



# FRONT-END DESIGN ENGINEERING





[aula 03]

[Atualizado em: 23/01/2024]



[React – Estilização]



Material cedido gentilmente pelo professor Alexandre Carlos ([profalexandre.jesus@fiap.com.br](mailto:profalexandre.jesus@fiap.com.br)) e professor Luís Carlos ([lsilva@fiap.com.br](mailto:lsilva@fiap.com.br)) e atualizado pelo professor Adriano Milanez ([profadriano.milanez@fiap.com.br](mailto:profadriano.milanez@fiap.com.br))



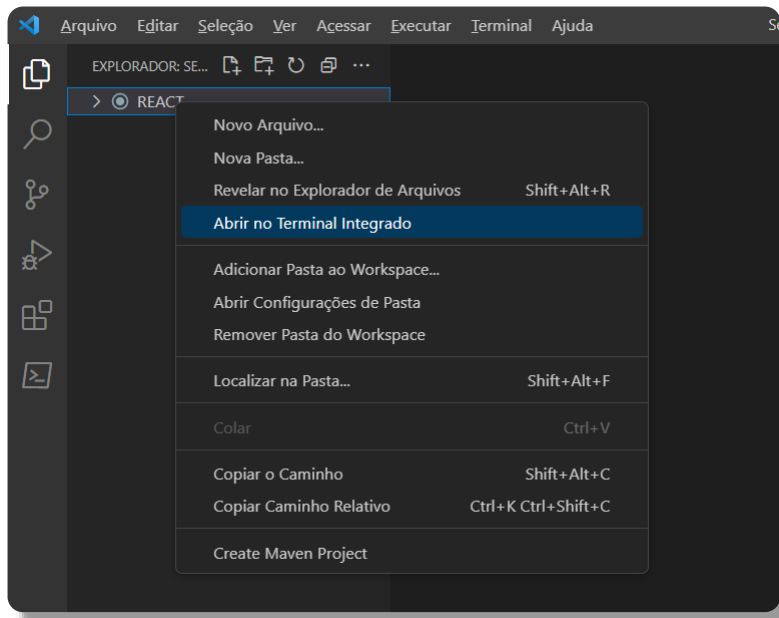
## REACT – ESTILIZAÇÃO

# REACT – ESTILIZAÇÃO

- ✓ A estilização no REACT acontece de forma muito semelhante ao que estamos acostumados no HTML.
- ✓ Podemos utilizar os três tipos de estilização:
  - ✓ Externo
  - ✓ Incorporado e
  - ✓ Inline

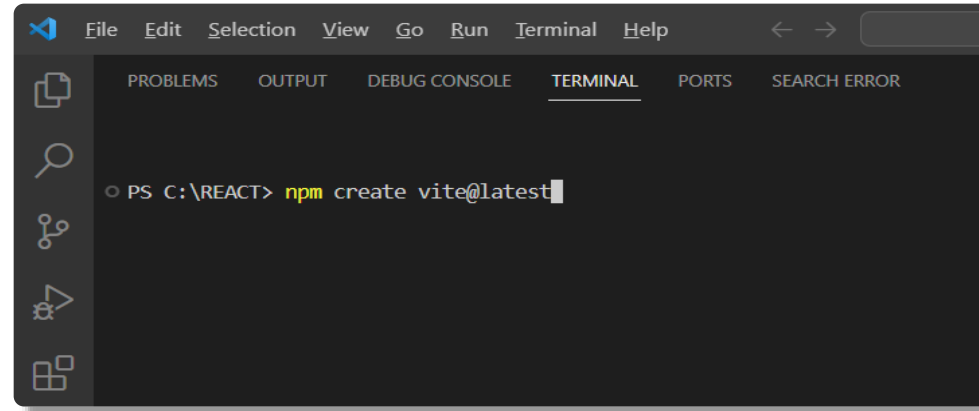
# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
  - ✓ Vamos abrir o VS Code.
  - ✓ Vamos criar uma pasta chamada “REACT” e com o botão direito, abrir o Terminal, diretamente nessa pasta



# REACT – ESTILIZAÇÃO

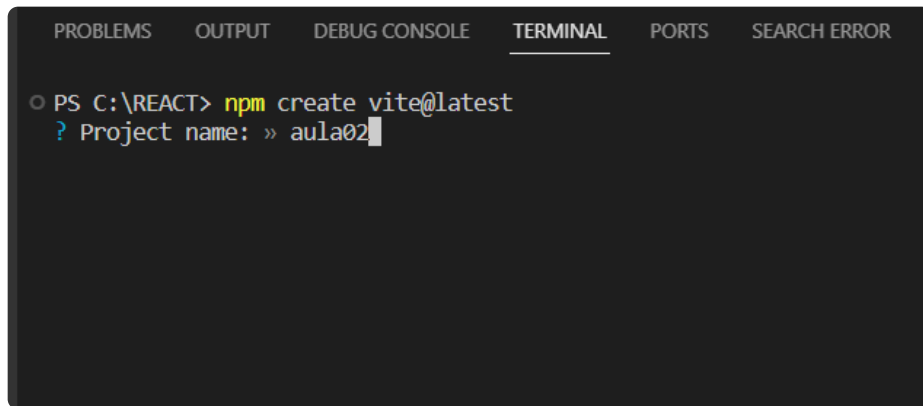
- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ No terminal aberto vamos digitar o seguinte:
  - ✓ `npm create vite@latest`



A screenshot of a Visual Studio Code terminal window. The terminal has a dark theme and shows the command prompt `PS C:\REACT>` followed by the command `npm create vite@latest` which has been entered. The terminal window includes a menu bar with options like File, Edit, Selection, View, Go, Run, Terminal, and Help. Below the menu bar are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL (which is active). On the left side of the terminal, there is a vertical toolbar with icons for file operations, search, and other development tools.

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ No terminal aberto vamos digitar o seguinte:
  - ✓ `npm create vite@latest`
  - ✓ Na sequência daremos um nome ao nosso projeto

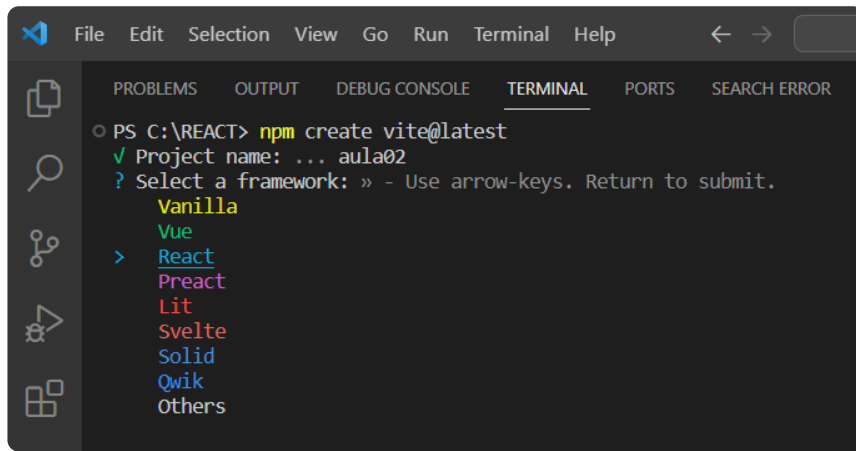


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

○ PS C:\REACT> npm create vite@latest
  ? Project name: » aula02
```

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ No terminal aberto vamos digitar o seguinte:
  - ✓ `npm create vite@latest`
  - ✓ Na sequência daremos um nome ao nosso projeto
  - ✓ Descendo com a seta de direção no teclado, selecionamos o framework (REACT) que vamos criar nosso projeto.



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS C:\REACT> npm create vite@latest
✓ Project name: ... aula02
? Select a framework: » - Use arrow-keys. Return to submit.
  Vanilla
  Vue
  > React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Others
```



# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

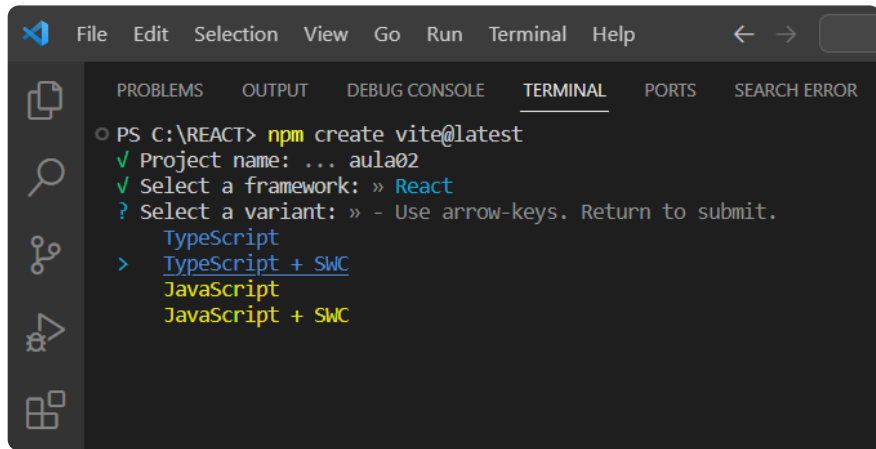
✓ No terminal aberto vamos digitar o seguinte:

✓ `npm create vite@latest`

✓ Na sequência daremos um nome ao nosso projeto

✓ Descendo com a seta de direção no teclado, selecionamos o framework (REACT) que vamos criar nosso projeto

✓ E na sequência o “linguagem” (TYPESCRIPT + SWC) que vamos trabalhar

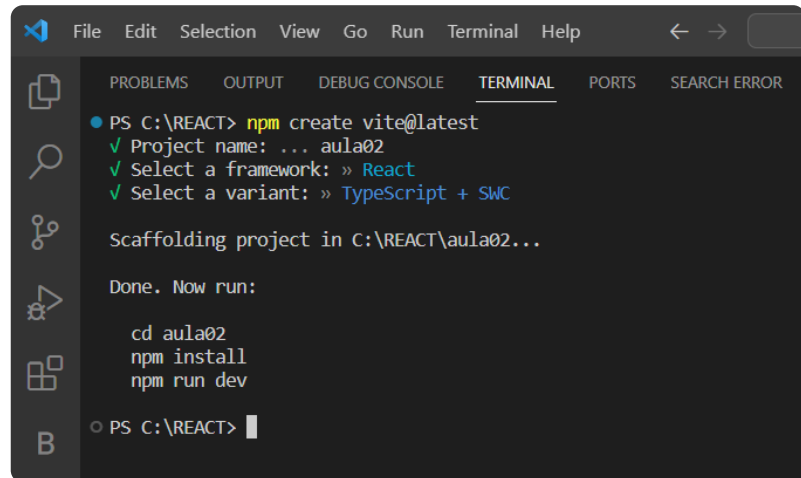


```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS C:\REACT> npm create vite@latest
✓ Project name: ... aula02
✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
>  TypeScript + SWC
   JavaScript
   JavaScript + SWC
```

# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, teremos a seguinte resposta.



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
• PS C:\REACT> npm create vite@latest
✓ Project name: ... aula02
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC

Scaffolding project in C:\REACT\aula02...

Done. Now run:

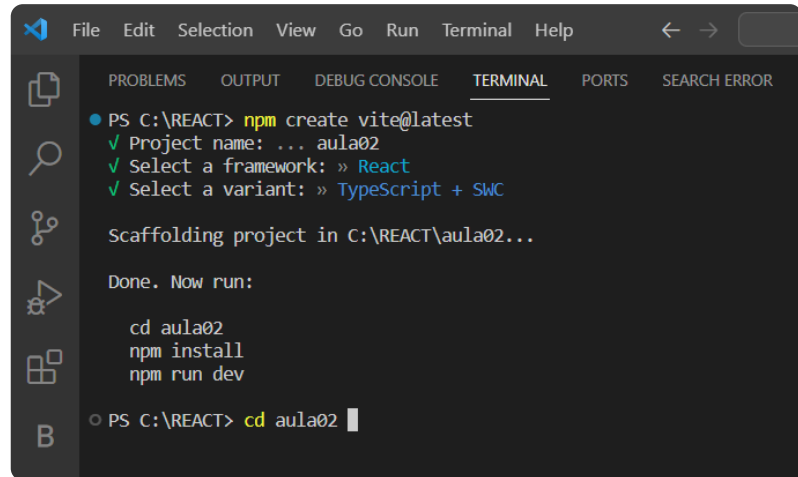
  cd aula02
  npm install
  npm run dev

○ PS C:\REACT> |
```

# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, termos a seguinte resposta.
- ✓ Como na imagem, vamos digitar o que se pede

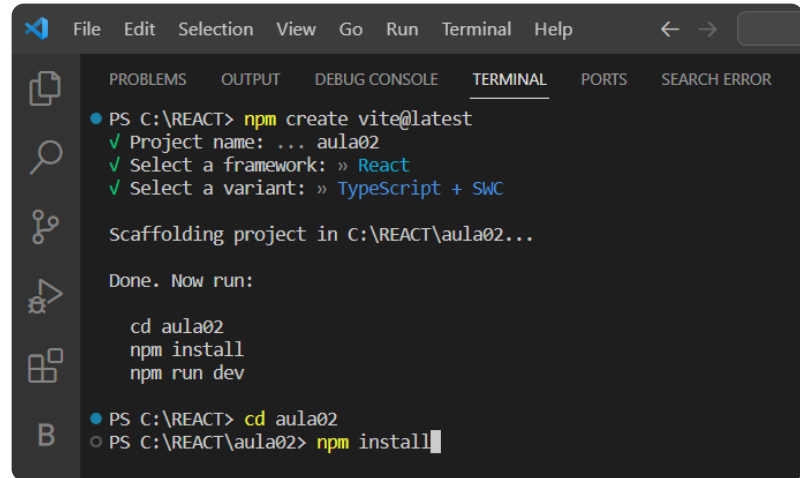


```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS C:\REACT> npm create vite@latest
✓ Project name: ... aula02
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC
Scaffolding project in C:\REACT\aula02...
Done. Now run:
  cd aula02
  npm install
  npm run dev
PS C:\REACT> cd aula02
```

# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, termos a seguinte resposta.
- ✓ Como na imagem, vamos digitar o que se pede



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS C:\REACT> npm create vite@latest
✓ Project name: ... aula02
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC
Scaffolding project in C:\REACT\aula02...

Done. Now run:

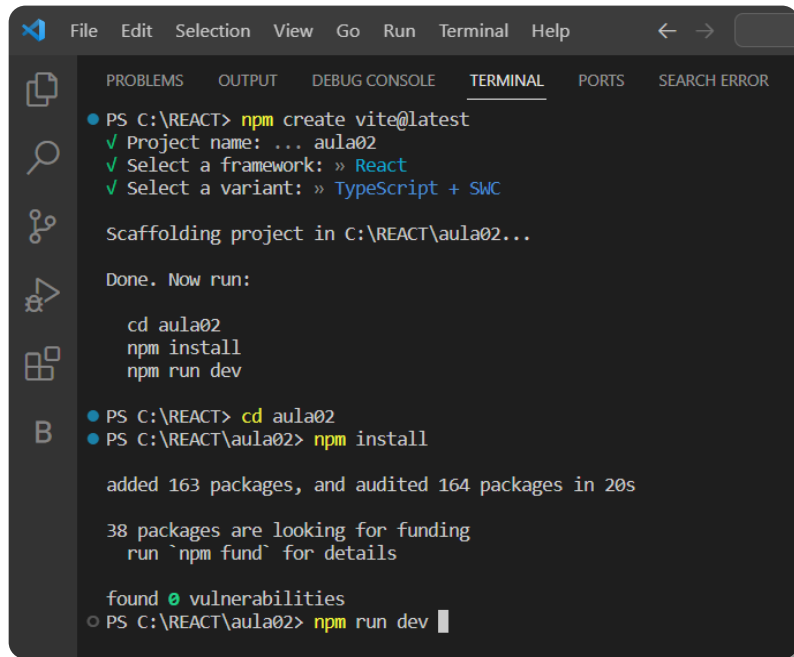
  cd aula02
  npm install
  npm run dev

PS C:\REACT> cd aula02
PS C:\REACT\aula02> npm install
```

# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, teremos a seguinte resposta.
- ✓ Como na imagem, vamos digitar o que se pede, veremos que foram adicionados em nosso projeto alguns pacotes
- ✓ Na sequência, digitamos à última instrução solicitada



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR

PS C:\REACT> npm create vite@latest
✓ Project name: ... aula02
✓ Select a framework: » React
✓ Select a variant: » TypeScript + SWC

Scaffolding project in C:\REACT\aula02...

Done. Now run:

  cd aula02
  npm install
  npm run dev

PS C:\REACT> cd aula02
PS C:\REACT\aula02> npm install

added 163 packages, and audited 164 packages in 20s

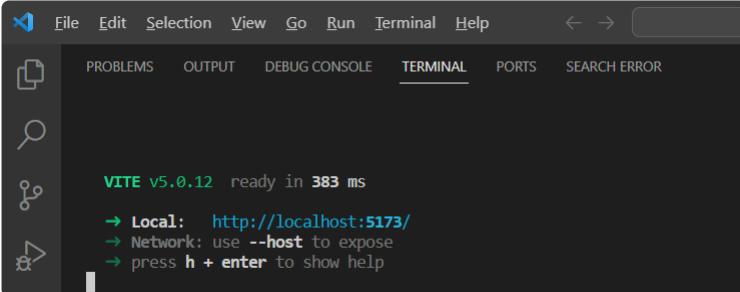
38 packages are looking for funding
run `npm fund` for details

found 0 vulnerabilities
PS C:\REACT\aula02> npm run dev
```

# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

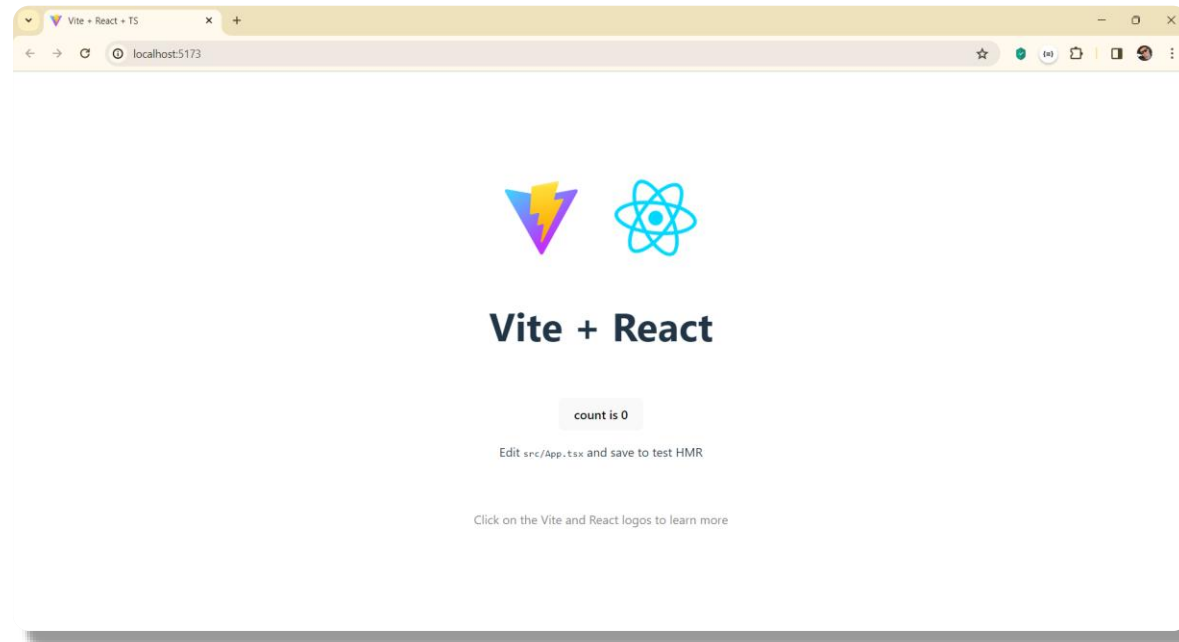
- ✓ No terminal aberto vamos digitar o seguinte:
- ✓ Após o término da criação, teremos a seguinte resposta.
- ✓ Como na imagem, vamos digitar o que se pede, veremos que foram adicionados em nosso projeto alguns pacotes
- ✓ Na sequência, digitamos a última instrução solicitada
- ✓ Pressionamos a tecla CTRL e clicamos na primeira opção.



```
File Edit Selection View Go Run Terminal Help  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR  
VITE v5.0.12 ready in 383 ms  
→ Local: http://localhost:5173/  
→ Network: use --host to expose  
→ press h + enter to show help
```

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ E no seu navegador deverá ser carregada a seguinte página.



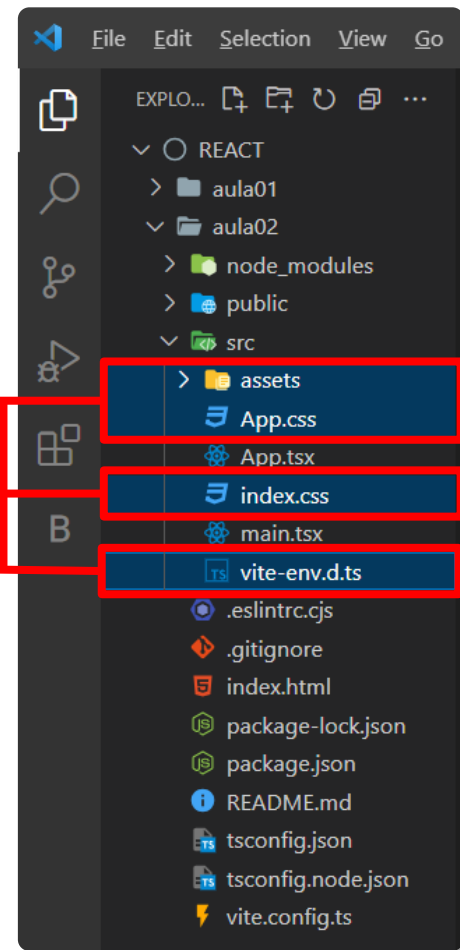
# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

- ✓ Vamos apagar alguns arquivos da pasta SRC:
  - ✓ Conteúdo da pasta ASSETS
  - ✓ Arquivos CSS (APP.CSS e INDEX.CSS)
  - ✓ Arquivo VITE-ENV.D.TS

Apagamos os arquivos selecionados

Lembrando que devemos agora reescrever nossos arquivos MAIN.TSX e modificar o arquivo APP.TSX

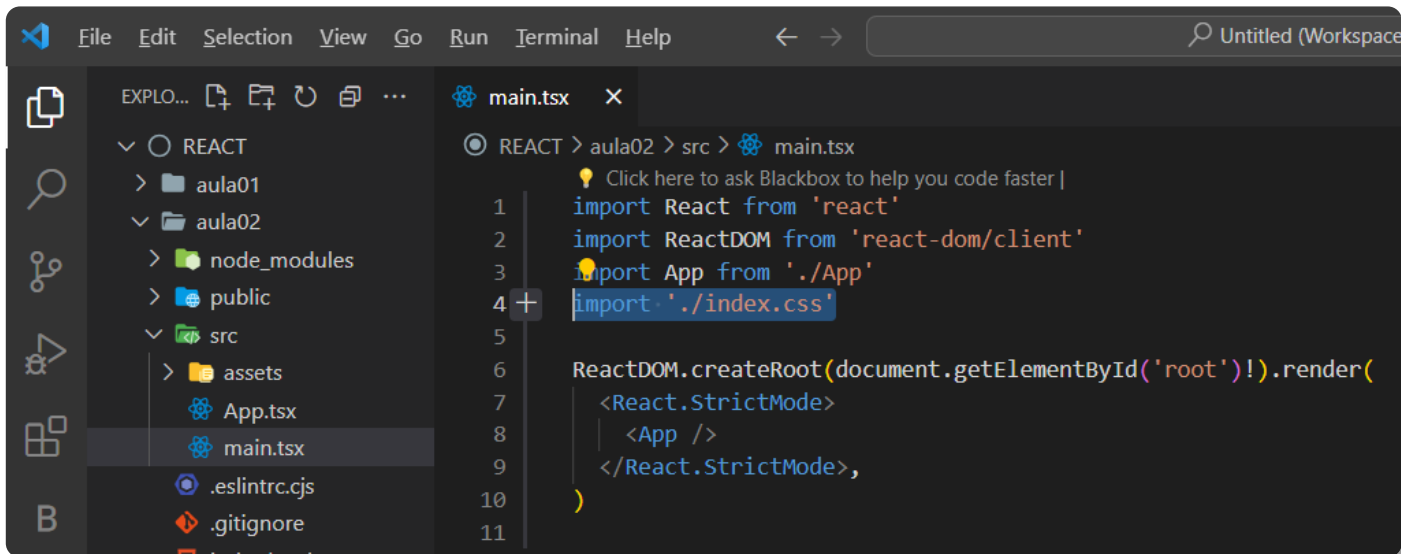




# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

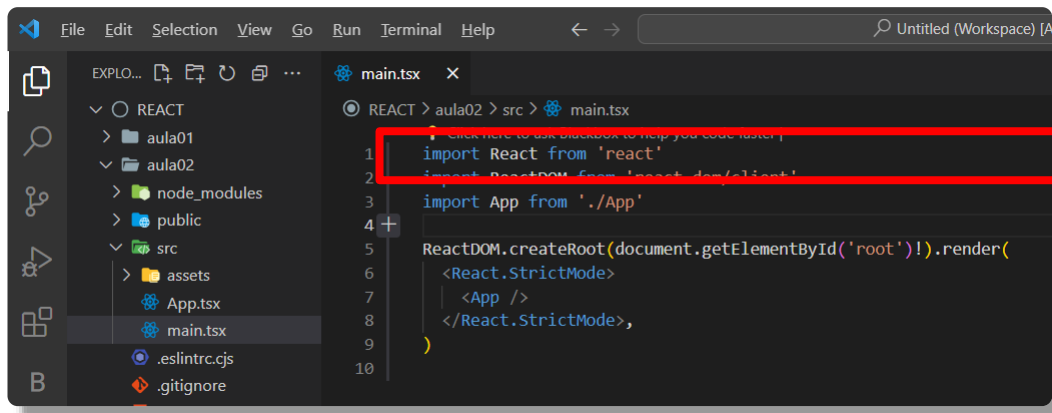
- ✓ No arquivo MAIN.TSX, vamos apenas excluir a linha 4, pois nessa linha existe a chamada para o arquivo INDEX.CSS que acabamos de excluir.



```
File Edit Selection View Go Run Terminal Help
EXPLO... main.tsx
  ○ REACT
    > aula01
    > aula02
    > node_modules
    > public
    > src
      > assets
      > App.tsx
      > main.tsx
      > .eslintrc.cjs
      > .gitignore
      > index.html
1  import React from 'react'
2  import ReactDOM from 'react-dom/client'
3  import App from './App'
4  import './index.css'
5
6  ReactDOM.createRoot(document.getElementById('root')).render(
7    <React.StrictMode>
8      <App />
9    </React.StrictMode>,
10 )
11
```

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ Ainda no arquivo MAIN.TSX, vamos analisar as demais linhas.

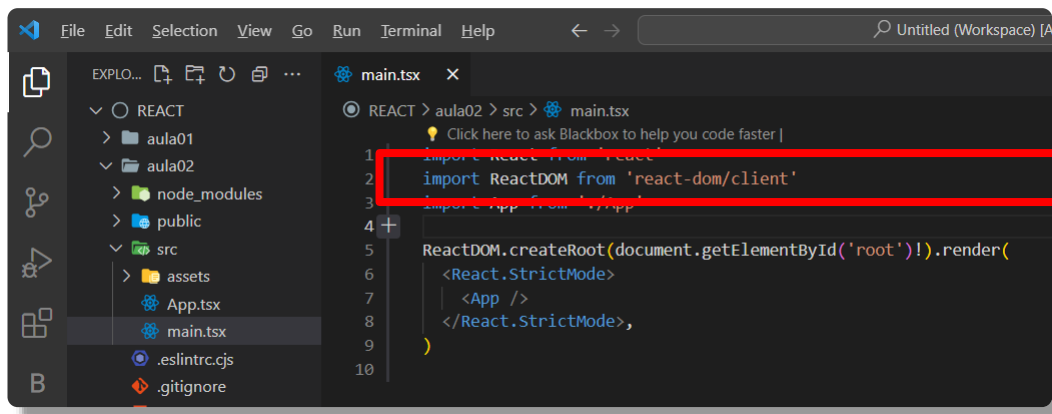


```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App'
4
5 ReactDOM.createRoot(document.getElementById('root')!).render(
6   <React.StrictMode>
7     <App />
8   </React.StrictMode>,
9 )
10
```

Nos permite usar o JSX

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ Ainda no arquivo MAIN.TSX, vamos analisar as demais linhas.



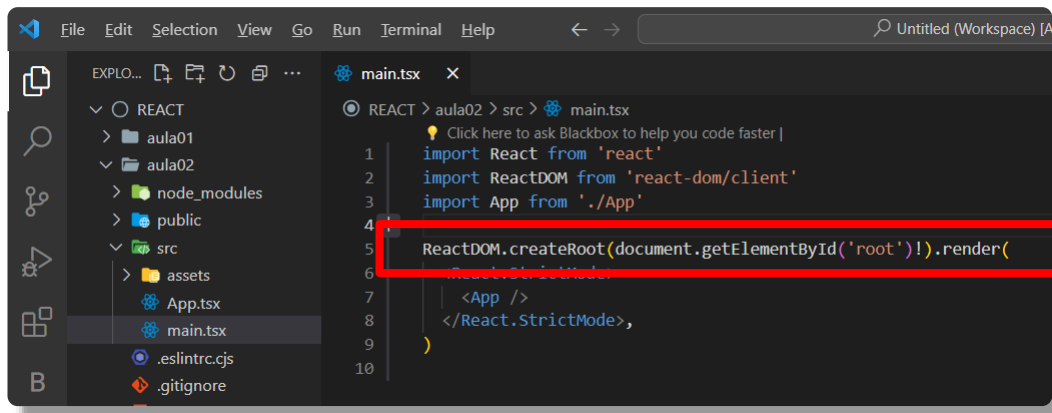
```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App'
4
5 ReactDOM.createRoot(document.getElementById('root')!).render(
6   <React.StrictMode>
7     <App />
8   </React.StrictMode>,
9 )
10
```

Nos permite usar VDOM<sup>1</sup>

<sup>1</sup> O virtual DOM (VDOM) é um conceito de programação onde uma representação ideal, ou “virtual”, da interface do usuário é mantida em memória e sincronizada com o DOM “real” por uma biblioteca como o ReactDOM. Esse processo é chamado de reconciliação.

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ Ainda no arquivo MAIN.TSX, vamos analisar as demais linhas.

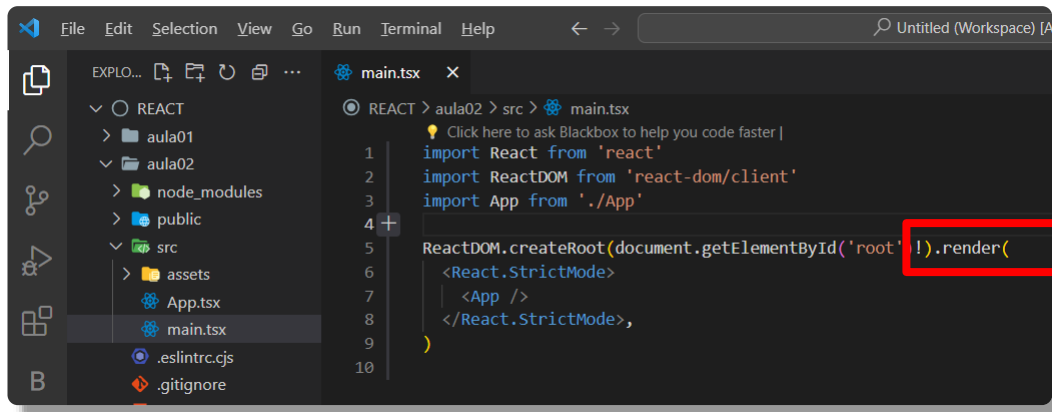


```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App'
4
5 ReactDOM.createRoot(document.getElementById('root')!).render(
6   <React.StrictMode>
7     <App />
8   </React.StrictMode>,
9 )
10
```

Local onde o conteúdo será renderizado

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ Ainda no arquivo MAIN.TSX, vamos analisar as demais linhas.



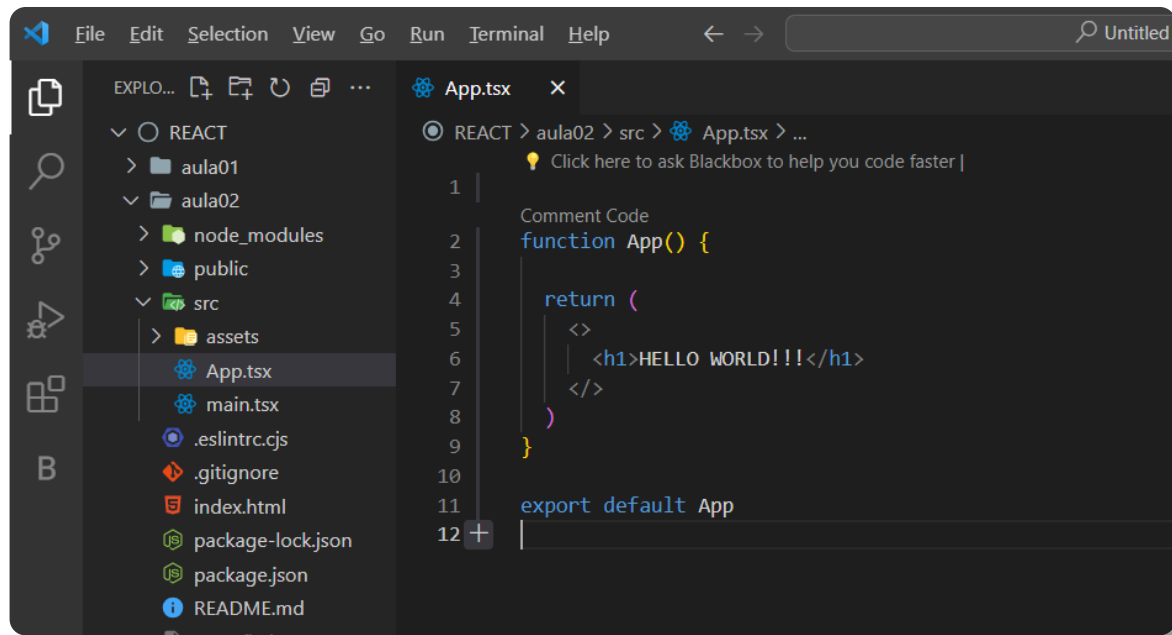
```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App'
4
5 ReactDOM.createRoot(document.getElementById('root')!).render(
6   <React.StrictMode>
7     <App />
8   </React.StrictMode>,
9 )
10
```

Método para renderizar componentes na tela

# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

- ✓ O arquivo APP.TSX, vamos reescrever como abaixo.



The screenshot shows a code editor with a dark theme. On the left, the Explorer sidebar displays a file tree for a project named 'REACT'. The tree structure is as follows:

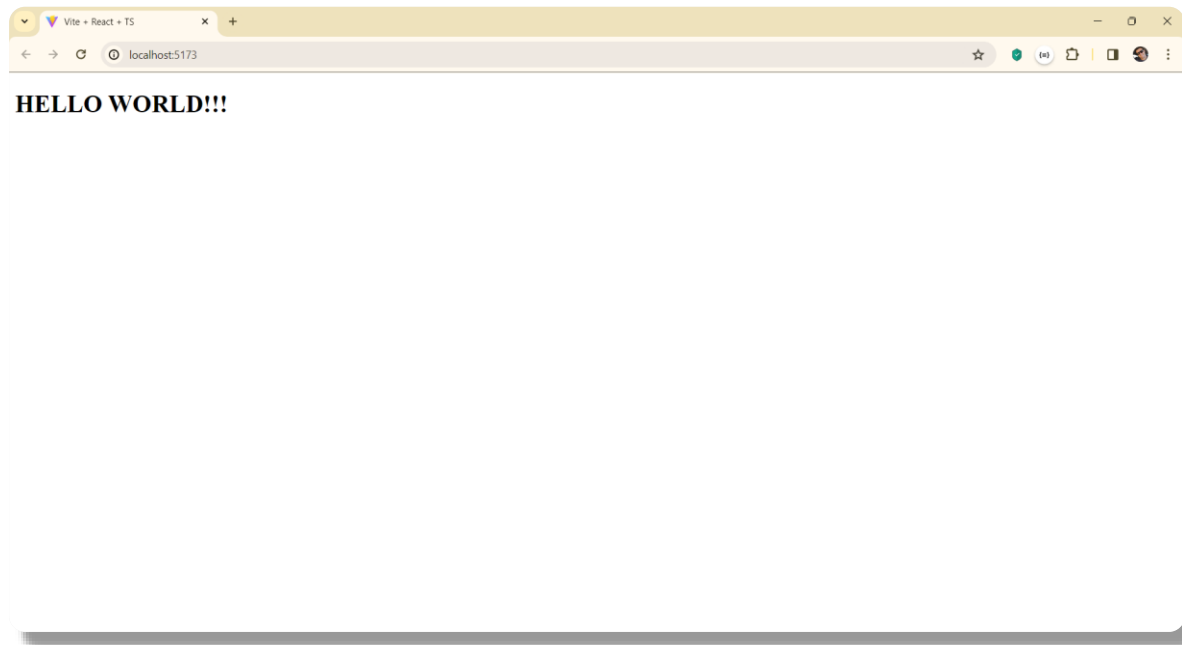
- REACT
  - aula01
  - aula02
    - node\_modules
    - public
    - src
      - assets
        - App.tsx** (selected)
        - main.tsx
      - .eslintrc.cjs
      - .gitignore
      - index.html
      - package-lock.json
      - package.json
      - README.md

The main editor area shows the content of 'App.tsx'. The code is as follows:

```
1 |  
2 |  
3 |  
4 |   return (  
5 |     <>  
6 |     <h1>HELLO WORLD!!!</h1>  
7 |     </>  
8 |   )  
9 | }  
10 |  
11 | export default App  
12 |
```

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ E no seu navegador deverá ser carregada a seguinte página.



# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

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

```
1  
2  
3 export default () =>{  
4   return (  
5     <>  
6     <h1>HELLO WORLD!!!</h1>  
7     </>  
8   )  
9 }  
10 +
```

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

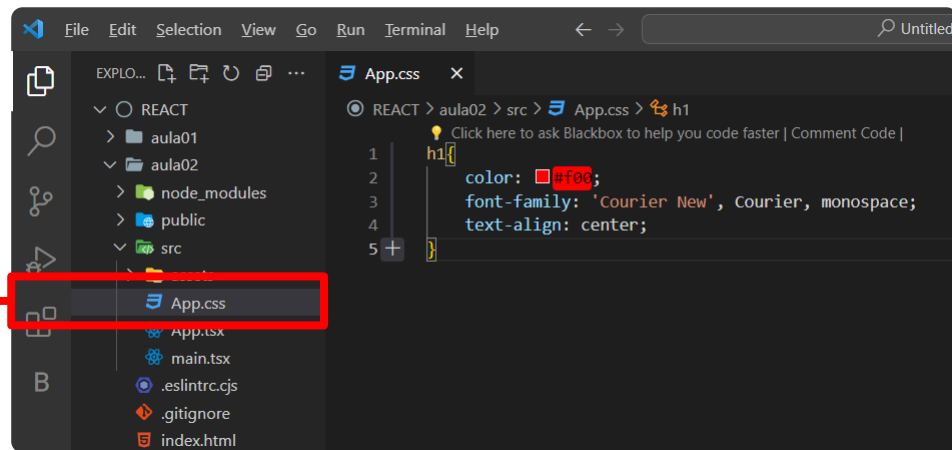


# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

- ✓ 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:

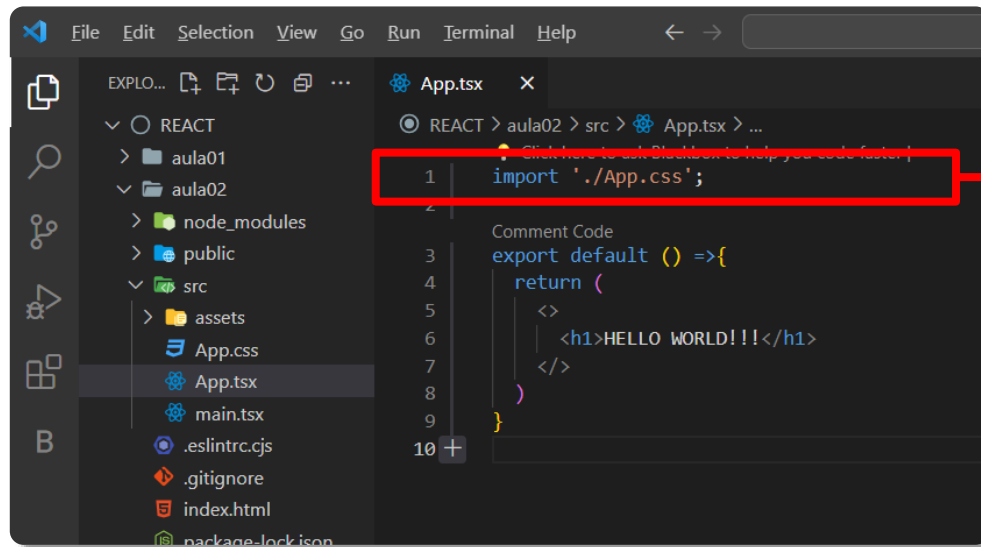
Por boa prática, colocamos os nomes dos arquivos CSS iguais aos dos componentes.



# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

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

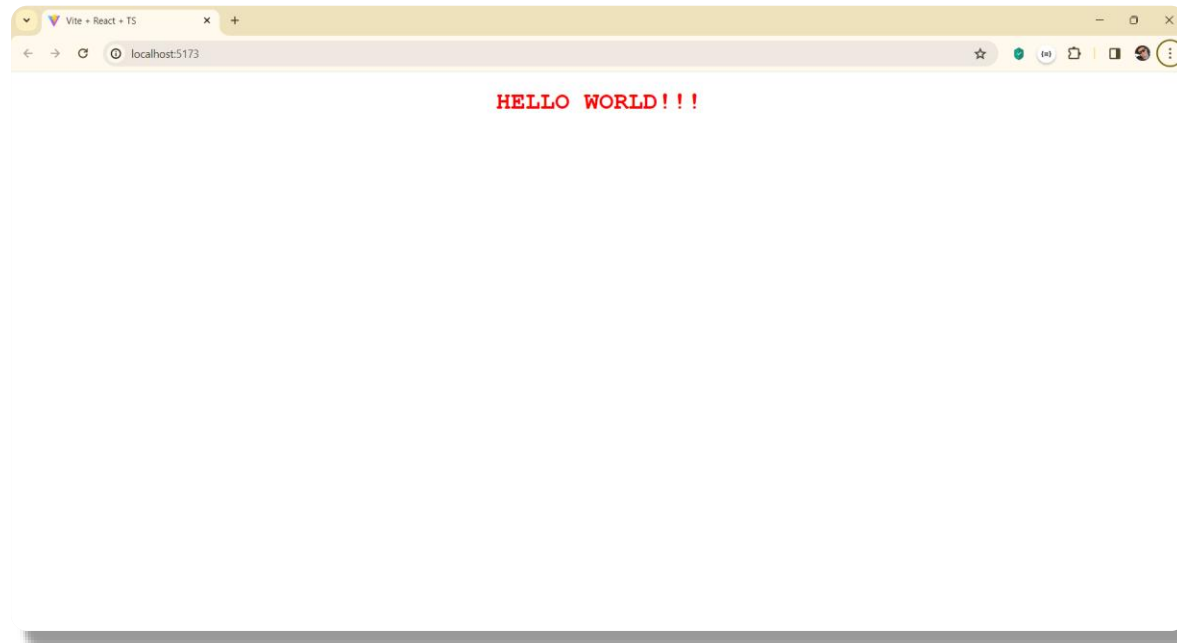


```
1 import './App.css';
2
3 export default () =>{
4   return (
5     <>
6     <h1>HELLO WORLD!!!</h1>
7     </>
8   )
9 }
10 +
```

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

# REACT – ESTILIZAÇÃO

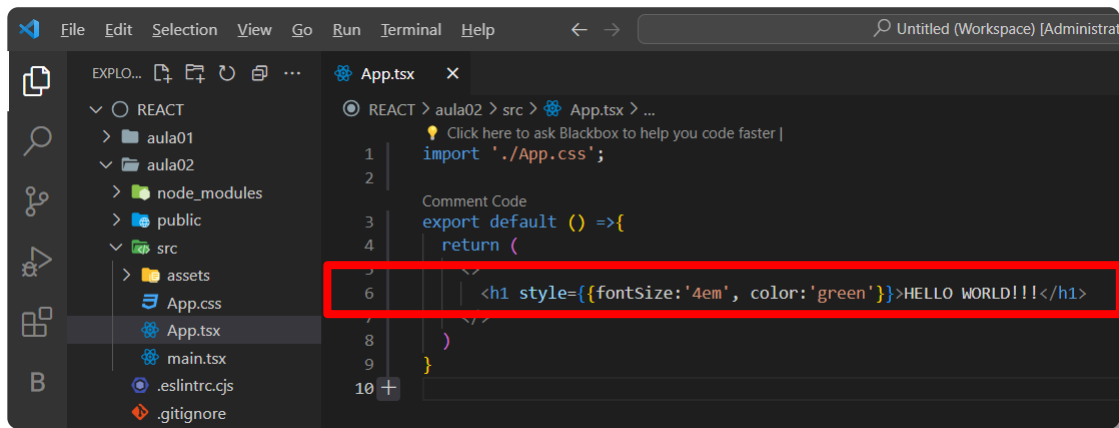
- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ E no seu navegador deverá ser carregada a seguinte página.



# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

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



```
1 import './App.css';
2
3 export default () =>{
4   return (
5     <h1 style={{fontSize:'4em', color:'green'}}>HELLO WORLD!!!</h1>
6   )
7 }
8
9
10
```

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

# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

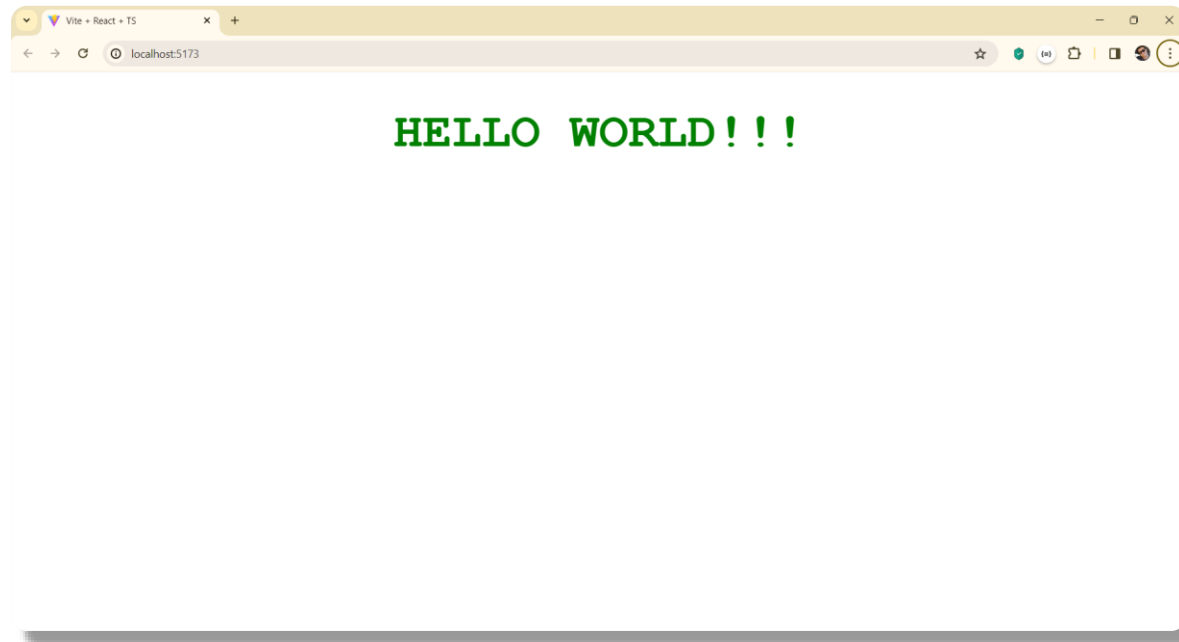
- ✓ Para inserirmos valores de forma **inline** no elemento devemos nos atentar a pequenas diferenças:

```
1 import React from 'react';
2 import './App.css'
3
4
5 export default () =>{
6   return (
7     <h1 style={{fontSize: '4em', color: 'red'}}>Conteúdo de App.js</h1>
8   )
9 }
10
```

Para separar as propriedades devemos usar a virgula.

# REACT – ESTILIZAÇÃO

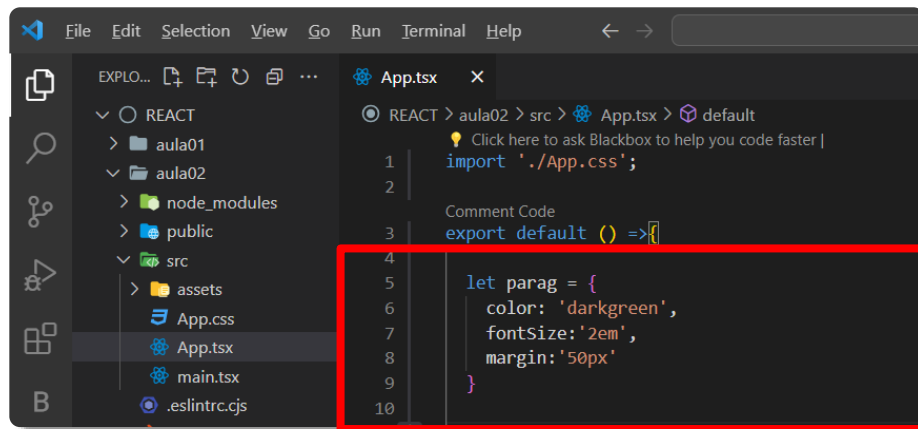
- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ E no seu navegador deverá ser carregada a seguinte página.



# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

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



```
1 import './App.css';
2
3 export default () => {
4
5   let parag = {
6     color: 'darkgreen',
7     fontSize: '2em',
8     margin: '50px'
9   }
10 }
```

Criando um objeto chamado **parag** e inserindo as propriedades.

# REACT – ESTILIZAÇÃO

## ✓ Vamos criar nossa segunda aplicação REACT

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

Desta vez para inserir  
usamos chaves simples.

```
Run Terminal Help  Untitled (Workspace) [Administrator]

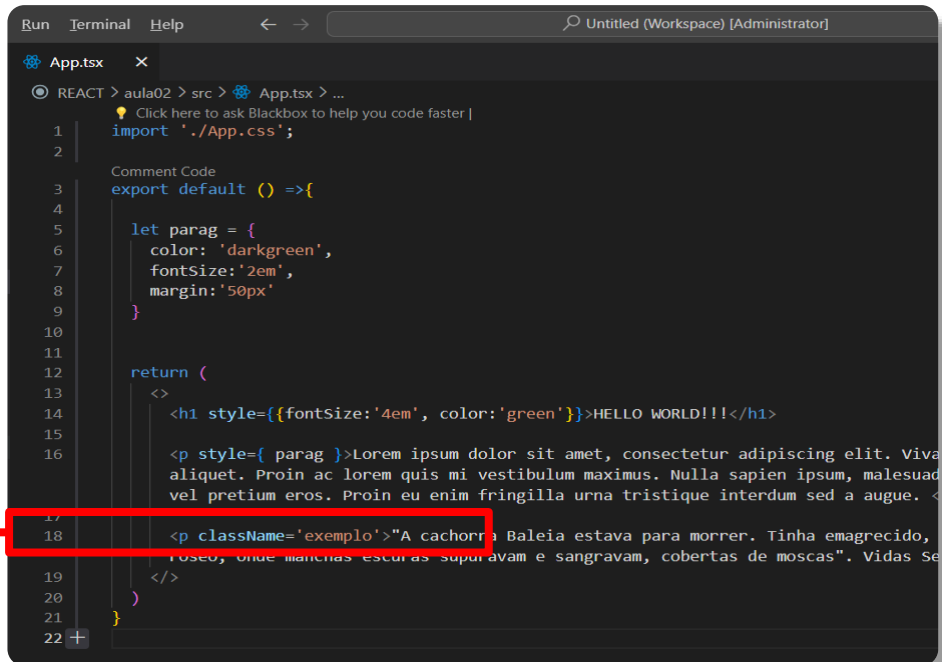
App.tsx x
REACT > aula02 > src > App.tsx > default
Click here to ask Blackbox to help you code faster |
1 import './App.css';
2
Comment Code
3 export default () => {
4
5   let parag = {
6     color: 'darkgreen',
7     fontSize: '2em',
8     margin: '50px'
9   }
10
11 +
12   return (
13     <>
14       <h1 style={{fontSize:'4em', color:'green'}}>HELLO WORLD!!!</h1>
15       <p style={ parag }>Lorem ipsum dolor sit amet, consectetur adipiscing elit,
16       aliquet. Proin ac iorem quis mi vestibulum maximus. Nulla sapien ipsum, ma
17       vel pretium eros. Proin eu enim fringilla urna tristique interdum sed a aug
18     </>
19   )
20 }
```



# REACT – ESTILIZAÇÃO

- ✓ Vamos criar nossa segunda aplicação REACT
- ✓ Outro detalhe importante é que, para inserir uma classe no elemento devemos usar **className** ou invés de **class**:

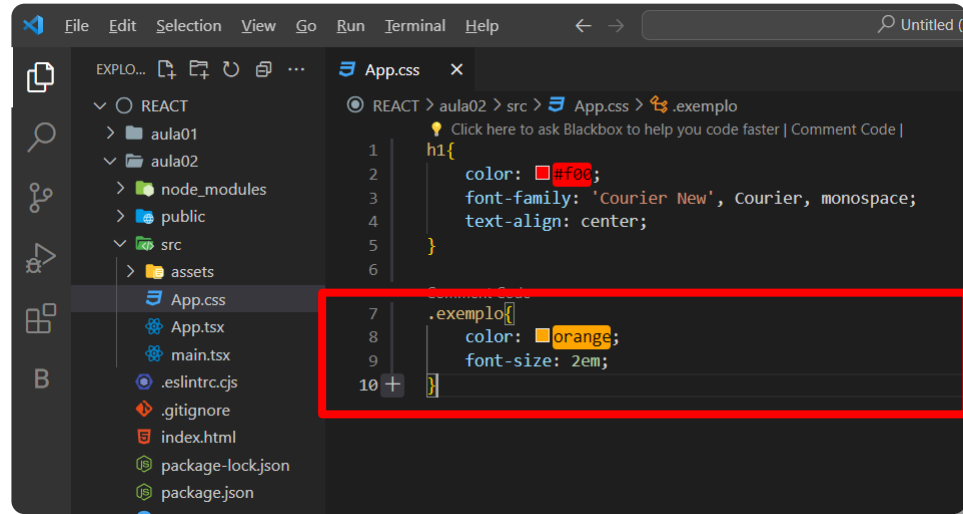
Usando o atributo **className**  
com o valor exemplo



```
Run Terminal Help ← → Untitled (Workspace) [Administrator]
App.tsx ×
REACT > aula02 > src > App.tsx > ...
Click here to ask Blackbox to help you code faster |
1 import './App.css';
2
3 Comment Code
4 export default () =>{
5
6   let parag = {
7     color: 'darkgreen',
8     fontSize: '2em',
9     margin: '50px'
10  }
11
12  return (
13    <>
14      <h1 style={{fontSize:'4em', color:'green'}}>HELLO WORLD!!!</h1>
15
16      <p style={ parag }>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Viva
17      aliquet. Proin ac lorem quis mi vestibulum maximus. Nulla sapien ipsum, malesuad
18      vel pretium eros. Proin eu enim fringilla urna tristique interdum sed a augue. <
19
20      <p className='exemplo'>"A cachorra Baleia estava para morrer. Tinha emagrecido,
21      roseo, onde manchas escuras supuravam e sangravam, cobertas de moscas". Vidas Se
22    </>
23  )
24 }
```

# REACT – ESTILIZAÇÃO

- ✓ **CSS em componentes**
- ✓ Criando a formatação de exemplo.



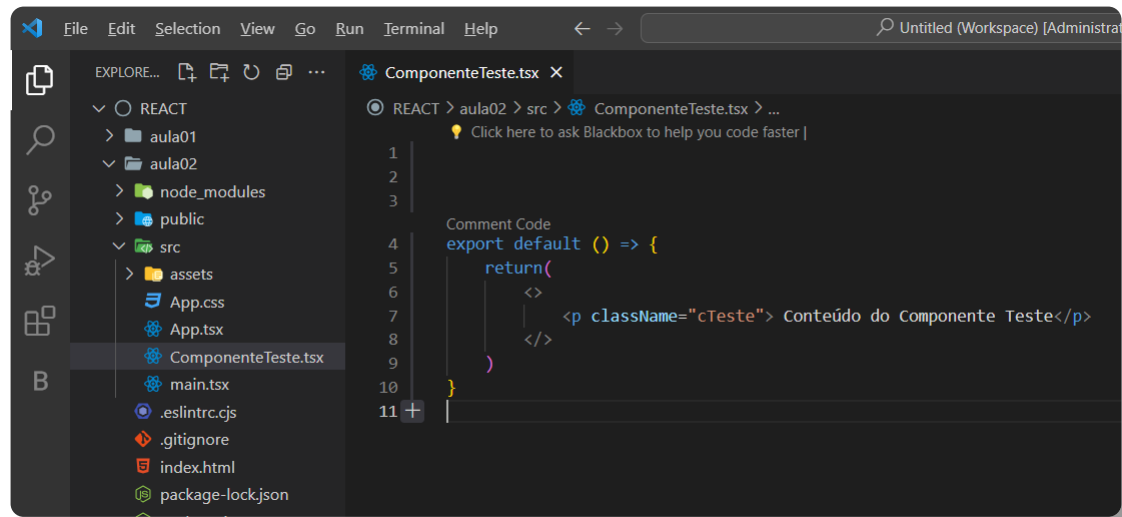
# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ E no seu navegador deverá ser carregada a seguinte página.



# REACT – ESTILIZAÇÃO

- ✓ **CSS em componentes**
- ✓ O componente também pode receber a estilização quando for inserido no componente pai através de seu arquivo CSS.  
Crie chamado **ComponenteTeste.tsx** e insira o código abaixo:

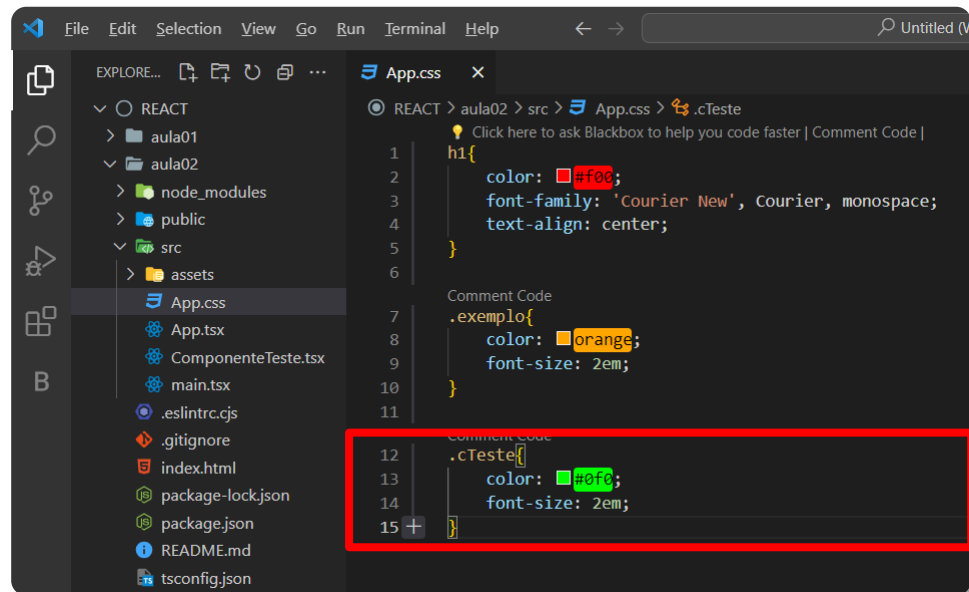


The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORE' sidebar shows a file tree for a project named 'REACT'. The tree includes folders 'aula01' and 'aula02', and a 'src' folder containing 'assets', 'App.css', 'App.tsx', 'ComponenteTeste.tsx' (selected), 'main.tsx', and configuration files like '.eslintrc.cjs', '.gitignore', 'index.html', and 'package-lock.json'. The main editor area displays the content of 'ComponenteTeste.tsx'. The code is as follows:

```
1  
2  
3  
4 export default () => {  
5   return(  
6     <>  
7       <p className="cTeste"> Conteúdo do Componente Teste</p>  
8     </>  
9   )  
10 }  
11
```

# REACT – ESTILIZAÇÃO

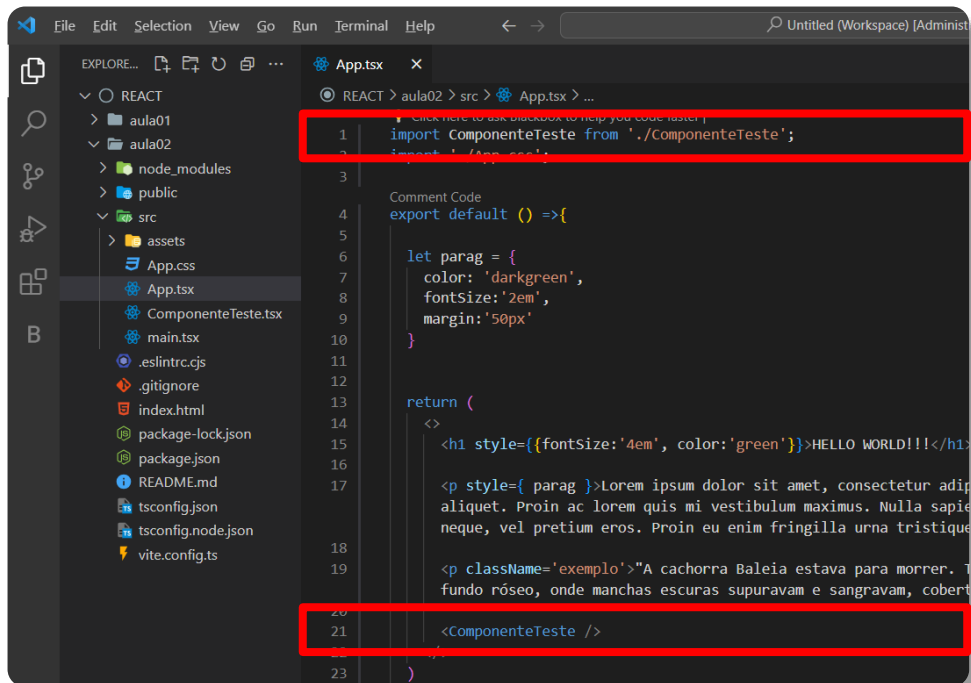
- ✓ **CSS em componentes**
- ✓ No arquivo App.css, vamos criar as propriedades de estilização da classe cTeste:



```
File Edit Selection View Go Run Terminal Help
App.css
REACT > aula02 > src > App.css > .cTeste
Click here to ask Blackbox to help you code faster | Comment Code |
1  h1{
2    color: #f00;
3    font-family: 'Courier New', Courier, monospace;
4    text-align: center;
5  }
6
7  .exemplo{
8    color: orange;
9    font-size: 2em;
10 }
11
12 .cTeste{
13   color: #0f0;
14   font-size: 2em;
15 }
```

# REACT – ESTILIZAÇÃO

- ✓ CSS em componentes
- ✓ Agora é só chamar o componente no App.tsx:



```
1 import ComponenteTeste from './ComponenteTeste';
2 import './App.css';
3
4 export default () =>{
5
6   let parag = {
7     color: 'darkgreen',
8     fontSize: '2em',
9     margin: '50px'
10  }
11
12  return (
13    <>
14      <h1 style={{fontSize: '4em', color: 'green'}}>HELLO WORLD!!!</h1>
15
16      <p style={parag}>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin ac lorem quis mi vestibulum maximus. Nulla sapien-
17        neque, vel pretium eros. Proin eu enim fringilla urna tristique
18
19      <p className='exemplo'>"A cachorra Baleia estava para morrer. T-
20        fundo róseo, onde manchas escuras supuravam e sangravam, cobert
21
22      <ComponenteTeste />
23    </>
24  )
25 }
```

Importamos o componente.

Em seguida o utilizamos

# REACT – ESTILIZAÇÃO

- ✓ **Vamos criar nossa segunda aplicação REACT**
- ✓ E no seu navegador deverá ser carregada a seguinte página.





## REACT – ARROW FUNCTION

LEVEL – ARROW FUNCTION



# REACT – ARROW FUNCTION

## ✓ **ARROW FUNCTION**

- ✓ É uma forma concisa e simplificada de escrever funções em JavaScript.
- ✓ São muito utilizadas em React para definir métodos em classes e também como funções dentro de componentes.
- ✓ Em vez de escrever uma função da forma tradicional utilizando a palavra-chave "function", uma arrow function é definida com uma sintaxe mais simples, utilizando a seta "=>" para indicar o corpo da função. Por exemplo, ao invés de escrever uma função da seguinte forma:

# REACT – ARROW FUNCTION

## ✓ ARROW FUNCTION

### Forma Tradicional

```
1 function minhaFuncao (argumento) {  
2     // set de instruções  
3     return resultado;  
4 }  
5
```

### Arrow Function

```
1 const minhaFuncao = (argumento) => {  
2     // set de instruções  
3     return resultado;  
4 }  
5
```

# REACT – ARROW FUNCTION

## ✓ ARROW FUNCTION

- ✓ Além disso, quando a função tem apenas uma expressão, é possível simplificar ainda mais a sintaxe, removendo as chaves e a palavra-chave "return":

```
1  const minhaFuncaoSimples = (argumento) => resultado
```

# REACT – ARROW FUNCTION

## ✓ **ARROW FUNCTION**

### ✓ Quando devemos usá-las

- ✓ As arrow functions têm seu momento de glória em tudo aquilo que exija que `this` esteja vinculado ao contexto, e não à própria função.
- ✓ Apesar de serem anônimas, podemos usá-las com métodos como `map` e `reduce`, pois tornam seu código mais legível.

# REACT – ARROW FUNCTION

## ✓ ARROW FUNCTION

### ✓ Quando NÃO devemos usá-las

#### ✓ Métodos de objetos

- ✓ Ao chamar *gato.saltar* abaixo, o número de vidas não diminui. Isso ocorre porque *this* não está vinculado a nada, herdando o valor de *this* de seu escopo pai.

#### ✓ Funções de callback com contexto dinâmico

- ✓ Se você precisa de que seu contexto seja dinâmico, as *arrow functions* não são a escolha certa. Vejamos este manipulador de evento a seguir:  

```
var button = document.getElementById('press');  
  
    button.addEventListener('click', () => {  
        this.classList.toggle('on');  
    });
```
- ✓ Se você clicar no botão (button), verá um *TypeError*. Isso ocorre porque *this* não está vinculado ao botão, mas ao escopo pai.

### ✓ Quando elas tornarem seu código menos legível

- ✓ Vale a pena levar em consideração a variedade de sintaxes que tratamos aqui. Com as funções regulares, as pessoas sabem o que devem esperar. Com as *arrow functions*, pode ser difícil decifrar o que você está vendo de maneira direta.

# REACT – EXERCÍCIOS

## ✓ Exercício 01 - Estilização em React: Construindo um Card Interativo

- ✓ Você foi designado para criar um componente React que represente um card interativo. Este card deve exibir informações fictícias sobre um produto e ter interações visuais quando o usuário passar o mouse sobre ele.
- ✓ Requisitos:
  - ✓ Crie um componente funcional React chamado InteractiveCard.
  - ✓ O card deve exibir as seguintes informações fictícias:
    - ✓ Nome do produto: "Smartphone XYZ"
    - ✓ Preço: R\$ 999,99
    - ✓ Descrição: "O smartphone XYZ é repleto de recursos incríveis para atender às suas necessidades diárias."
    - ✓ Estilize o card de forma agradável e responsiva usando CSS-in-JS.
    - ✓ Adicione uma animação de transição suave para as interações do mouse. Quando o usuário passar o mouse sobre o card, as cores de fundo e texto devem mudar gradualmente.
    - ✓ Certifique-se de que o card seja visualmente atraente e que as informações sejam legíveis.
    - ✓ Dicas:
      - ✓ Utilize as propriedades de estilo do React para aplicar estilos diretamente ao componente.
      - ✓ Considere a utilização de propriedades de estado para controlar as interações do mouse.
- ✓ Teste diferentes combinações de cores, sombras e tamanhos para criar um design atraente.

# REFERÊNCIAS BIBLIOGRÁFICAS

ANTONIO, C. Pro React: Build Complex Front-End Applications in a Composable Way With React. Apress, 2015.

BOSWELL, D; FOUCHER, T. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code. Estados Unidos: O'Reilly Media, 2012.

BRITO, Robin Cris. Android Com Android Studio - Passo A Passo. Editora Ciência Moderna.

BUNA, S. React Succinctly. Estados Unidos: [s.n], 2016. Disponível em: <[www.syncfusion.com/ebooks/reactjs\\_succinctly](http://www.syncfusion.com/ebooks/reactjs_succinctly)>. Acesso em: 12 de janeiro de 2023.

FACEBOOK (2019a). React: Getting Started. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-api.html](https://reactjs.org/docs/react-api.html)>. Acesso em: 13 de janeiro de 2023.

FACEBOOK (2019b). React Without ES6. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-without-es6.html](https://reactjs.org/docs/react-without-es6.html)>. Acesso em: 10 de janeiro de 2023.

FACEBOOK (2019c). React Without JSX. React Docs, 2019. Disponível em: <[reactjs.org/docs/react-without-jsx.html](https://reactjs.org/docs/react-without-jsx.html)>. Acesso em: 10 de janeiro de 2023.

FREEMAN, Eric ROBSON, Elisabeth. Use a Cabeça! Programação em HTML5. Rio de Janeiro: Editora Alta Books, 2014

GACKENHEIMER, C. Introduction to React: Using React to Build scalable and efficient user interfaces.[s.i.]: Apress, 2015.

HUDSON, P. Hacking with React. 2016. Disponível em: <[www.hackingwithreact.com/read/1/3/introduction-to-jsx](http://www.hackingwithreact.com/read/1/3/introduction-to-jsx)>. Acesso em: 13 janeiro de 2023.

KOSTRZEWA, D. Is React.js the Best JavaScript Framework in 2018? 2018. Disponível em: <[hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8](https://hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8)>. Acesso em: janeiro de 2023.

MARTIN, R. Clean Code: A Handbook of Agile Software Craftsmanship. Estados Unidos: Prentice Hall, 2009.

MDN WEB DOCS. Guia JavaScript. Disponível em <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>>. Acessado em 29 de janeiro de 2023.

# REFERÊNCIAS BIBLIOGRÁFICAS

NELSON, J. Learn React's Fundamentals Without the Buzzwords? 2018. Disponível em: <[jamesknelson.com/learn-react-fundamentals-sans-buzzwords](https://jamesknelson.com/learn-react-fundamentals-sans-buzzwords)>. Acesso em: 12 janeiro de 2023.

NIELSEN, J. Response Times: The 3 Important Limits. 1993. Disponível em: <[www.nngroup.com/articles/response-times-3-important-limits](https://www.nngroup.com/articles/response-times-3-important-limits)>. Acesso em: 10 janeiro de 2023.

O'REILLY, T. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: <[www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap](https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap)>. Acesso em: 10 de janeiro de 2023.

PANDIT, N. What Is ReactJS and Why Should We Use It? 2018. Disponível em: <[www.c-sharpcorner.com/article/what-and-why-reactjs](https://www.c-sharpcorner.com/article/what-and-why-reactjs)>. Acesso em: 12 de janeiro de 2023.

RAUSCHMAYER, A. Speaking JavaScript: An In-Depth Guide for Programmers. Estados Unidos: O'Reilly Media, 2014.

REACTIVA. O arquivo package-lock.json. Disponível em: <<https://nodejs.reativa.dev/0020-package-lock-json/index>>. Acessado em 13 de janeiro de 2023.

\_\_\_\_\_. O guia do package.json. Disponível em: <<https://nodejs.reativa.dev/0019-package-json/index>>. Acessado em 13 de janeiro de 2023.

RICOY, L. Desmitificando React: Uma Reflexão para Iniciantes. 2018. Disponível em: <[medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114](https://medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114)>. Acesso em: 13 janeiro de 2023.

SILVA, Maurício Samy. Ajax com jQuery: requisições Ajax com a simplicidade de jQuery. São Paulo: Novatec Editora, 2009.

\_\_\_\_\_. Construindo Sites com CSS e XHTML. Sites Controlados por Folhas de Estilo em Cascata. São Paulo: Novatec, 2010.

\_\_\_\_\_. CSS3 - Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS. São Paulo: Novatec Editora, 2010.

STACKOVERFLOW. Most Popular Technologies: Web Frameworks. Developer Survey Results, StackOverflow, 2019. Disponível em: <[insights.stackoverflow.com/survey/2019#technology](https://insights.stackoverflow.com/survey/2019#technology)>. Acesso em: 13 de janeiro de 2023.



# REFERÊNCIAS BIBLIOGRÁFICAS

W3C. HTML5 - A linguagem de marcação que revolucionou a web. São Paulo: Novatec Editora, 2010.

\_\_\_\_\_. A vocabulary and associated APIs for HTML and XHTML. Disponível em <<https://www.w3.org/TR/2018/SPSD-html5-20180327/>>. Acessado em 28 de abril de 2020, às 20h53min.

\_\_\_\_\_. Cascading Style Sheets, level 1. Disponível em <<https://www.w3.org/TR/2018/SPSD-CSS1-20180913/>>. Acessado em 28 de abril de 2020, às 21h58min.

\_\_\_\_\_. Cascading Style Sheets, level 2 Revision 2. Disponível em <<https://www.w3.org/TR/2016/WD-CSS22-20160412/>>. Acessado em 28 de abril de 2020, às 22h17min.

\_\_\_\_\_. Cascading Style Sheets, level 2. Disponível em <<https://www.w3.org/TR/2008/REC-CSS2-20080411/>>. Acessado em 28 de abril de 2020, às 22h03min.

\_\_\_\_\_. Cascading Style Sheets, level 3. Disponível em <<https://www.w3.org/TR/css-syntax-3/>>. Acessado em 28 de abril de 2020, às 22h18min.

\_\_\_\_\_. HTML 3.2 Reference Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html32-20180315/>>. Acessado em 28 de abril de 2020, às 19h37min.

\_\_\_\_\_. HTML 4.0 Specification. Disponível em <<https://www.w3.org/TR/1998/REC-html40-19980424/>>. Acessado em 28 de abril de 2020, às 19h53min.

\_\_\_\_\_. HTML 4.01 Specification. Disponível em <<https://www.w3.org/TR/2018/SPSD-html401-20180327/>>. Acessado em 28 de abril de 2020, às 20h04min.

\_\_\_\_\_. Cascading Style Sheets, level 2 Revision 1. Disponível em <<https://www.w3.org/TR/CSS2/>>. Acessado em 28 de abril de 2020, às 22h13min.

WIKIPEDIA. JavaScript. Disponível em <<https://pt.wikipedia.org/wiki/JavaScript>>. Acessado em 29 de abril de 2020, às 10h.

- Dúvidas?
- Críticas?
  - Sugestões?
  - Ameaças?