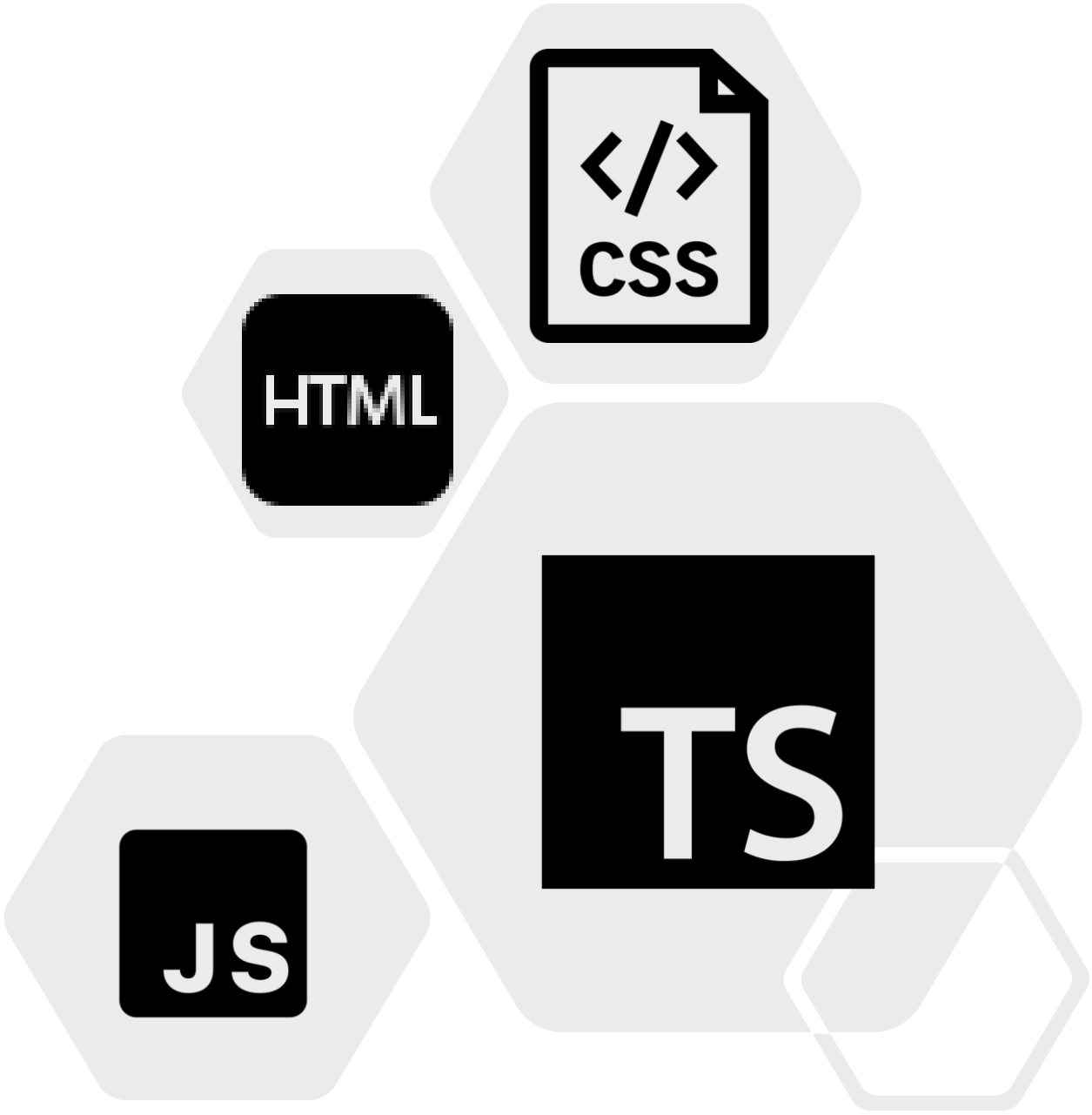


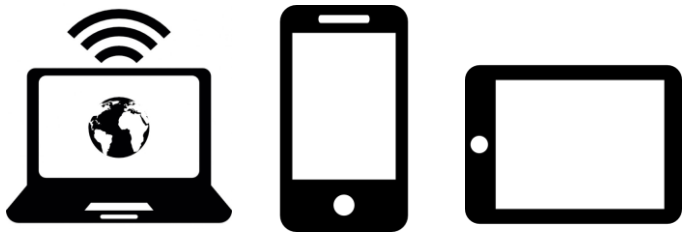
React e single page application

Aplicativos de página única

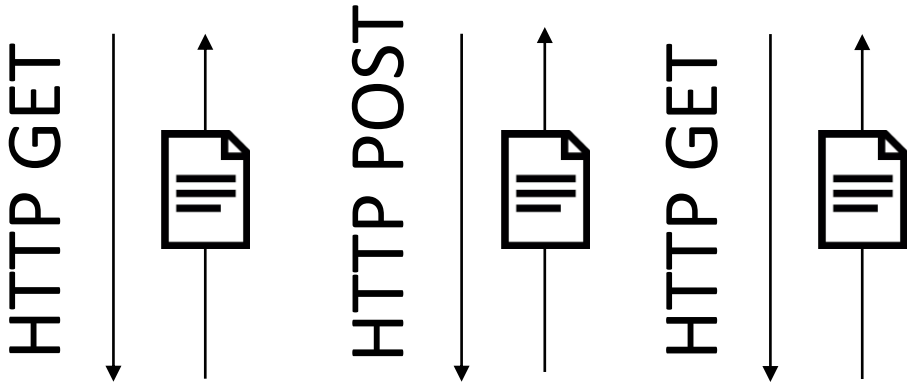
Um aplicativo de página única é uma aplicação web que consiste de uma única página com o objetivo de fornecer uma experiência do usuário similar à de um aplicativo desktop/mobile. O termo em inglês é single page application (SPA).

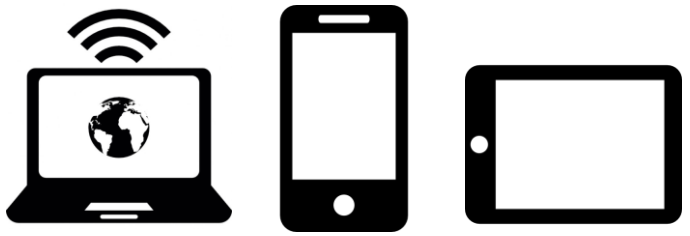
O aplicativo de página única interage com o usuário reescrevendo dinamicamente a página atual, em vez de carregar páginas novas inteiras do servidor.

Tradicional versus SPA

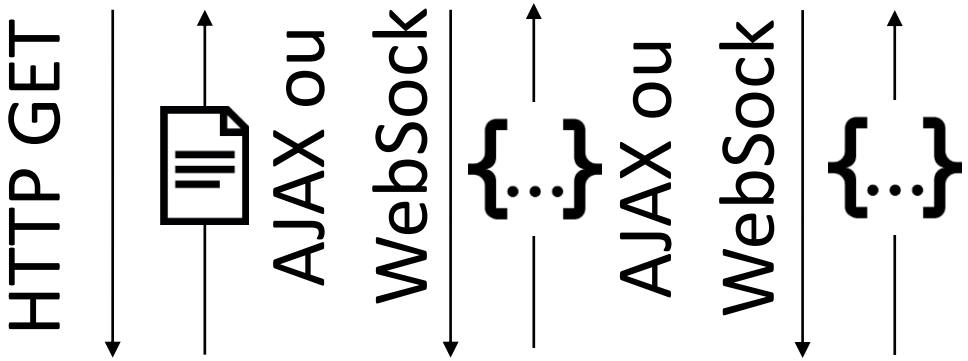


Interação com usuário

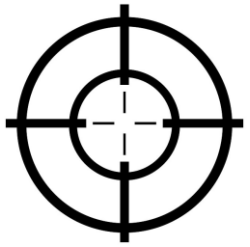




Interação com usuário



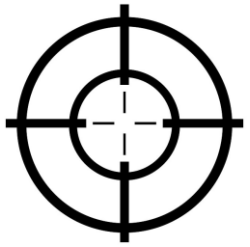
Arquitetura de SPA



Essa abordagem evita a interrupção da experiência do usuário entre páginas sucessivas, fazendo com que o aplicativo se comporte mais como um aplicativo de desktop/mobile.

Torna o tempo de carregamento mais rápido para os usuários, a quantidade de informações que um servidor precisa enviar muito menor e com muito mais economia.

Vantagens no desenvolvimento?



Na maioria dos sites e aplicações web, há muito conteúdo repetido. Alguns dos conteúdos permanecem iguais, como cabeçalhos, rodapés, barra de navegação e menus, ou seja, o layout se repete. Os SPAs aproveitam essa repetição, aplicando dinamismo apenas onde precisa.

O tempo de desenvolvimento do front-end, geralmente, é menor. O que facilita o desenvolvimento do front-end não é a arquitetura em si, são os frameworks disponíveis para construção de aplicações com SPA.

Quais as vantagens dos aplicativos de página única?

Os aplicativos de página única são uma ótima opção para criar experiências envolventes para seus usuários.

SPA é uma ótima alternativa para fazer design responsivo para desktops e dispositivos móveis.

Com SPA o servidor web não precisa gastar tempo e energia fazendo o desenho e envio completo da página para o usuário, isto reduz o impacto nos servidores em geral – economia de tempo, processamento e trânsito de dados.

Quais as desvantagens de SPA?

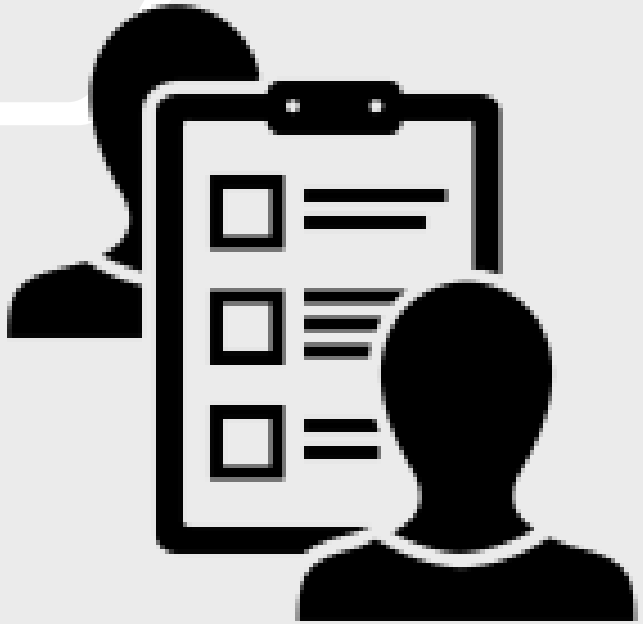
Altamente dependente de JavaScript. Os usuários precisam ativar o JavaScript no navegador ou o aplicativo não funcionará.

SPA são aplicativos que exigem trabalho de desenvolvimento para modificar a exibição e entrega da página, isto pode gerar um gargalo a cada ajuste, pois é necessário uma programação além do básico com HTML e CSS.

SPAs usam JavaScript, por isso pode ser difícil rastrear erros, dado que o erro pode ocorrer no momento da execução no navegador do usuário.

Quais empresas usam/usaram SPA?

- Alguns exemplos de aplicativos de página única são: Gmail, Google Maps, AirBNB, Netflix, Pinterest, Paypal entre outros. Muitas empresas estão usando SPA para construir uma experiência fluida e escalável.



Frameworks em destaque

Existem alguns frameworks que se destacam no desenvolvimento com SPA. Angular, React e Vue são os principais, os mais usados. Essas tecnologias são uma coleção de componentes reutilizáveis e extensíveis, que seguem um conjunto definido de regras de construção.

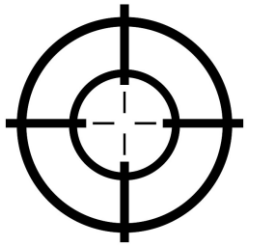
Grande parte dos frameworks disponíveis são mantidos ou recebem forte influência de empresas mundialmente conhecidas, como Google e Meta.

React

React é uma biblioteca JavaScript para construção de interfaces (front-end). Importante! Seus fabricantes e principais desenvolvedores não consideram o React como um framework, mas como uma biblioteca JavaScript.

React é um projeto de código aberto criado pelo Facebook (Meta). Atualmente a empresa é sua principal mantenedora e a biblioteca é utilizada nas interfaces de suas principais redes sociais.

React App



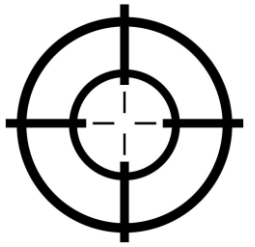
Comandos para criar um React App:

```
npm init react-app nome_app
npx create-react-app nome_app
```

Nome do diretório onde o aplicativo será criado. Importante! O diretório não precisa existir, ele será criado e, se for necessário, deve-se passar o caminho completo.

Os comandos fazem a mesma coisa. Eles dependem da versão do node.js instalado, o **npx** está disponível a partir da versão 5.2 e o **npm init** a partir da versão 6.0

React App

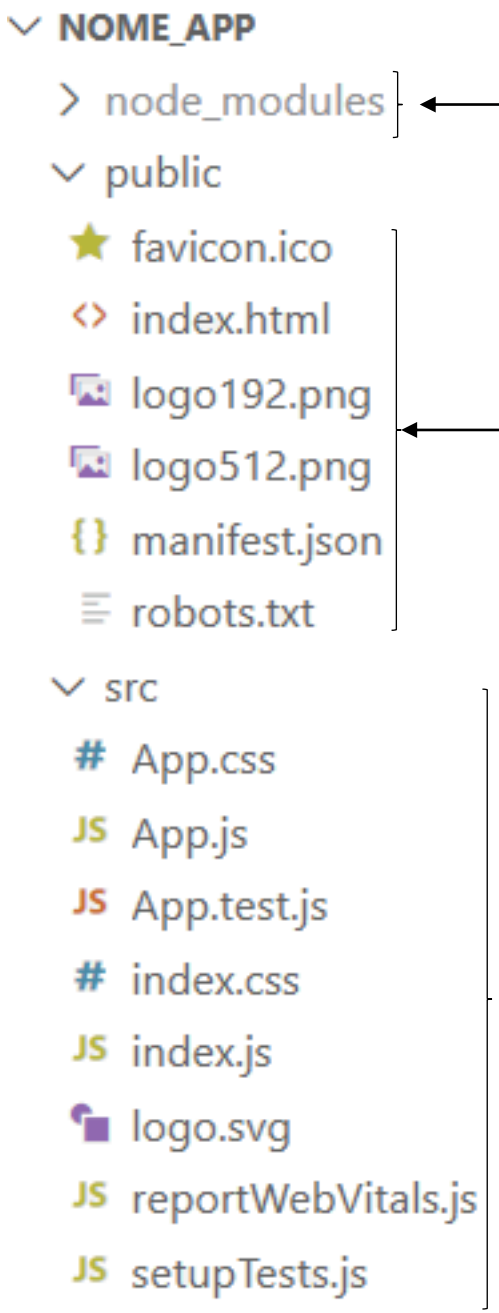


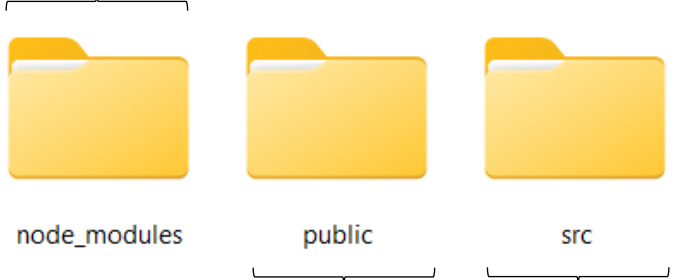
Comando para iniciar a aplicação:

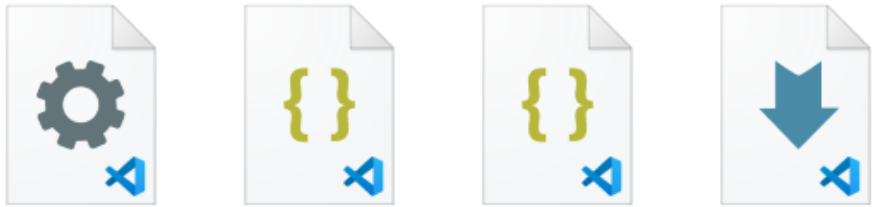
```
npm start
```

O comando iniciar um serviço web, com acesso padrão pelo endereço **http://localhost:3000/**

React App

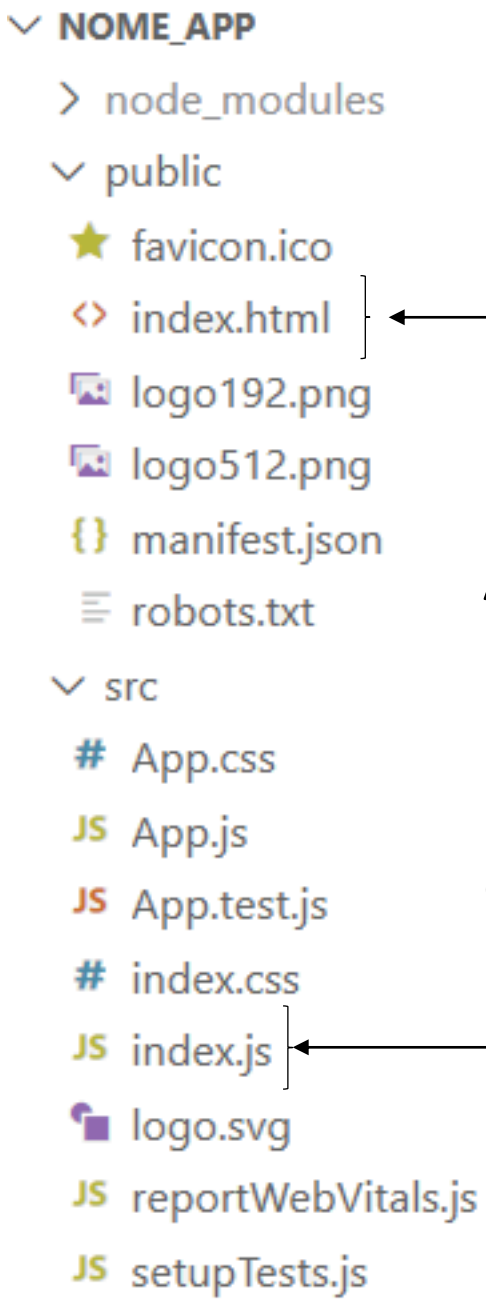






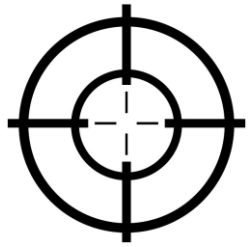
Arquivos criados para o aplicativo padrão. A lógica da aplicação e os códigos fontes, estão localizados no diretório src.

React App



A página única, ou seja, o arquivo HTML, está localizado no diretório **public**. Ela é o contêiner DOM. O arquivo index.js, localizado no diretório src, é responsável pela correlação entre o contêiner DOM e o código fonte.

Contêiner DOM



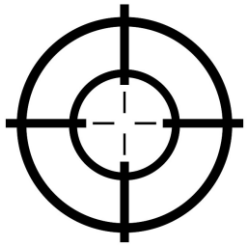
O contêiner DOM é a página HTML enviada ao cliente. Nela deve-se destacar qual ou quais as partes dinâmicas, correlacionadas com o código fonte.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
reportWebVitals();
```

```
<!DOCTYPE html>
<html lang="en">
  <head>…
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Contêiner DOM



O componente ReactDom é responsável por construir e aplicar o dinamismo dos demais componentes, que irão preencher a página única.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
reportWebVitals();
```

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">…
      </header>
    </div>
  );
}

export default App;
```

React App com TypeScript

Quando se utiliza o criador padrão de projeto, o React App criado vem configurado para funcionar apenas com a linguagem JavaScript.

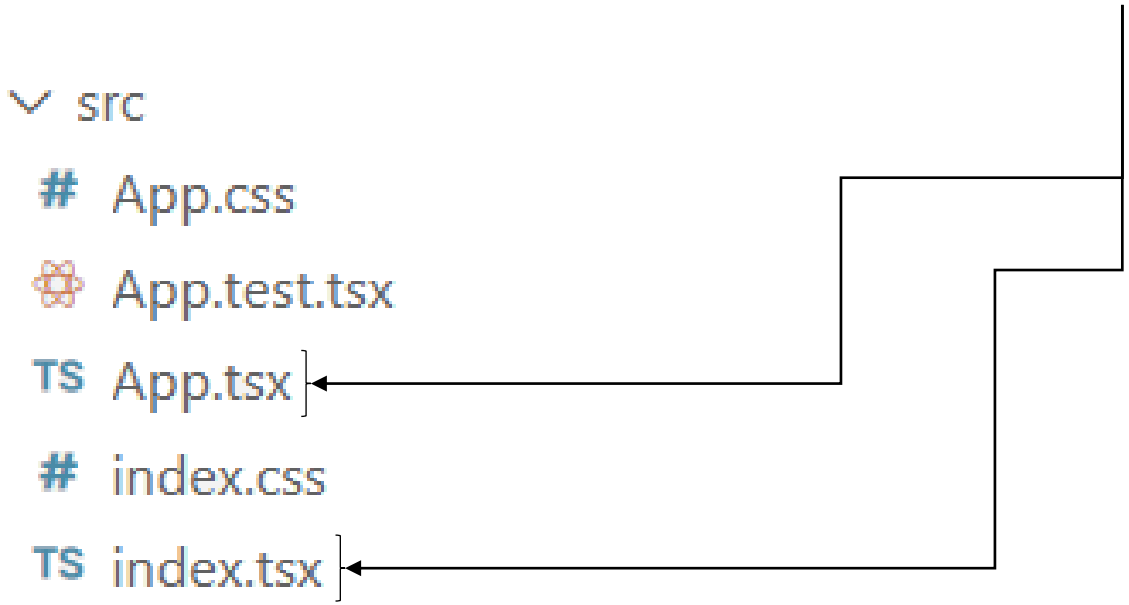
Comando para criar um React App, com TypeScript:

```
npx create-react-app nome_app --template typescript
```

A partir da versão 3.0 do React, adicionou-se a opção de criar aplicações com templates específicos, ou seja, com configurações específicas.

React App com TypeScript

Apesar do template configurado para TypeScript, não há diferença na estrutura do projeto criado. Contudo, existe diferença na extensão dos arquivos de código fonte, que passa a ser .tsx

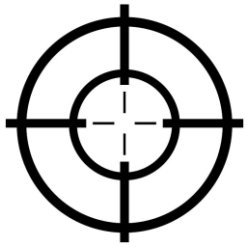


JSX e TSX

JSX (JavaScript Syntax Extension) é uma extensão da sintaxe da linguagem JavaScript, que fornece uma maneira de estruturar a renderização de componentes. JSX permite ao desenvolvedor escrever HTML no React. TSX é a sintaxe TypeScript do JSX.

Importante! A extensão “.tsx” deve ser usada somente em componentes, ou seja, nos arquivos de classes que irão renderizar algum conteúdo HTML.

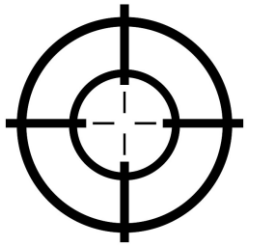
Componentes funcionais e de classe



Utilizar uma função JavaScript é, geralmente, a maneira mais simples de construir um componente. Contudo está não é a única maneira, com a linguagem TypeScript pode-se criar componentes com classes.

Importante! Há vantagens e desvantagens para as duas abordagens e depende de o desenvolvedor escolher qual a melhor forma para o contexto do seu desenvolvimento.

Função ou classe



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

function App() {
  return (
    <div className="App">
      <header className="App-header">…
      </header>
    </div>
  );
}

export default App;
```

```jsx
import logo from './logo.svg';
import './App.css';
import { Component } from 'react';

class App extends Component {
  render() {
    return (
    <div className="App">
      <header className="App-header">…
      </header>
    </div>
    )
  }
}

export default App;
```