpo.js, dentro coloque um subtítulo, uma imagem e 4 parágrafos;
6. Crie um componente chamado Footer.js, dentro uma tag footer e um parágrafo;
7. Crie arquivos CSS para os componentes Header, Corpo e Footer e estilize a página.





React

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.

src > componentes > JS ComponenteTeste.js > ...

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

Chamando o useState

```
2
```

```
3 export default ()=>{
```

```
4
```

Forma de declarar o useState

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

```
6   return(
```

```
7     <>
```

```
8     <p className="cTeste">Aluno: {aluno}</p>
```

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

```
10    </>
```

```
11  )
```

```
12 }
```

Valor sempre será alterado pela sua função



Usando State no filho

Para entendermos melhor a diferença entre o state e uma variável simples, vamos fazer mais um exemplo. Crie um componente chamado **TesteState** 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`.

```
ComponenteTeste.js U  TesteState.js U x  App.js M
src > componentes > JS TesteState.js > ...
1  import React, { useState } from 'react'
2
3  export default ()=>{
4
5      const [valorState, setValorState]=useState(5)
6      let valorVariavel = 5
7      function aumentar(){
8          setValorState(valorState + 5)
9          valorVariavel+=5
10     }
11     return(
12         <>
13             <p>ValorState: {valorState}</p>
14             <p>ValorVariavel: {valorVariavel}</p>
15             <button onClick={()=>aumentar()}>Aumentar</button>
16         </>
17     )
18 }
```



Usando State

45697056
■ ■ ■

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

Crie um componente chamado TesteStateFilho.js, nele vamos usar o valorState.

```
ComponenteTeste.js U  TesteState.js U  TesteStateFilho.js U X  App.js M

src > componentes > TesteStateFilho.js > ...
1  import React from 'react'
2
3  export default props=>{
4
5      return(
6          <div>
7              <p>Valor de State no filho: {props.valor}</p>
8          </div>
9      )
10 }
```



No componente TesteState devemos utilizar o TesteStateFilho, mas não esquecendo de passar o state como atributo para ser usado pelo props.

Usando State

```
45697056
■ ■ ■

import React, { useState } from 'react'
import TesteStateFilho from './TesteStateFilho'

export default ()=>{

  const [valorState, setValorState]=useState(5)
  let valorVariavel = 5
  function aumentar(){
    setValorState(valorState + 5)
    valorVariavel+=5
  }
  return(
    <>
      <p>ValorState: {valorState}</p>
      <p>ValorVariavel: {valorVariavel}</p>
      <button onClick={()=>aumentar()}>Aumentar</button>
      <TesteStateFilho valor={valorState}/>
    </>
  )
}
```



Usando State

45697056

■ ■ ■

Para o filho atualizar o valor do pai ele pode usar a função que recebe dele. Vamos usar o botão aumentar dentro do componente filho. Não se esqueça que agora ele esta dentro de props.

```
ComponenteTeste.js U  TesteState.js U  TesteStateFilho.js U X  App.js M

src > componentes > TesteStateFilho.js > ...
1  import React from 'react'
2
3  export default props=>{
4
5      return (
6          <div>
7              <p>Valor de State no filho: {props.valor}</p>
8              <button onClick={() => props.aumentar()}>Aumentar</button>
9          </div>
10     );
11 }
```

. . . + . . .

.

.

.



Usando State

45697056
■ ■ ■

Dentro de TesteState
devemos passar a função
aumentar como atributo.

```
}  
return(  
  <>  
    <p>ValorState: {valorState}</p>  
    <p>ValorVariavel: {valorVariavel}</p>  
    <TesteStateFilho valor={valorState} aumentar={aumentar}/>  
  </>  
)  
}
```





Exercício

45697056
■ ■ ■

1. Na aplicação chamada exercício2, Crie ao menos o valor de um parágrafo utilizando state em cada componente e um botão com uma função para alterar o valor dele;
2. Crie um novo componente chamado CaixaTeste.js, dentro uma div com um parágrafo. Insira este componente dentro da section criada no componente corpo.
3. Usando o conhecimento de props somado ao de state, faça com que o valor do parágrafo de CaixaTeste se atualize junto com o valor do seu componente pai.



DUVIDAS





Copyright © 2015 - 2021 Prof. Luís Carlos S. Silva
Prof. Alexandre Carlos de Jesus

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