ReactJS

Programa
3000 TALENTOS TI

O que vamos ver hoje?

- Um pouco de história
- O que é ReactJS
- Configuração
- Estrutura
- Componentes
- Props
- CSS

Um pouco de história

Surgimento

- Foi criado pelo Meta (antiga empresa Facebook) em meados de 2011.
- Em 2013 se tornou Open Source e foi lançado oficialmente.
- Em 2015 grandes empresas como Netflix e Airbnb passam a adotar o React.

ReactJS o que é?

O que é ReactJS?

- React é uma biblioteca JavaScript para a criação de interfaces de usuário que utilizam o padrão SPA (Single Page Application).
    - SPA é um padrão em que seu carregamento dos recursos (como CSS, JavaScript e HTML das páginas) ocorre apenas uma única vez: na primeira vez em que o usuário acessa a aplicação. Isso faz com que o carregamento das novas páginas seja bem mais rápido.

O que é ReactJS?

- Com o React, as interfaces do usuário são compostas por pedaços de código isolados, chamados de "componentes".
- Isso torna o código mais fácil de dar manutenção e reutiliza o mesmo código para outras funcionalidades.

JSX

- O JSX é uma extensão de sintaxe JavaScript usada na criação do elemento React.
- Os desenvolvedores a empregam para incorporar código HTML em objetos JavaScript.
- Como o JSX aceita a incorporação de funções e expressões JavaScript válidas, ele pode simplificar estruturas de código complexas.

JSX

```
1  function App() {
2   return <h1>Sou um titulo</h1>;
3  }
4
5  export default App;
```

Virtual DOM

- O Document Object Model (DOM) apresenta uma página da web através de uma estrutura de árvore de dados.
- O ReactJS armazena árvores DOM virtuais na memória.
- Ao fazer isso, o React pode aplicar atualizações a partes específicas da árvore de dados, o que é mais rápido do que renderizar novamente toda a árvore DOM.

Virtual DOM

- Sempre que houver uma alteração nos dados, o ReactJS irá gerar uma nova árvore Virtual DOM e compará-la com a anterior para encontrar a maneira mais rápida possível de implementar alterações no DOM real.
- Este processo é conhecido como diffing.

Componentização

- O ReactJS divide a interface de usuário em pedaços de código isolados e reutilizáveis, conhecidos como componentes.
- Os componentes do React funcionam de maneira semelhante às funções JavaScript, pois aceitam entradas arbitrárias chamadas propriedades ou props.

Por que usar React?

- O React tem sido usado por grandes companhias ao redor do mundo.
- Algumas delas: Netflix, Airbnb, American Express, Facebook, WhatsApp, eBay e Instagram.
- Essa é a prova de que a ferramenta tem um número de vantagens que não têm nem comparação nos seus competidores.

Por que usar React?

1. Fácil de usar
2. Reusabilidade de componentes
3. Componentes fáceis de escrever
4. Melhor desempenho

Iniciando

Iniciando

- O pré-requisito é ter o Node.js instalado no computador.
- Por se tratar de uma biblioteca, o React precisa ser instalado no projeto.
- Instalamos digitando o seguinte comando no terminal:
  - npm init vite@latest nome-projeto – –template react

```
npm create vite@latest projeto-exemplo -- --template react
```

Iniciando

- Seguir as instruções:

```
Need to install the following packages:
  create-vite@latest
Ok to proceed? (y)
```

```
Done. Now run:

  cd projeto-exemplo
  npm install
  npm run dev
```

Iniciando

- Instalar as dependências do node (pode demorar):
  - npm install

```
added 266 packages, and audited 267 packages in 57s

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

- Após a instalação abrir o VSCode na pasta do projeto.

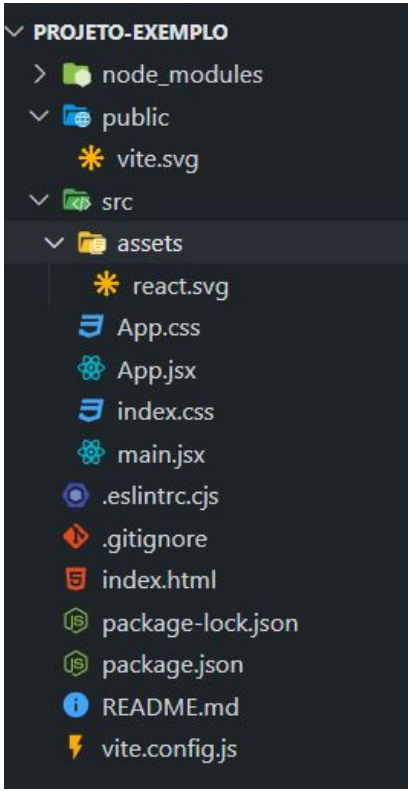
Iniciando

- Para rodar o projeto usar o comando:
  - npm run dev

```
VITE v4.4.11  ready in 830 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h to show help
```

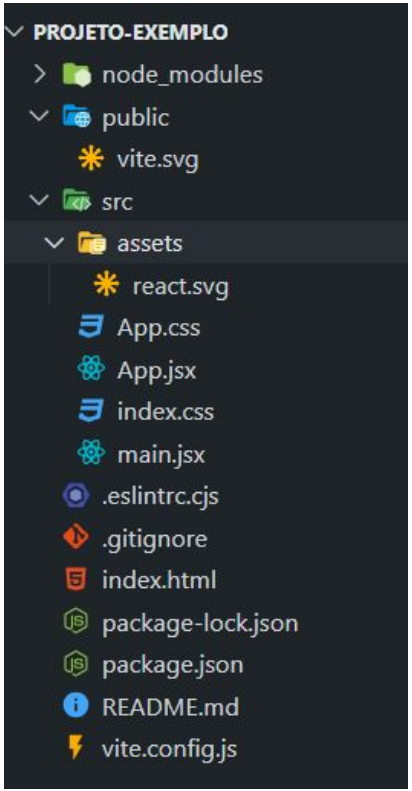
Estrutura do projeto

Estrutura do projeto



- node_modules: dependências do projeto.
- public: arquivos estáticos como fontes ou imagens.
- src: pasta principal onde os componentes devem ser criados.
- assets: pasta de imagens.
- index.html: arquivo principal onde é inserido o js e os componentes React (via main.jsx).
- main.jsx: Arquivo de entrada do React.
- App.jsx e App.css e index.css: Arquivos criados como exemplo e geralmente devem ser excluídos.

Estrutura do projeto

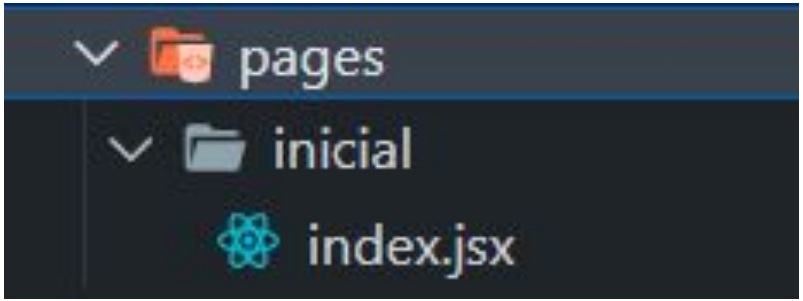


- .eslintrc.cjs: Regras de formatação do código. Como não faremos validação de tipos de dados, é necessário incluir o trecho "react/prop-types": 0 em rules.
- .gitignore: Lista de arquivos e/ou pastas que serão excluídas pelo git.
- package.json: Informações do projeto e das bibliotecas instaladas.
- vite.config.js: Configurações do compilador.

Criando um componente

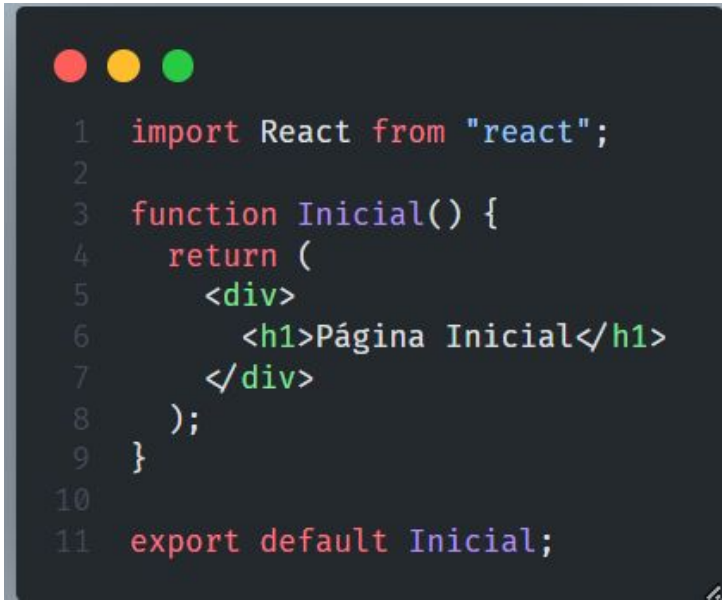
Criando um componente

- Apague os arquivos App.css, App.jsx e index.css
- Na pasta src crie uma pasta pages e dentro dela uma pasta inicial e dentro dela um arquivo chamado index.jsx



Criando um componente

- Insira o código abaixo:



```
1  import React from "react";
2
3  function Inicial() {
4    return (
5      <div>
6        <h1>Página Inicial</h1>
7      </div>
8    );
9  }
10
11 export default Inicial;
```

Criando um componente

- No arquivo main.jsx apague a importação do App.jsx e insira a do componente Inicial.

```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import Inicial from "./pages/inicial/index.jsx";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <Inicial />
  </React.StrictMode>
);
```

Criando um componente

- Na pasta src crie uma pasta chamada components
- Dentro da pasta components cria uma pasta card
- Dentro da pasta card crie um arquivo index.jsx

Criando um componente

- Insira o código abaixo:

```jsx
const Card = () => {
  return (
    <article>
      <img src="src/assets/react.svg" alt="imagem do card" />
      <div>
        <h2>Título de um post</h2>
        <p>Descrição do post</p>
      </div>
      <a href="www.github.com">Acessar post</a>
    </article>
  );
};

export default Card;
```

Criando um componente

- No index.jsx da page inicial insira a referência para o card:

```jsx
import Card from "../../components/card";

function Inicial() {
  return (
    <div>
      <h1>Página Inicial</h1>
      <Card />
    </div>
  );
}

export default Inicial;
```

Props

Props

- Em nosso exemplo criamos um card. Mas e se quiséssemos que nossa aplicação tivesse vários cards? Não é correto ficar replicando código. É aí que o React se destaca.
- Por meio do uso de props podemos fazer um componente que vai receber dinamicamente valores que serão utilizados.

Props

- Vamos transformar nosso card para que ele recebe props:

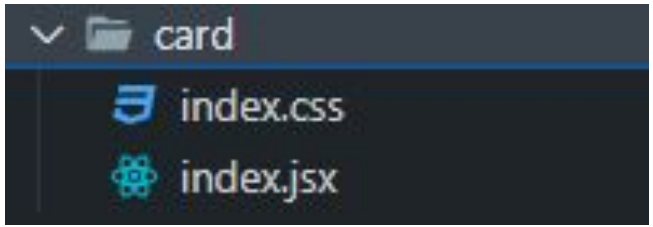
```jsx
const Card = ({ title, content }) => {
  return (
    <article>
      <img src="src/assets/react.svg" alt="imagem do card" />
      <div>
        <h2>{title}</h2>
        <p>{content}</p>
      </div>
      <a href="www.github.com">Acessar post</a>
    </article>
  );
};

export default Card;
```

```jsx
<Card title="Título do card" content="Conteúdo do card" />
```

CSS

CSS

- Funciona bem parecido com o Html.
- Classes são adicionadas utilizando className.
- Dentro da pasta card adicione um arquivo index.css:



CSS

- Dentro dele definimos as regras css para o componente:

```css
.card {
  background: rgba(19, 41, 61, 0.53);
  box-shadow: 0 8px 32px 0 rgba(70, 130, 180, 0.2);
  backdrop-filter: blur(20px);
  -webkit-backdrop-filter: blur(20px);
  border-radius: 10px;
  border: 1px solid rgba(70, 130, 180, 0.1);
  padding: 1rem;
  color: #fff;
  text-align: center;
  width: 40%;
}

.card h2 {
  padding: 0.5rem;
  font-size: 1.5rem;
  margin-bottom: 0.3rem;
  margin-top: 0.3rem;
  font-weight: bolder;
}

.card p {
  text-align: center;
  padding: 0.5rem;
  font-size: 1.5rem;
}
```

CSS

- No card.jsx adicionamos a referência ao arquivo e aplicamos a classe ao componente:

```jsx
import "./index.css";
const Card = ({ title, content }) => {
  return (
    <article className="card">
      <img src="src/assets/react.svg" alt="imagem do card" />
      <div>
        <h2>{title}</h2>
        <p>{content}</p>
      </div>
      <a href="www.github.com">Acessar post</a>
    </article>
  );
};

export default Card;
```

CSS

Página Inicial

Resumo

Resumo 📓

- Com React podemos criar componentes reutilizáveis.
- React aceita sintaxe Html, CSS e JS.

Dúvidas? 

Programa

3000 TALENTOS TI

Obrigado(a)!