Introdução ao React

Labenu_

O que vamos ver hoje?

- Evolução da web até hoje
- Por que React?
- Criando um projeto React
- JSX: HTML + JS no React

Mas antes disso...

DOM

Labenu_

DOM 📄

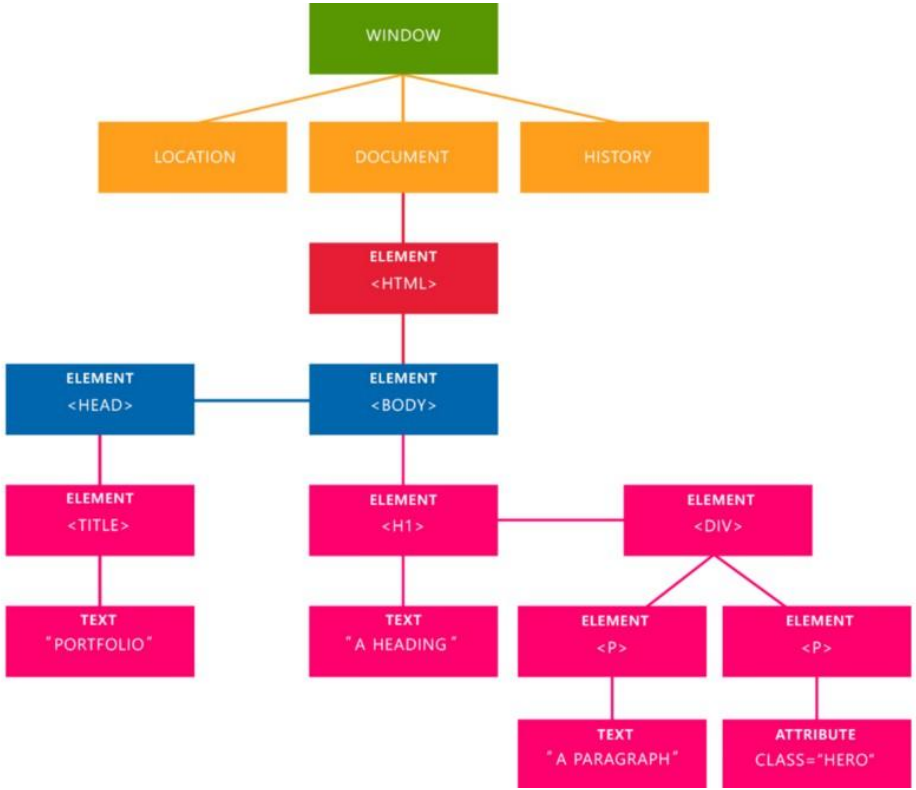
- DOM significa **Document Object Model**, ou seja, "**modelo de objeto do documento**"

- É um objeto gigante dentro do navegador com todos recursos para criar uma aplicação (seriam análogas a ferramentas)

- **Árvore DOM:** Podemos dizer que o DOM tem formato de “árvore”, e para criar nossas aplicações, precisamos percorrer ela subindo e descendo, utilizando suas ramificações

Vamos ver no console do navegador!

DOM


WINDOW
LOCATION
DOCUMENT
HISTORY
ELEMENT
<HTML>
ELEMENT
<HEAD>
ELEMENT
<BODY>
ELEMENT
<TITLE>
ELEMENT
<H1>
ELEMENT
<DIV>
TEXT
"PORTFOLIO"
TEXT
"A HEADING"
ELEMENT
<P>
ELEMENT
<P>
TEXT
"A PARAGRAPH"
ATTRIBUTE
CLASS="HERO"

Eventos

Labenu_

Eventos 

- De uma maneira geral, chamamos de **evento** um processo em que o **status** ou o **valor** de um elemento HTML é **alterado**

- Exemplos de evento:
  - Usuário clica em algum elemento
  - Usuário insere um valor em uma tag input
  - Usuário clica na tecla "enter" do seu computador

Eventos 🎟️

- Para pegar um evento de um elemento HTML, usamos esta sintaxe:

```
<elemento evento="nomeDaFuncao()">
```

Eventos 🎟️

- **Eventos mais comuns**
  - **onchange:** um elemento ou seu valor foram alterados
  - **onclick:** usuário clicou no elemento
  - **onmouseover:** usuário posicionou mouse em cima do elemento
  - **onmouseout:** usuário tirou o mouse do elemento

Exemplos

Um pouco de história

Labenu_

Antigamente... 👨‍🦳

- A programação era feita **manipulando o DOM diretamente**

- **Problemas** dessa abordagem:
  - Funções disponíveis difíceis de manipular, além de ser difícil manter um padrão no código
  - Dev precisa de preocupar com muitos detalhes, sendo improdutivo
  - Toda vez que um dado é alterado, o DOM inteiro precisa ser atualizado

Bibliotecas 

- Surgiram diversas **bibliotecas** para facilitar esse processo de desenvolvimento Web

- Uma biblioteca é um **conjunto de funções** com um determinado **propósito**

- Escondem código **complexo** atrás de código **amigável**

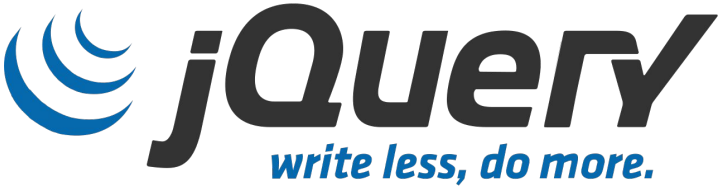
Evolução 

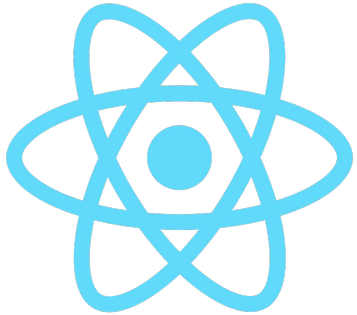
- Com a **evolução dos computadores**, as possibilidades de criação de sites aumentaram

- **Navegadores** aguentam códigos mais pesados e aplicações mais robustas

- **Bibliotecas** com mais funcionalidades surgem, facilitando ainda mais o desenvolvimento

Bibliotecas de Desenvolvimento Web





React 💙

- Uma dessas bibliotecas é o **React**!

- Desenvolvida pelo **Facebook**, é uma das bibliotecas mais usadas para desenvolvimento web no mundo

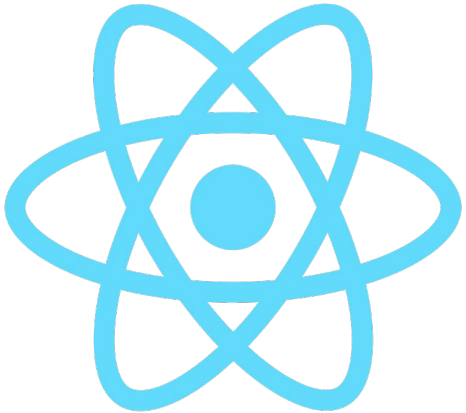
- Está em constante evolução: possui atualizações e otimizações frequentes

Por que React?

Labenu_

Por que React? 🤔

1) Muito utilizada por vários devs
2) Só é preciso utilizar Javascript
3) Criada e mantida pelo Facebook
4) Reatividade
5) Componentização



1) Muito utilizado 👬

- Já vimos que React é uma das bibliotecas mais **populares** no mundo para desenvolvimento web

- Mais gente usando gera uma **comunidade maior**

- Mais recursos, bibliotecas, tutoriais, vídeos no YouTube, perguntas respondidas no StackOverflow...

2) Só Javascript 😀

- Todo código que vamos escrever é **Javascript**
- Centraliza os 3 pilares da Web (HTML, CSS e JS)
- Diferentemente de outras bibliotecas, não requer o aprendizado de outras linguagens de programação ou particularidades que **fogem dos padrões do JS**

3) Criado pelo Facebook 👩‍💻

- Grande empresa no comando nos dá segurança
- Garantia de qualidade de código
- Garantia de manutenção frequente
- Tecnologia e engenheiros(as) de ponta

4) Reatividade 👊

- Pelo nome, é possível inferir que a **reatividade** é um grande **pilar do React**

- Ela representa o fato de que quando os **dados mudam**, a interface (DOM) reflete a mudança imediatamente e automaticamente

- **Facilita** muito a vida da equipe de desenvolvimento e previne bugs

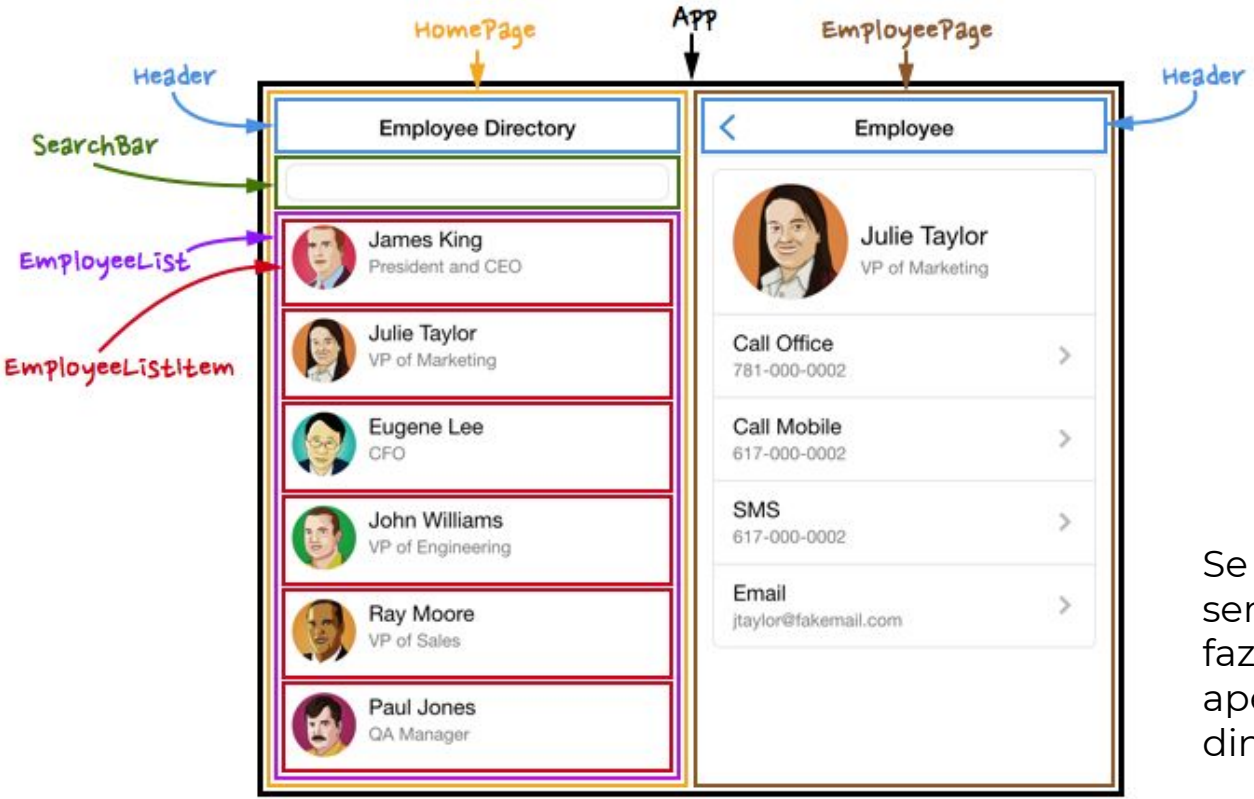
5) Componentização 🧩

- O React permite de forma muito simples quebrar o código em **componentes**

- Componentes são **blocos de código** que podem ser nomeados e reutilizados (como funções)

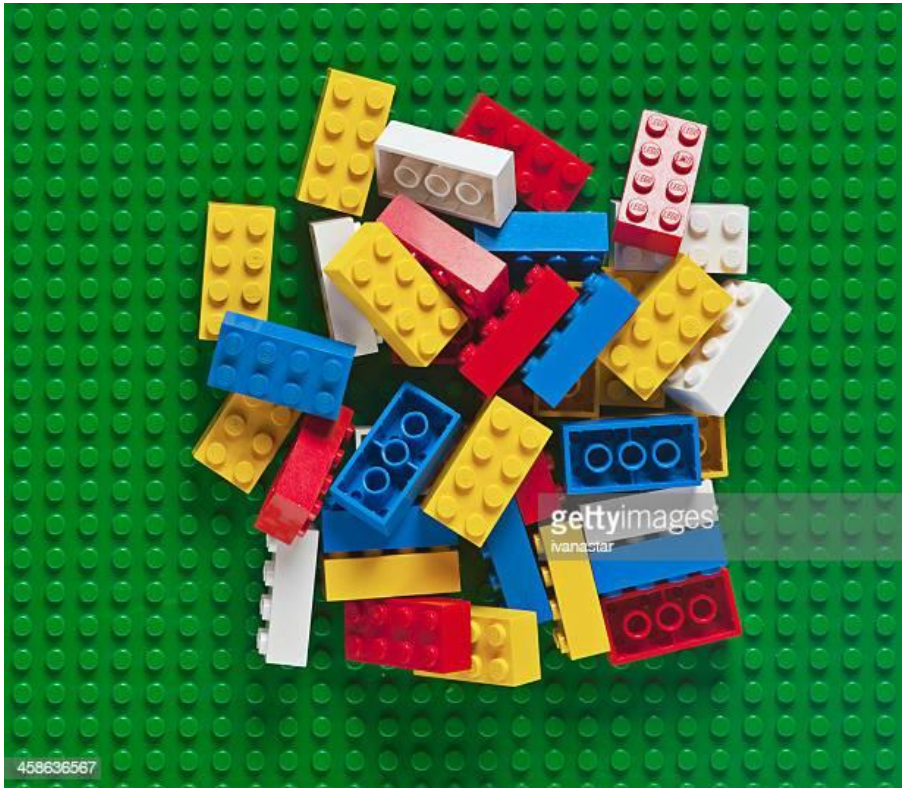
- Podem receber **propriedades** de outros componentes

5) Componentização



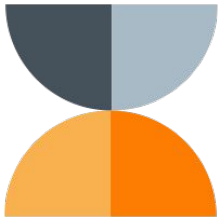
Se o layout é sempre igual, faz sentido tornar apenas as infos dinâmicas

5) Componentização



Analogia com Lego

Os blocos são componentes que se repetem. São blocos prontos de código que pode repetir na aplicação.
Ex: caixa de comentário, botões...



Pausa para relaxar 😴

10 min

- Para o Javascript poder interagir com o HTML, precisamos do **DOM (Document Object Model)**
- **Evento** ocorre quando o **status** ou o **valor** de um elemento HTML é **alterado**

Criando um Projeto React

Labenu_

Node 📦

- Para desenvolver com React, precisamos usar um programa chamado **Node**

- Ele nos dá **ferramentas** para instalar, rodar, gerar builds e deployar nossas aplicações

- Mais para frente, vamos entender melhor o que é o Node e como usá-lo. Por hora, **foco no React**

para checar se você tem o node instalado, basta rodar node -v no terminal

Criando um Projeto React ✨

- Para criar um app React, basta:
  1. Navegar até a **pasta** desejada no terminal
  2. Rodar o seguinte comando:

```
npx create-react-app nome-do-app
```

Indica o nome do app a ser criado

Vamos ver na prática! 

Criando um Projeto React ✨

- O comando **cria uma pasta** com o nome escolhido
- Dentro dela, serão criados vários arquivos e algumas pastas
- Eles são necessários para o React funcionar
- Por enquanto, vamos olhar só para os que interessam

Criando um Projeto React ✨

- Das arquivos criados, vamos mexer somente dentro da pasta **src**

- Os arquivos que serão editados (por enquanto) são somente o **App.js** e o **App.css**

- Todos os outros arquivos e pastas devem ser **ignorados** por enquanto

App.js ✨

- Arquivo que contém o **componente principal** da aplicação
- Essencialmente, é o “pai” de todo o site
- Por um tempo vamos trabalhar com um único componente e, portanto, **somente nesse arquivo**

App.css ✨

- Arquivo que contém o código CSS para a aplicação
- Por enquanto, vamos manter todo o CSS nele

Rodando o Projeto ✨

- Com o projeto criado, podemos rodá-lo para ver se tudo foi criado corretamente

- Rode os dois comandos abaixo, na mesma pasta:

Entra na pasta criada anteriormente

```
cd nome-do-app
npm start
```

Roda o projeto

JSX

Labenu_

JSX 

- JSX é uma sintaxe que permite gerar **código HTML** a partir de **código Javascript**

- Ela é **muito parecida** com HTML, mas possui algumas particularidades

- No momento de executar, ela é transformada em Javascript puro

Componentes 🧩

- Um componente em React é representado por uma **função que retorna um JSX**

- Durante a semana, vamos nos aprofundar mais sobre componentes

- Por enquanto, vamos observar somente o componente **App**, no arquivo **App.js**

```jsx
import React from 'react';
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;

```

App.js

```
1  import React from 'react';
2  import logo from './logo.svg';
3  import './App.css';
4
5  function App() {
6    return (
7      <div className="App">
8        <header className="App-header">
9          <img src={logo} className="App-logo" alt="logo" />
10         <p>
11           Edit <code>src/App.js</code> and save to reload.
12         </p>
13         <a
14           className="App-link"
15           href="https://reactjs.org"
16           target="_blank"
17           rel="noopener noreferrer"
18         >
19           Learn React
20         </a>
21       </header>
22     </div>
23   );
24 }
25
26 export default App;
27
```

App.js

- Componente App

```
1  import React from 'react';
2  import logo from './logo.svg';
3  import './App.css';
4
5  function App() {
6    return (
7      <div className="App">
8        <header className="App-header">
9          <img src={logo} className="App-logo" alt="logo" />
10         <p>
11           Edit <code>src/App.js</code> and save to reload.
12         </p>
13         <a
14           className="App-link"
15           href="https://reactjs.org"
16           target="_blank"
17           rel="noopener noreferrer"
18         >
19           Learn React
20         </a>
21       </header>
22     </div>
23   );
24 }
25
26 export default App;
27
```

App.js

- Componente App
- JSX Retornado

Diferenças JSX e HTML

Labenu_

Diferenças entre JSX e HTML ⚠️

- O JSX possui algumas diferenças para o HTML

- Vamos ver as principais:
  - Atributos - Nomes e Valores
  - Funções de Eventos
  - Regra do pai único
  - Usando expressões Javascript
  - Self-closing tags

Atributos - Nomes ℹ️

- É possível utilizar os mesmos atributos presentes em elementos HTML

- Porém, alguns nomes são levemente diferentes:
  - **class ⇒ className**
  - Atributos com múltiplas palavra viram **camelCase**
    - onclick ⇒ onClick
    - onchange ⇒ onChange

Atributos - Nomes ℹ️

```jsx
import React from 'react';
import logo from './img/logo.png';
import './App.css';

function App() {

  const onClickBotao = () => {
    alert('Botão foi clicado!')
  }

  return (
    <div className="App">
      <h1>Aprenda React com a Labenu!</h1>
      <div>
        <div>
          <img src={logo} alt="logo"/>
        </div>
        <div>
          <a href="https://labenu.com.br">Site da Labenu!</a>
        </div>
        <div>
          <button onClick={onClickBotao}>Clique aqui!</button>
        </div>
      </div>
    </div>
  );
}

export default App;
```

Na dúvida, **pesquise**!

Ex: [Tal Atributo] em JSX

Atributos - Valores ℹ️

- Em geral, os valores dos atributos são passados da mesma forma que antes

- Existem duas diferenças principais:
  - Imagens locais
  - Funções de eventos (onClick, onChange, etc...)

Atributos - Imagens Locais ℹ️

- Antes, podíamos passar o **path** (caminho) da imagem, relativo ao arquivo atual, para o atributo **src**

- Agora, para usar uma imagem local, é preciso **importá-la no topo do arquivo**, e só então passar ela para o src entre {}

- Sintaxe: `import nomeImagem from '../path/da/imagem'`

Atributos - Imagens Locais ℹ️

```
1  import React from 'react';
2  import logo from './img/logo.png';
3  import './App.css';
4
5  function App() {
6
7    const onClickBotao = () => {
8      alert('Botão foi clicado!')
9    }
10
11   return (
12     <div className="App">
13       <h1>Aprenda React com a Labenu!</h1>
14       <div>
15         <div>
16           <img src={logo} alt="logo"/>
17         </div>
18         <div>
19           <a href="https://labenu.com.br">Site da Labenu!</a>
20         </div>
21         <div>
22           <button onClick={onClickBotao}>Clique aqui!</button>
23         </div>
24       </div>
25     </div>
26   );
27 }
28
29 export default App;
```

Funções de Eventos - ANTES ▶️

- Antes, era possível declarar funções no arquivo JS e referenciá-las no HTML, nos atributos de evento

- As funções eram chamadas entre aspas e com os parênteses, por exemplo:

```
<button onclick="onClickBotao()">Clique aqui</button>
```

Funções de Eventos 🥳

- Agora, podemos declarar a função **no próprio componente**, antes do return

- Passamos a função **entre chaves** e **sem parênteses**:

```
<button onClick={onClickBotao}>Clique aqui</button>
```

Funções de Eventos ▶️

```
1  import React from 'react';
2  import logo from './img/logo.png';
3  import './App.css';
4
5  function App() {
6
7    const onClickBotao = () => {
8      alert('Botão foi clicado!')
9    }
10
11   return (
12     <div className="App">
13       <h1>Aprenda React com a Labenu!</h1>
14       <div>
15         <div>
16           <img src={logo} alt="logo"/>
17         </div>
18         <div>
19           <a href="https://labenu.com.br">Site da Labenu!</a>
20         </div>
21         <div>
22           <button onClick={onClickBotao}>Clique aqui!</button>
23         </div>
24       </div>
25     </div>
26   );
27 }
28
29 export default App;
```

- Declaração da função
- Atrelando ao clique do botão

Regra do Pai Único 1️⃣

- Todas as expressões JSX devem ter **um único pai**

```jsx
import React from 'react';
import logo from './img/logo.png';
import './App.css';

function App() {

  const onClickBotao = () => {
    alert('Botão foi clicado!')
  }

  return (
    <div className="App">
      <h1>Aprenda React com a Labenu!</h1>
      <div>
        <div>
          <img src={logo} alt="logo"/>
        </div>
        <div>
          <a href="https://labenu.com.br">Site da Labenu!</a>
        </div>
        <div>
          <button onClick={onClickBotao}>Clique aqui!</button>
        </div>
      </div>
    </div>
  );
}

export default App;
```

```jsx
import React from 'react';
import logo from './img/logo.png';
import './App.css';

function App() {

  const onClickBotao = () => {
    alert('Botão foi clicado!')
  }

  return (
    <h1>Aprenda React com a Labenu!</h1>
    <div>
      <div>
        <img src={logo} alt="logo"/>
      </div>
      <div>
        <a href="https://labenu.com.br">Site da Labenu!</a>
      </div>
      <div>
        <button onClick={onClickBotao}>Clique aqui!</button>
      </div>
    </div>
  );
}

export default App;
```

Expressões Javascript 🔣

- Podemos usar expressões Javascript no meio do código JSX

- Uma expressão Javascript é tudo aquilo que representa **um único valor**

- Para usar o valor da expressão, basta colocá-la entre **{chaves}**

Expressões Javascript

```
1  import React from 'react';
2  import logo from './img/logo.png';
3  import './App.css';
4
5  function App() {
6    const tituloPagina = 'Aprenda React com a Labenu!'
7
8    const onClickBotao = () => {
9      alert('Botão foi clicado!')
10   }
11
12   return (
13     <div className="App">
14       <h1>{tituloPagina}</h1>
15       <div>
16         <div>
17           <img src={logo} alt="logo"/>
18         </div>
19         <div>
20           <a href="https://labenu.com.br">Site da Labenu!</a>
21         </div>
22         <div>
23           <button onClick={onClickBotao}>Clique aqui!</button>
24         </div>
25       </div>
26     </div>
27   );
28 }
29
30 export default App;
```

- Declaração da variável
- Usando a variável no meio do JSX - ela é uma expressão pois representa um único valor

Self-closing Tags 🔄

- Elementos que não possuem filhos (img, input, br, hr) possuem uma sintaxe levemente diferente

- Devem seguir sintaxe *self-closing*

- Qualquer outro elemento que não tenha filhos pode seguir essa sintaxe

```
<elemento />
```

Self-closing Tags 🔄

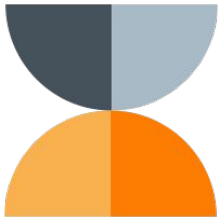
```
import React from 'react';
import logo from './img/logo.png';
import './App.css';

function App() {
  const tituloPagina = 'Aprenda React com a Labenu!'

  const onClickBotao = () => {
    alert('Botão foi clicado!')
  }

  return (
    <div className="App">
      <h1>{tituloPagina}</h1>
      <div>
        <div>
          <img src={logo} alt="logo"/>
        </div>
        <div>
          <a href="https://labenu.com.br">Site da Labenu!</a>
        </div>
        <div>
          <button onClick={onClickBotao}>Clique aqui!</button>
        </div>
      </div>
    </div>
  );
}
```

Exercício 1

- Edite o arquivo App.js para renderizar a página ao lado

```jsx
import React from 'react';
import logo from './img/logo.png';
import './App.css';

function App() {

  const onClickBotao = () => {
    alert('Botão foi clicado!')
  }

  return (
    <div className="App">
      <h1>Aprenda React com a Labenu!</h1>
      <div>
        <div>
          <img src={logo} alt="logo"/>
        </div>
        <div>
          <a href="https://labenu.com.br">Site da Labenu!</a>
        </div>
        <div>
          <button onClick={onClickBotao}>Clique aqui!</button>
        </div>
      </div>
    </div>
  );
}

export default App;
```

[Ver no Code Sandbox]



Pausa para relaxar 😴

5 min

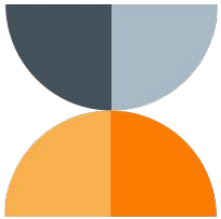
- Criar um app React:
  `npx create-react-app nome-do-projeto`

- Rodar um app React:
  `npm run start`

- Arquivos relevantes: **App.js** e **App.css**

Pausa para relaxar 😴

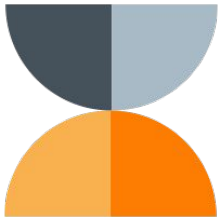
5 min



- **Componente:** função que retorna um JSX

- **JSX:** linguagem parecida com HTML misturada com Javascript

- Muitas coisas são comuns entre HTML e JSX mas ressaltamos **algumas diferenças**



Pausa para relaxar 😴

5 min

- `class` ⇒ `className`
- `onclick` ⇒ `onClick` (eventos em camelCase)
- Imagens locais precisam ser **importadas**
- Declaramos função **no próprio componente**
- `onclick="apertou()"` ⇒ `onClick={apertou}`
- Regra do **pai único**
- Podemos referenciar expressões no JSX
- Self-closing tags: `<evento />`

Deploy com Surge

Labenu_

Deploy ⬆️

- Quando fazemos um site (ou qualquer outro programa), queremos que o mundo possa acessá-lo pela internet

- Subir um programa para a internet é o que chamamos de **deploy**

- Vocês já conhecem uma ferramenta que faz isso, o surge

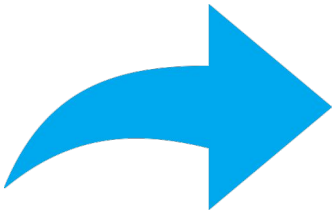
Build 📦

- Para fazer o deploy de um site React, é necessário gerar um **build** antes
- Um build é uma versão **estática** da aplicação, que não precisa do comando *npm run start* para rodar
- O código gerado é ilegível, e serve apenas pro deploy
- Para gerar o build, usamos: `npm run build`

Build 📦

- O build pega o código “amigável” do React e transforma em Javascript puro para o navegador entender

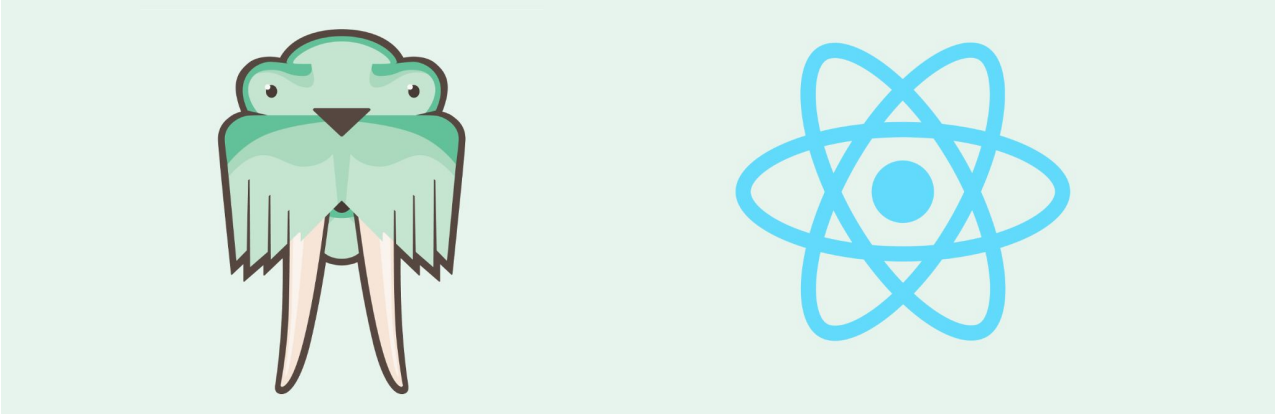
Código que você criou em React



Código ilegível, sem indentação, em javascript puro, uma bagunça só

Surge 🔧

- Surge é a aplicação que possibilita “subir” o código pra algum lugar pra podermos compartilhar com o mundo



Surge 🔧

- Para fazer o deploy, vamos usar o **Surge**
- Para instalá-lo, rode: `npm install -g surge`
- Para fazer o deploy, rode (na pasta raiz do projeto, **depois** de fazer o build): `surge ./build`
- **COLOQUEM O LINK DO SURGE NO PR**

Resumo

Labenu_

Resumo 📓

Criar app React	`npx create-react-app` **`nome-do-app`**
Rodar app React	`npm run start`

- Arquivos relevantes: **App.js** e **App.css**

Resumo 📓

- **Componente:** função que retorna um JSX
- **JSX:** linguagem parecida com HTML misturada com Javascript
- Muitas coisas são comuns entre HTML e JSX mas ressaltamos **algumas diferenças**

Resumo 📓

- `class` ⇒ `className`
- `onclick` ⇒ `onClick` (eventos em camelCase)
- Imagens locais precisam ser **importadas**
- Declaramos função **no próprio componente**

Resumo 📓

- `onclick="apertou()"` ⇒ `onClick={apertou}`
- Regra do **pai único**
- Podemos referenciar expressões no JSX
- Self-closing tags: `<evento />`

Resumo 📓

Instalar surge (só 1a vez)	`npm install -g surge`
Buildar app React	`npm run build`
Deployar com surge	`surge ./build`

- **COLOQUEM O LINK DO SURGE NO PR**

Dúvidas? 

Labenu_

Labenu_

Obrigado(a)!