REACT: Prop Drilling e Context API

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

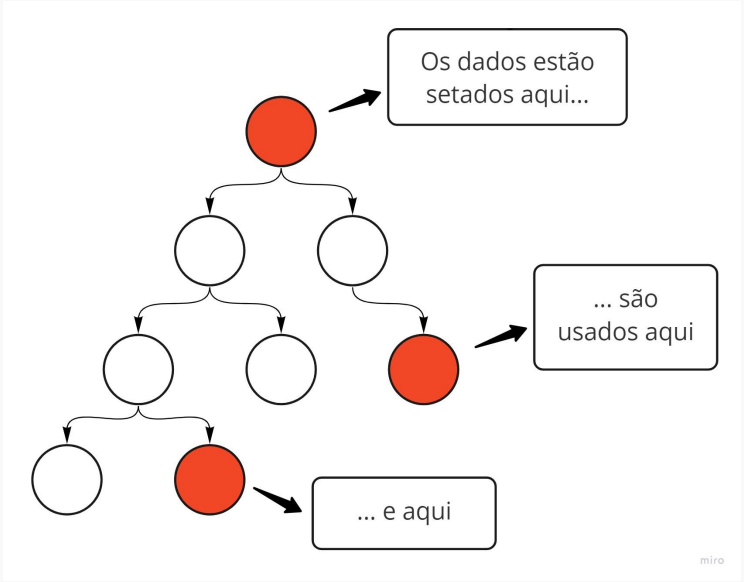
AGENDA

- Prop Drilling
- Context API

PROP DRILLING

Acontece quando precisamos obter dados que estão em várias camadas na **árvore de componentes** React.

Nesse caso as informações são passadas por **props**, que por vezes não são usadas pelo componente logo abaixo na árvore, mas em níveis mais abaixo.



PROP DRILLING

#Exemplo: **não** é prop drilling

```jsx
function App() {
  const [tipoDePlano, setTipoDePlano] = useState("free");
  const nomeDoUsuario = "Rosana Rezende";

  return (
    <div className="App">
      <Cabecalho nome={nomeDoUsuario} />
      {tipoDePlano === "free" ? (
        <Gratuito
          alteraTipoDePlano={setTipoDePlano}
          nome={nomeDoUsuario}
        />
      ) : (
        <Pago
          alteraTipoDePlano={setTipoDePlano}
          nome={nomeDoUsuario}
        />
      )}
    </div>
  );
}
```

```jsx
export const Cabecalho = ({ nome }) => {

  return (
    <header>
      <img src={logo} alt="Clamedfy" />
      <h1>{nome}</h1>
    </header>
  )
}
```

PROP DRILLING

#Exemplo: é prop drilling

```jsx
function App() {
  const [tipoDePlano, setTipoDePlano] = useState("free");
  const nomeDoUsuario = "Rosana Rezende";

  return (
    <div className="App">
      <Cabecalho nome={nomeDoUsuario} />
      {tipoDePlano === "free" ? (
        <Gratuito
          alteraTipoDePlano={setTipoDePlano}
          nome={nomeDoUsuario}
        />
      ) : (
        <Pago
          alteraTipoDePlano={setTipoDePlano}
          nome={nomeDoUsuario}
        />
      )}
    </div>
  );
}
```

```jsx
export const Gratuito = ({ alteraTipoDePlano, nome }) => {
  return (
    <>
      <BemVindo nome={nome} />
      <>...
      </>
    </>
  );
};
```

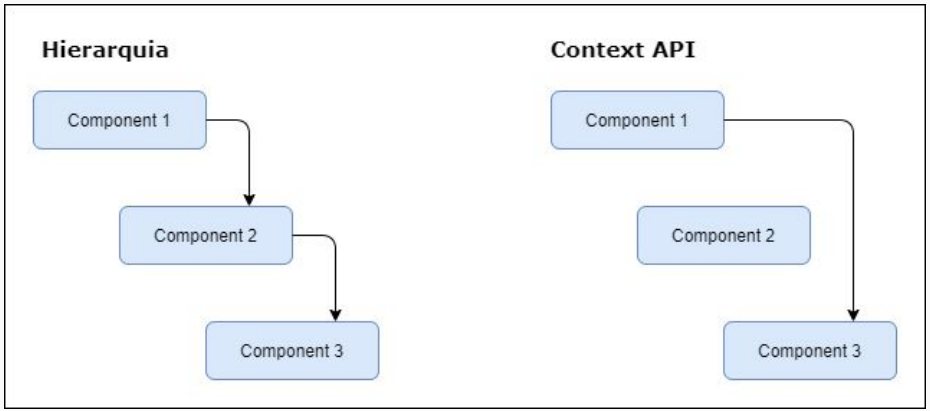
```jsx
export const BemVindo = ({ nome }) => {
  return (
    <h2 className="BemVindo">
      Boas Vindas ao Clamedfy <span>{nome}</span>
    </h2>
  );
};
```

Como evitar prop drilling?

CONTEXT API

Uma das formas de evitar prop drilling no React é através do compartilhamento de “estados globais” através da **CONTEXT API**.

Poderemos compartilhar propriedades entre componentes que não fazem parte da mesma hierarquia.



CONTEXT API

Para isso utiliza-se **contextos**.

Detalhes:

- O uso de estados globais deve ser moderado, pois dificulta a reutilização de componentes
- Sempre que possível, quebre em pequenos contextos, pois quando um valor do contexto é atualizado todos os componentes ligados a ele serão renderizados novamente
- Indicado em casos como:
  - tema
  - idioma
  - se o usuário está autenticado

CONTEXT API

Utilizaremos a função **createContext()** do React.

```
import { createContext } from "react";
```

Com ele criaremos o contexto (que pode ou não ter um valor inicial)

```
const UsuarioContext = createContext();
```

```
const UsuarioContext = createContext({ nome: "João" });
```

CONTEXT API

Em seguida construiremos um **Provider**, que fornecerá os dados que serão consumidos por todos os componentes que precisarem.

O Provider deverá "**abraçar**" todos os descentes que consumirão seus valores

```
const UsuarioProvider = ({ children }) => {
  const [tipoDePlano, setTipoDePlano] = useState("free");

  return (
    <UsuarioContext.Provider
      value={{
        nome: "Rosana Rezende",
        tipoDePlano,
        alteraTipoDePlano: (tipoDePlano) => {
          setTipoDePlano(tipoDePlano);
        },
      }}
    >
      {children}
    </UsuarioContext.Provider>
  );
};
```

```
import { UsuarioProvider } from './context/usuarioContext';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <UsuarioProvider>
      <App />
    </UsuarioProvider>
);
```

CONTEXT API

Em seguida construiremos **Consumers** para cada componente que necessite usar as propriedades do contexto global.

```jsx
import { UsuarioContext } from "./context/usuarioContext";

function App() {
  return (
    <UsuarioContext.Consumer>
      {value => (
        <div className="App">
          <Cabecalho />
          {value.tipoDePlano === "free" ? <Gratuito /> : <Pago />}
        </div>
      )}
    </UsuarioContext.Consumer>
  );
}
```

```jsx
import { UsuarioContext } from "../../context/usuarioContext";

export const Cabecalho = () => {
  return (
    <UsuarioContext.Consumer>
      {value => (
        <header>
          <img src={logo} alt="Clamedfy" />
          <h1>{value.nome}</h1>
        </header>
      )}
    </UsuarioContext.Consumer>
  );
};
```

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40

EXERCÍCIO 01 - Setup Inicial

Durante a semana 06, recebemos uma demanda da **M1P1 Software House Ltda** para construirmos um sistema de base de conhecimento chamado **DEVInKnowledge**.

Agora chegou a hora de implementarmos este sistema utilizando os conhecimentos adquiridos até aqui nas aulas de React.

Para esta aplicação, você pode:
- utilizar o mesmo código do seu projeto desenvolvido durante a avaliação
- ou fazer uma nova implementação completamente do zero

PASSOS INICIAIS:

- Crie uma aplicação React chamada **devinknowledge**.
- Prepare a aplicação, instalando o **prop-types**.
- Modifique o **index.html** e adicione no título o nome da aplicação DEVInKnowledge.

EXERCÍCIO 02 - Trazendo o HTML para JSX

Como primeira tarefa, vamos criar as estruturas iniciais da aplicação.

- Crie, dentro de **src**, uma pasta components e uma pasta pages.
- Crie, dentro da página **pages**, um componente Home.
  Traga todo o código HTML da sua aplicação para este componente.
- Traga os seus **estilos CSS** para a aplicação e importe o arquivo no **index.js**
- Apague o conteúdo padrão do **App.js** e importe o componente **Home**.
  Por enquanto, o App.js deve apenas retornar o Home.js.

OBS1: A estrutura do seu componente fica a seu critério. Uma dica é criar uma pasta com o nome do componente, um arquivo `.js` com o nome do componente e um arquivo `index.js` que faz a exportação do conteúdo do arquivo `.js`.

OBS2: Ao rodar a aplicação, o seu projeto deve abrir normalmente, sem as funcionalidades implementadas com JavaScript. A partir do código no componente ***Home***, vamos extrair os demais componentes.

EXERCÍCIO 03 - Criando os componentes

Vamos começar a quebrar nossa aplicação em componentes. A princípio, vamos criar os arquivos e, depois, trazer o código JSX de cada pedaço da aplicação (por enquanto podemos deixar cada componente retornando apenas uma ***div com algum texto***).

Dentro da pasta **components**, crie os componentes: Sidebar, MainContent, Button, Summary, SummaryCard, Filter, CardList, TipCard.

Após criar os componentes:

- Extraia **somente** o código JSX responsável pela renderização da sidebar para o componente Sidebar
- Extraia **somente** o código JSX responsável pelo conteúdo restante do projeto para o componente MainContent
- Importe e utilize os componentes Sidebar e MainContent dentro do componente Home.

OBS: Após essas modificações, a aplicação ainda deve funcionar corretamente (sem as funcionalidades do javascript, claro)

EXERCÍCIO 04 - Quebrando o código em componentes

Vamos extrair mais alguns componentes a partir do código que agora está no **MainContent**.

- Extraia todo o código responsável por renderizar a lista de cards de resumo (os que exibem a quantidade de dicas por categoria) para o componente **Summary**
- Deste código extraído, dentro do Summary, extraia o código responsável pela renderização de um único card de resumo para o componente **SummaryCard**
    - O componente SummaryCard vai receber duas props: ***title*** e ***count***. Faça com que as props sejam renderizadas no lugar do *título* e *números* que estavam hardcoded no HTML.
    - Não se esqueça de usar prop-types!

Insira alguns **SummaryCard**, com *dados falsos* passados como prop, no componente **Summary** para verificar se a tela está sendo renderizada corretamente.

Por último, insira o componente **Summary** no componente **MainContent**.

OBS: A página deve renderizar normalmente.

EXERCÍCIO 05 - Mais componentes (PARTE 1)

Agora é a vez do componente **Filter** e **TipCard** sairem do **MainContent**.

FILTER:

- Extraia todo o código JSX do campo de texto e botões de filtro para o componente **Filter**
- Insira o componente Filter no **MainContent**

TIPCARD:

- Extraia o código responsável pela renderização de um card completo para o componente **TipCard**. Neste ponto, se você criava os cards de forma dinâmica através do JS na sua aplicação, o JSX do componente deve ser criado com base no HTML gerado pela sua aplicação no momento da adição de um card.
- TipCard deve receber, como ***props***, os dados de uma *dica*. Você pode receber as props separadamente (`titulo`, `descricao`, `categoria`, `linguagem`, `video`) ou receber um objeto **dica** com todas estas propriedades.
- Remova o antigo código dos cards de **MainContent** e insira algumas instancias de TipCard para simular a renderização dos cards.
- Substitua os valores *hardcoded* dos dados da dica pelo valor recebido via props no TipCard

EXERCÍCIO 05 - Mais componentes (PARTE 2)

Composição de componentes:

Para trabalharmos com composição, vamos fazer com que o nosso componente **CardList** retorne um elemento com o **children** sendo os **TipCards**.

- Modifique o **CardList** para que ele receba children como props.
- Retire do componente **MainContent** o elemento responsável por agrupar a lista de cards (provavelmente um `ul` ou uma `div`)
- Faça com que o **CardList** retorne este elemento e, dentro dele, retorne o `children`.
- Não se esqueça de fazer a correta tipagem do `children` com o prop-types.

Um exemplo de implementação:

```
export const CardList = ( { children } ) => {
  return (
    <ul> {children} </ul>
  )
}
```

Modifique o **MainContent** para que a lista de cards **TipCard** seja passada dentro do componente **CardList**

MATERIAL COMPLEMENTAR

- API de Referência dos Hooks – React
- Context – React

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>