
otterwise

TIPOS DE APLICAÇÃO

otterwise

SPA

- **Single-Page Application** - Aplicação de página única.
- Aplicação Web onde não há carregamento de páginas, todo o conteúdo é enviado pelo servidor ao entrar no endereço ou conforme o usuário for realizando ações.
- Grande ganho na experiência do usuário (UX) e no tempo de carregamentos dos recursos.
- Consome mais energia e banda pois o navegador tem maior demanda.
- SEO não otimizado pois sendo uma única página é difícil para os motores de busca identificarem os conteúdos internos.
- Exemplos: Github, Gmail, Twitter, etc...

MPA

- **Multi-page Application** - Aplicação de múltiplas páginas.
- Aplicação Web mais tradicional onde a cada navegação uma nova página é solicitada ao servidor e carregada.
- Melhor desempenho no SEO e mais leve para o navegador.
- Tempo de carregamento pode atrapalhar a experiência de usuário.
- Exemplos: Amazon, Medium, etc...

Fonte: https://asperbrothers.com/blog/spa-vs-mpa/

FRAMEWORKS E BIBLIOTECAS

otterwise

Frameworks e bibliotecas

- Pacote mais complexo e robusto que, além de adicionar novas funcionalidades, adiciona uma nova **estrutura** para o desenvolvimento.
- Facilitam o processo de desenvolver aplicações complexas tanto no lado do cliente quanto no lado do servidor.
- Fundamentais na criação de uma aplicação moderna.

Exemplos em JavaScript

- Front-end:
  - React JS (https://pt-br.reactjs.org/)
  - Angular (https://angular.io/)
  - Vue JS (https://vuejs.org/)
- Back-end:
  - Koa JS (https://koajs.com/)
  - Express JS (https://expressjs.com/pt-br/)
- Mobile:
  - React Native (https://reactnative.dev/)

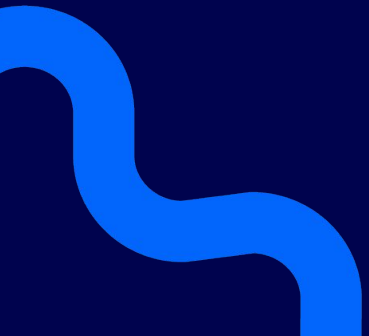
REACT

otterwise

React

- https://pt-br.reactjs.org/
- Biblioteca para desenvolvimento front-end na construção de sistemas Web.
- Construção baseada em componentes que são renderizados na tela.
- Combinamos componentes simples (como um campo de texto) em componentes complexos (como a página principal) para facilitar manutenção e leitura do código.
- Troca de informações entre os componente.
- Controle de estado dos componentes.
- Componentes possuem um ciclo de vida com momentos onde pode ser associado execuções de código.

CREATE REACT APP



otterwise

Create React App

- Create React App é um starter kit que cria uma estrutura básica com tudo que é necessário.
- Sem preocupação com webpack/parcel e outras configurações avançadas.
- Fácil de executar e de gerar versão de produção.

Rodando o Projeto

- Script *start* para rodarmos o projeto.
- O projeto agora representa uma aplicação, que não possui “fim”.
- Acesso via **localhost**.
- **Hot reload** permite que as alterações realizadas (e salvar) nos arquivos sejam executadas imediatamente na aplicação.

Buildando o projeto

- Estrutura final otimizada da aplicação para ser colocada em produção.
- Script *build* permite gerar essa estrutura final.
- Podemos usar pacotes como o **http-server** para testarmos a build da aplicação.

Exemplo:

- Rodar o comando yarn build
- Instalar o pacote http-server via npm e rodar o comando http-server dentro da pasta gerada pelo comando yarn build

EXERCÍCIO

Exercício

Criar seu projeto react com o create-react-app.

Rodar localmente o projeto.

Realizar mudança no texto da página (App.js) para testar o hot reload.

JSX

otterwise

JSX

- Extensão da sintaxe do JavaScript adicionada pelo React para descrever elementos da interface de usuário;
- Semelhante ao HTML;
- Expressões válidas com JSX:

```
// JSX em constantes
const paragrafo = <p>meu paragrafo</p>

// JSX como argumento de função e retornado por uma função
const retornaParagrafo = p => p
retornaParagrafo(paragrafo)

const retornaH1 = () => return <h1>Meu título</h1>
```

JSX

- Exemplo: **App.js.**
- Dentro do JSX pode-se utilizar as **chaves** para resolver uma **expressão JS**.
- Elementos React que representam tags do DOM.
- *null* é um JSX válido.

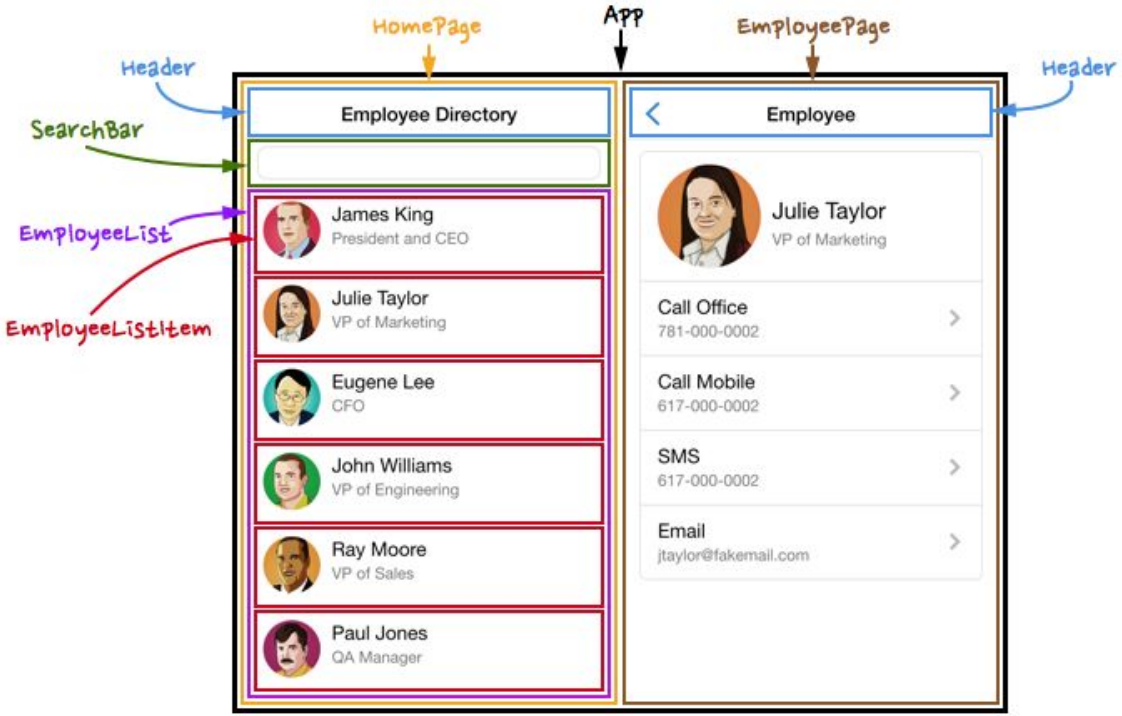
```
const retornaParagrafo = conteudo => <p>{conteudo}</p>
retornParagrafo("conteudo do paragrafo")
// Retorno: <p>conteudo do paragrafo</p>
```

COMPONENTES

Componentes

- Permite a divisão da UI em **partes independentes**.
- Torna o código reutilizável, além de facilitar a legibilidade e manutenibilidade.
- É possível realizar a troca de informação entre componentes.
- Cada componente pode manter um estado.
- Retornam **elementos React (JSX)** que devem aparecer na tela.
- Podem ser descritos como uma **função** (Function Component) ou uma **classe** (Class Component).

Exemplo de componentização de um site



otterwise

Props (Propriedades do componente)

- Informações de entrada de um componente.
- Em um Function Component as props serão o único parâmetro da função e será sempre um objeto.
- Facilita a criação de componentes reusáveis.
- Quando alteradas atualizam o componente.

```
// Exemplo de componente
const Title = props => {
  return <h1>{props.children}</h1>
}

// Utilização: <Title>Meu título</Title>
```

RENDERIZAÇÃO CONDICIONAL

otterwise

Renderização Condicional

- Técnica para definir se um componente será renderizado ou não na tela.
- Pode ser realizada com uma operação lógica AND.
- Caso a condição seja **verdadeira** o componente **é** renderizado na tela.
- Caso a condição seja **falsa** o componente **não é** renderizado na tela.

ESTILIZAÇÃO

otterwise

Estilização

- Utilizamos className e não class para incorporar classes no JSX;
- Arquivos de estilo são criados e importados separadamente;

```
import "./styles.css"

const Title = props => {
  return <h1 className="title">{props.children}</h1>
}
```

EXERCÍCIO

Exercício

1. Crie uma página com uma lista de artigos.
2. Essa página deve estar dividida em três seções: Header, Main e Footer.
3. No Header deve haver o título da página.
4. No Main devem ser renderizados os artigos com um título e uma descrição.
5. No Footer coloque um texto indicando quem desenvolveu a página.

Os textos e a quantidade de artigos é de decisão de vocês.

Dicas de componentes para serem criados: Header, Main, Footer e Article.