

Luz, Câmera, React!

Módulo 3 - Aula 4 - HARD



Photo by Mason Kimbarovsky on Unsplash



Todos os direitos reservados
©2022 Resilia Educação

RESILIA |  **Senac**

Introdução ao React

JUST START.



Todos os direitos reservados
©2022 Resilia Educação

React

- ⇒ Uma das bibliotecas mais populares e usadas no mercado de trabalho;
- ⇒ Biblioteca JavaScript para front-end;
- ⇒ Projeto open-source criado pela Meta (ex-Facebook)
- ⇒ Em um modelo MVC, React é a camada da View

React - Configuração e instalação

⇒ **Para usufruirmos dessa biblioteca devemos ter instalados:**

- NPM (gerenciador de pacotes do Node.js)
- Webpack
- Babel

⇒ **Opcionais:**

- React Dev Tools (extensão do navegador)
- Prettier, ES7 React/Redux/GraphQL (extensão do VS code)

React - Configuração e instalação

- ⇒ Para instalação do React em nosso projeto podemos:
- Instalar a partir de um arquivo **HTML estático**
 - Utilizar o template do **Create React App (CRA)**
 - Utilizar o build tool **Vite (fortemente recomendado)**

Create React App

Cria um ambiente de desenvolvimento já configurado e otimizado para implementação de aplicações React.



Babel

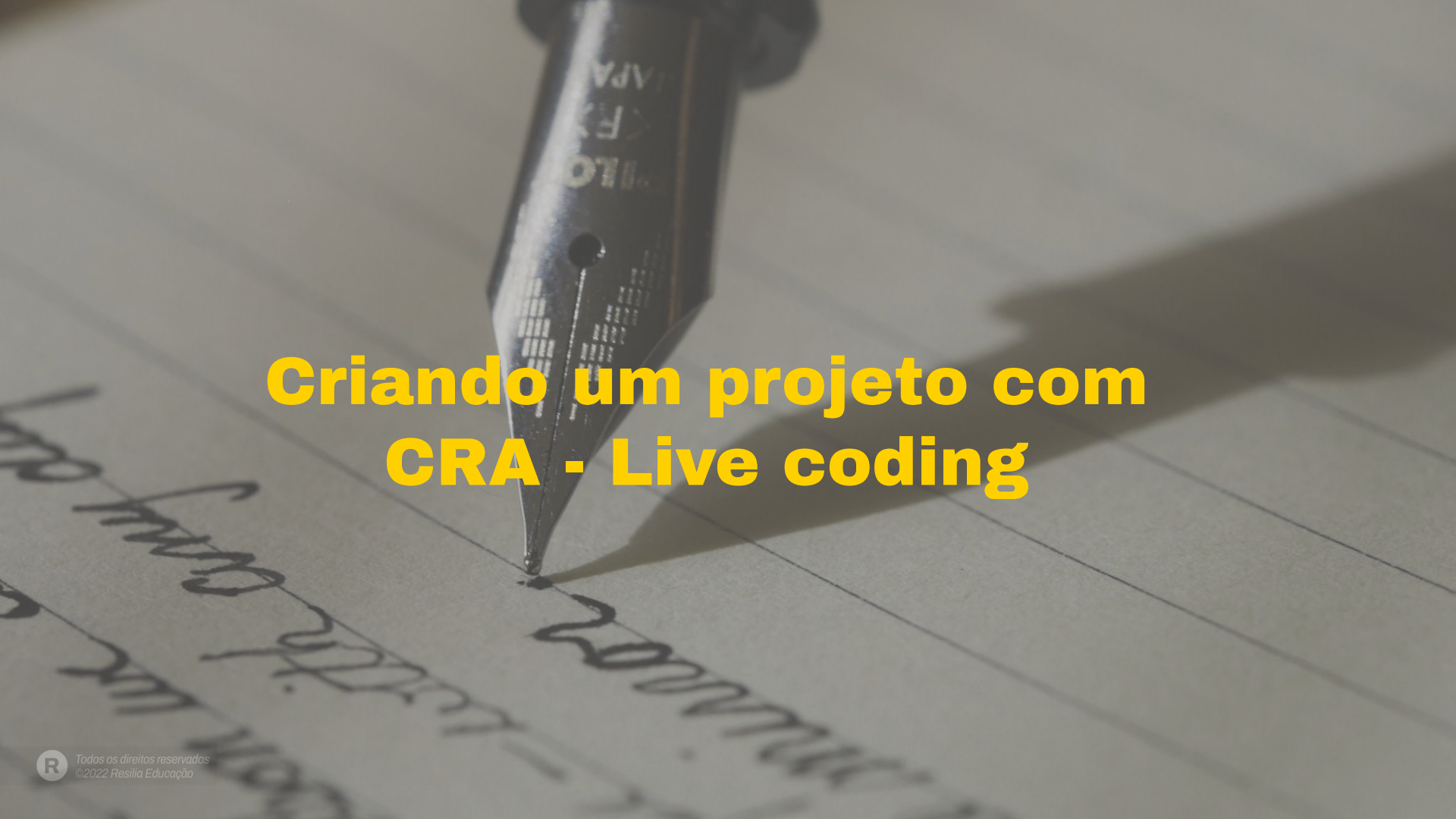
É um **transpilador** JavaScript que permite a escrita de código JavaScript com sintaxes e recursos mais atuais, como do ES6. Sendo assim, é responsável por **transformar esse código em um equivalente**, mas com uma sintaxe mais antiga, como a do ES3.

Isso nos traz vantagens de **suporte nos navegadores**.

Webpack

Webpack é um **bundler**, responsável por agrupar todo o código e separá-lo em módulos. Pode ser, portanto, entendido como um **empacotador de módulos**.

Ele garante, por exemplo, que não tenhamos conflitos de código no nosso projeto, desde os mais simples, como nome de variáveis, até outros mais complexos.

A close-up photograph of a fountain pen's nib writing on a piece of paper. The paper has faint, cursive handwriting that appears to be a list or notes. The pen is dark and has some text on its barrel, including 'PILLO' and 'F3'. The lighting is soft, creating a slight shadow of the pen on the paper.

Criando um projeto com CRA - Live coding





Vite é um **build tool**, serve para facilitar a configuração e instalação de vários projetos, com diversas tecnologias, como React e Vue.

Tecnologia que tem entrado em ascensão, por ser uma ferramenta de build (build tool) para diversas tecnologias e oferecer configurações iniciais melhores que outras do mercado.



Vite vs CRA

- ⇒ Servidor **mais rápido** que do CRA
- ⇒ Processo de **build otimizado**
- ⇒ Oferece uma API de **Hot Module Replacement**, que oferece atualizações instantâneas e precisas sem recarregar a página ou destruir o estado da aplicação, integrado com o Fast-Refresh do React



Criando um projeto com Vite - Live coding



JSX (1)

JavaScript XML / Extension

É a **extensão de sintaxe do JavaScript**, introduz elementos de XML e HTML, que são convertidos em funções React, facilitando a sintaxe e a legibilidade do código.

JSX:

```
function App() {  
    return <button>Comprar</button>  
}
```

É transformado para:

```
function App() {  
    return React.createElement('button',  
    null, 'Comprar')  
}
```

JSX (2)

- ⇒ Não é obrigatório sua sintaxe no React
- ⇒ É mais próximo de JavaScript do que HTML
- ⇒ Expressões JS podem ser colocadas dentro do JSX usando chaves { }
- ⇒ É mais fácil de escrever e entender do que criar e adicionar vários elementos com JavaScript puro

JSX (3)

- ⇒ Algumas diferenças importantes:
- ⇒ Atributos podem ser passados como no HTML, mas existem casos especiais
 - ex.: class para a ser **className**, for para a ser **htmlFor**
- ⇒ Atributos com nome composto devem seguir o padrão camelCase
- ⇒ Podemos passar expressões (funções, variáveis) dentro de chaves { }

```
function App() {  
  const nome = 'Marcelo'  
  return <p>{nome}</p>  
}
```



Componentes



Componentes

Podemos pensar em “funções” das quais podemos chamar e reutilizar diversas vezes desde que sejam passados os argumentos necessários. Isso torna o React poderoso e otimizado.

O ideal é dividir sua aplicação React em **diversos componentes**.

Componentes - Class

Antes da versão 16.8 do React ser lançada todos os componentes eram baseados em classes.

```
class Button extends React.component {  
  render() {  
    return <button>Comprar</button>  
  }  
}
```

Componentes - Class

Esses componentes tinham vários métodos herdados de `React.component`, mas o **único** obrigatório é o `render()`, já que é o **responsável pela renderização dos elementos**.

Dica: salvar o arquivo do componente com a extensão `.jsx` trará vantagens como snippets (sugestões de código ao você digitar)

Componentes - Functions

Desde o lançamento da versão 16.8 do React, os componentes funcionais foram adotados e são fortemente sugeridos, eles trazem diversas vantagens.

```
function Button() {  
  return <button>Comprar</button>  
}
```

Componentes - Functions

A sintaxe é bem mais curta e não precisamos estender ao `React.component`, o método `render()` já não é mais necessário, já que faz sentido que todo componente retorne elementos HTML.

```
function Button() {  
  return <button>Comprar</button>  
}
```

Componentes - Functions

Também podemos criar componentes funcionais com arrow functions.

```
function Button() {  
  return <button>Comprar</button>  
}
```



```
const Button = () => {  
  return <button>Comprar</button>  
}
```

Componentes - Functions

É válido ressaltar que **todo nome de componente inicia com a letra maiúscula**, essa foi uma convenção que permaneceu ainda dos tempos do React com abordagens de componentes de classe.

Mão na Massa: Criar componente de formulário

- ⇒ Criar uma pasta components na pasta src
- ⇒ Criar um arquivo Form.jsx
- ⇒ Esse arquivo terá:
 - 2 inputs (nome e e-mail)
 - Uma área de texto
 - Um botão para envio do formulário
- ⇒ O componente deve ser chamado no componente App.js

Props

As props são propriedades que podemos passar para componentes React, elas são somente de leitura e passamos elas via atributo.

```
function Button(props) {  
  const { text } = props  
  return (  
    <button>  
      {text}  
    </button>  
  )  
}
```

Props (2)

Outra forma de desestruturar as props é diretamente nos parênteses dos parâmetros.

```
function Button({ text }) {  
  return (  
    <button>  
      {text}  
    </button>  
  )  
}
```

Mão na Massa: Criar componente de botão

- ⇒ Dentro da pasta components criar um arquivo Button.jsx
- ⇒ Criar um objeto com 10 cores escolhidas pelo grupo
- ⇒ O componente Button terá uma prop chamada color
- ⇒ A cor de fundo deve ser alterada de acordo com a propriedade escrita em color
- ⇒ Substituir o botão do componente Form pelo novo componente Button



Momento Aprendizagem em Pares

