Aula02 - OiDevs

Presenças

Aumentar tela

Continuando aula passada

```
npx create-react-app my-app-js
cd my-app-js
npm start
```

```
Need to install the following packages:
  create-react-app@5.0.1
```

```
Installing react, react-dom, and react-scripts with cra-template...
```

```
npx create-react-app my-app-ts --template typescript
cd my-app-ts
npm start
```

var, let e const

Declarações com `var` tem escopo global ou escopo de função/local.

```
var teste = "boa hi";
```

```
  function newFunction() {
    var hello = "hello";
    console.log("🚀 ~ file: App.tsx:11 ~ newFunction ~ hell
o:", hello);
  }
  // console.log("🚀 ~ file: App.tsx:11 ~ newFunction ~ hell
o", hello);
  var greeter = "hey hi";
  var greeter = "say Hello instead";
  var greeter = "hey hi";
  greeter = "say Hello instead";
  newFunction();
  var greeter = "hey hi";
  var times = 4;
  if (times > 3) {
    var greeter = "say Hello instead2";
  }
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <span>{tester}</span>
        <span>{greeter}</span>
      </header>
    </div>
  );
```

let tem escopo de bloco

Um bloco é uma porção de código cercado por {}. Um bloco vive dentro dessas chaves. Tudo o que estiver cercado por chaves é um bloco.
Assim, uma variável declarada com `let` em um bloco estará disponível apenas dentro daquele bloco.

```
  let greeting = "say Hi";
  let times = 4;
  if (times > 3) {
    let hello = "say Hello instead";
    console.log("🚀 ~ file: App.tsx:11 ~ App ~ hello", hello);
    console.log(hello); // dirá "say Hello instead"
  }
  // console.log("🚀 ~ file: App.tsx:11 ~ App ~ hello", hell
o);
  // let greeting2 = "say Hi 2";
  greeting = "say Hello instead";
  // greeting2 = "say Hello instead";
  // let greeting2 = "say Hi 3";
  let greetingSameBlock = "say Hi";
  if (true) {
    let greeting = "say Hello instead";
    console.log("🚀 ~ file: App.tsx:26 ~ App ~ greeting", gree
ting);
  }
  console.log(
    "🚀 ~ file: App.tsx:24 ~ App ~ greetingSameBlock",
    greetingSameBlock
  );
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <span>{greeting}</span>
        {/* <span>{greeting2}</span> */}
      </header>
    </div>
  );
```

Const

Variáveis declaradas com `const` mantêm valores constantes. Declarações com `const` compartilham algumas semelhanças com as declarações com `let`.

```
  const greeting = "say Hi";
  greeting = "say Hello instead";
  const greeting = "say Hi";
  const greeting = "say Hello instead";
  const greeting = {
    message: "say Hi",
    times: 4,
  };
  greeting = {
    words: "Hello",
    number: "five",
  }; // erro:  atribuição a uma variável constante.
  greeting.message = "say Hello instead";
```

Arrow function

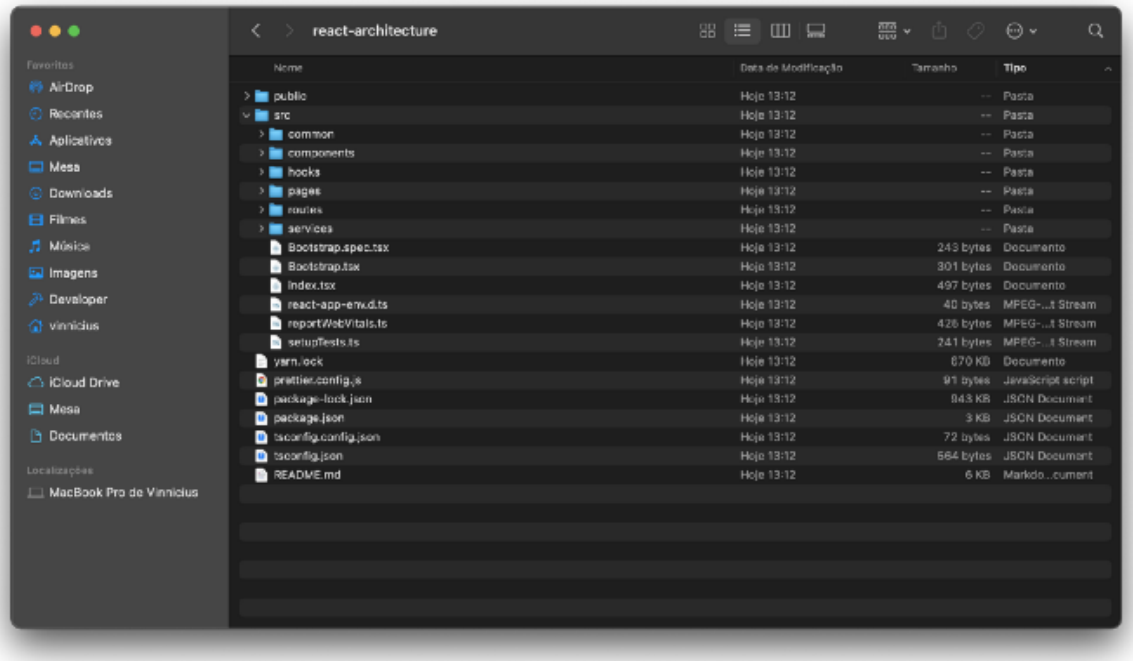
```
import React from 'react';
import logo from './logo.svg';
import './App.css';
const App = () => {
  // (parameter1, parameter2, ..., parameterN) => expression;
  // ES5: Without arrow function
  // const getResult = function (username, points) {
  const getResult = function (username: string, points: numbe
r) {
    return username + " scored " + points + " points!";
  };
  // ES6: With arrow function
```

```
  const getResult2 = (username: string, points: number): string => {
    return `${username} scored ${points} points!`;
  };
  console.log(
    "🚀 ~ file: App.tsx:41 ~ //getResult ~ getResult",
    getResult("raniel", 20)
  );
  console.log(
    "🚀 ~ file: App.tsx:41 ~ //getResult ~ getResult2",
    getResult2("mendonca", 10)
  );
  const sum = (a: number, b: number): number => {
    return a + b;
  };
  console.log(sum(20, 30));
  const print = () => console.log("Hello JavaTpoint!");
  print();
  let sumClean = (a: number, b: number) => a + b;
  console.log("SUM: " + sumClean(5, 15));
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.tsx</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
```

```
43          >
44            Learn React
45          </a>
46        </header>
47      </div>
48    );
49  };
50  export default App;
51
```

```
const Greeting = () => <h1>Hello World today!</h1>;
```

Arquitetura de componentes em pastas



O que é cada pasta:

- ***Common***

    - Aqui estão todos os asstes do projeto que serão usados pelo aplicativo, como estilos globais, imagens, fontes, mocks, stories, funções reutilizáveis como máscaras, entre outros.
- ***Components***
    - Aqui vai ficar todos os componentes que são utilizados de uma forma global pela aplicação, componentes utilizados somente por uma página em específica vai ficar em outro lugar.
- ***Configs***
    - Aqui vai ficar todos os arquivos de configuração, que são utilizados de uma forma global pela aplicação.
- ***Containers***
    - Aqui vai ficar todos os nossos containers responsáveis por desacoplar a nossa aplicação de alguma biblioteca, possibilitando alterar as libs sem precisar mexer em varios lugares do código.
- ***Hooks***
    - Aqui eu gosto de deixar todos os hooks customizáveis da aplicação, por exemplo, um hook que cuida da sessão do usuário.
- ***Pages***
    - Como o próprio nome já diz, aqui vai ficar todas as páginas da nossa aplicação, ah e lembra que falei que os componentes utilizados somente por uma página ficariam em outro lugar?! Então, aqui é o lugar também, dentro de cada pasta de página vamos ter uma pasta components que contém todos os componentes exclusivos da página.
- ***Routes***
    - Aqui vai ficar todos os nosso arquivos que gerenciam as rotas da nossa aplicação.
- ***Services***
    - Nessa pasta fica todos os arquivos responsáveis por consumir serviços externos, como por exemplo o arquivo de configuração do axios para consumir APIs RestFul.

```
components
|-- YourComponent
   |-- YourComponent.js
   |-- YourComponent.stories.js
   |-- YourComponent.spec.js
   |-- styles.js
   |-- index.js
```

```
...
```

- *Um arquivo responsável pelo componente:* `YourComponent.tsx`
- *Um arquivo para os stories do StoryBook:* `YourComponent.stories.tsx`
- *Um arquivo para testes:* `YourComponent.spec.tsx`
- *Um arquivo para as interfaces (Quando uso TypeScript):* `interfaces.ts`
- *Um arquivo para o styled-components:* `styles.ts`
- *E um arquivo responsável por exportar o componente:* `index.ts`

E por último, como fica a pasta de uma página:

```
pages
|-- YourPage
   |-- components/
   |-- YourPage.tsx
   |-- YourPage.spec.tsx
   |-- interfaces.ts
   |-- styles.ts
   |-- index.ts
...
```

- *Uma pasta responsável pelos componentes que **só são utilizados** naquela página, ah e essa pasta segue a mesma estrutura da pasta de componentes que mostrei acima* 👆
- *Um arquivo responsável pela página:* `YourPage.tsx`
- *Um arquivo para testes:* `YourPage.spec.tsx`
- *Um arquivo para as interfaces (Quando uso TypeScript):* `interfaces.ts`
- *Um arquivo para o styled-components:* `styles.ts`
- *E um arquivo responsável por exportar a página:* `index.ts`

O que são componentes?

Os componentes são os blocos de construção de qualquer aplicativo React e um aplicativo React típico terá muitos deles. Simplificando, um componente é uma

classe ou função JavaScript que opcionalmente aceita entradas, ou seja, propriedades (props) e retorna um elemento React que descreve como uma seção da interface do usuário (interface do usuário) deve aparecer.

Componentes são blocos de código (JS, HTML = JSX e CSS) que podem ser reutilizados em vários locais da aplicação.

```
const Greeting = () => {
  return (
        <h1>Hello World today!</h1>
    );
};
export default Greeting;

```

```
import Greeting from "./Greeting";
export { Greeting };
export default Greeting;

```

```
import Greeting from "./components";
```

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import Greeting from "./components";

const App = () => {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.tsx</code> and save to reload.
```

```
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
        <Greeting />
      </header>
    </div>
  );
};

export default App;
```