INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

ENSINO MÉDIO INTEGRADO - INFORMÁTICA
Disciplina de Desenvolvimento Web
Aula 13: React JS - Context API e Custom Hooks

Gil Eduardo de Andrade

Conceitos Preliminares

(https://react.dev/)
(https://playcode.io/react)

Custom Hooks (Hooks Personalizados)
Os Hooks personalizados (*ou custom hooks*) são funções JavaScript que permitem extrair e isolar a lógica de estado dos componentes React. Eles facilitam a reutilização de comportamentos complexos em vários componentes, mantendo a interface visual limpa e separada das regras de negócio.

Reutilizando a lógica - Hooks Personalizados
O React já possui um conjunto de Hooks integrados nativamente, como, por exemplo, "***useState***", "***useContext***" e "***useEffect***". Contudo, existem cenários em que há a necessidade de utilizar um Hook para uma finalidade mais específica, como, por exemplo, buscar dados ou verificar se o usuário está online.
Esse tipo de Hook (funcionalidade) não é encontrado, nativamente, no React, mas pode ser criado para atender às necessidades da aplicação.

Como eles funcionam?
Um hook personalizado é basicamente uma função cujo nome obrigatoriamente começa com a palavra "***use***" - "***useForm***", "***useWindowSize***". Dentro dele, você pode utilizar qualquer um dos hooks nativos do React ("***useState***", "***useEffect***").

Exemplo: Custom Hook

```JavaScript
import { useState } from 'react';

export function useTheme() {
  const [theme, setTheme] = useState('dark');
```

```
  const toggleTheme = () => {
    setTheme((prev) => (prev === 'dark' ? 'light' : 'dark'));
  };

  return { theme, toggleTheme };
}
```

Context API - Gerenciamento de Estados Globais
A Context API é uma funcionalidade nativa do React projetada para resolver o problema de prop drilling - prática de passar dados manualmente (via props) por vários níveis de componentes intermediários. Ela facilita o gerenciamento de estados globais, como temas visuais, dados de usuário ou carrinhos de compras, permitindo que qualquer componente acesse e atualize informações compartilhadas de forma direta.
Em outras palavras, a Context API gerencia e compartilha estados entre componentes sem precisar passar props manualmente por cada nível. Ela cria um estado global acessível a qualquer componente "filho", facilitando a comunicação em toda a árvore de elementos.

Funcionamento da Context API
A arquitetura do Contexto baseia-se em três pilares fundamentais:

- **createContext**: responsável por criar o contexto que será compartilhado.
- **Provider**: componente que fornece ("provê") o estado a ser compartilhado com todos os componentes dentro dele.
- **useContext**: hook que permite, aos componentes filho do provider, consumir o estado compartilhado.

Sempre que o valor dentro do Provider muda, todos os componentes que consomem esse contexto são renderizados novamente de forma automática com o dado atualizado.

Alternativas e Boas Práticas
Enquanto o Context é excelente para dados estáticos ou atualizações menos frequentes (como idioma ou tema), o uso excessivo em estados que mudam o tempo todo pode causar re-renderizações na árvore inteira. Para estados

complexos, muitos desenvolvedores utilizam bibliotecas como Zustand ou Redux. Para aprimorar o código, também é comum combinar o Context com useReducer.

Tabela Comparativa

Gerenciador de Estados	Arquitetura	Curva de Aprendizado	Impacto na Performance	Quando Usar
Context API	Nativa Provedor / Consumidor	Baixa (Muito simples)	Médio (Pode re-renderizar filhos)	Dados estáticos ou de baixa frequência (ex: temas, idioma, login)
Redux (Toolkit)	Flux Centralizada Global	Alta (Muito boilerplate)	Baixo (Altamente otimizado)	Aplicações corporativas grandes com lógica de estado complexa
Zustand	Atômica Baseada em Hooks	Baixa (Direta e limpa)	Baixo (Renderiza por seleção)	Projetos de médio a grande porte que exigem rapidez e performance.
Recoil / Jotai	Atômica Atoms e Selectors	Média	Baixo (Renderiza por átomo)	Telas com muitos elementos independentes (ex: ferramentas de desenho).

Exemplo: ThemeContext

JavaScript

```
import React, { createContext, useContext } from 'react';
import { useTheme } from './useTheme';

// Cria o contexto
const ThemeContext = createContext();

// Cria o Provider
export default function ThemeProvider({ children }) {
```

```
  const themeData = useTheme(); // Alimenta o contexto com a lógica do hook

  return (
    <ThemeContext.Provider value={themeData}>
      {children}
    </ThemeContext.Provider>
  );
}

// Cria o hook de consumo
export function useThemeContext() {
  const context = useContext(ThemeContext);
  if (!context) {
    throw new Error('useThemeContext deve ser usado dentro de um ThemeProvider');
  }
  return context;
}
```

Juntando os Exemplos numa Única Aplicação

Arquivo: useTheme.js

JavaScript

```
import { useState } from 'react';

export function useTheme() {
  const [theme, setTheme] = useState('dark');

  const toggleTheme = () => {
    setTheme((prev) => (prev === 'dark' ? 'light' : 'dark'));
  };

  return { theme, toggleTheme };
}
```

Arquivo: ThemeContext.js

JavaScript

```
import React, { createContext, useContext } from 'react';
```

```
import { useTheme } from './useTheme';

// Cria o contexto
const ThemeContext = createContext();

// Cria o Provider
export default function ThemeProvider({ children }) {
  const themeData = useTheme(); // Alimenta o contexto com a lógica do
hook

  return (
    <ThemeContext.Provider value={themeData}>
      {children}
    </ThemeContext.Provider>
  );
}

// Cria o hook de consumo
export function useThemeContext() {
  const context = useContext(ThemeContext);
  if (!context) {
      throw new Error('useThemeContext deve ser usado dentro de um
ThemeProvider');
  }
  return context;
}
```

Exemplo: ThemeButton.jsx

JavaScript

```
import React from 'react'
import { useThemeContext } from './ThemeContext';

export function ThemeButton() {
  // Usa o próprio hook
  const { theme, toggleTheme } = useThemeContext();

  return (
    <button
      onClick={toggleTheme}
      style={{
```

```
        background: theme === 'dark'
          ? '#333'
          : '#fff',
        color: theme === 'dark'
          ? '#fff'
          : '#333',
        padding: '10px 20px',
        cursor: 'pointer'
      }}
    >
      Mudar para o modo {theme === 'dark' ? 'claro' : 'escuro'}
    </button>
  );
}
```

Arquivo: App.jsx

JavaScript

```
import React from 'react'
import ThemeProvider from './ThemeContext';
import { ThemeButton } from './ThemeButton'

// Main App
export default function App() {
  return (
    <ThemeProvider>
      <main style={{ textAlign: 'center', marginTop: '50px' }}>
        <h1>Minha Aplicação</h1>
        <ThemeButton />
      </main>
    </ThemeProvider>
  )
}
```

FINALIZANDO A APLICAÇÃO - AULA ANTERIOR

PASSO A PASSO

1. **Clonando Repositório**

- Vite-React-Tailwind-Shadcn (Aplicação - Aula Anterior)
  https://github.com/Instituto-Federal-PR/vite-react-app-aula01

- Vite-React-Tailwind-Shadcn (Aplicação Pronta)
  https://github.com/Instituto-Federal-PR/vite-react-app-aula02

No terminal de comando

Shell
```
git clone https://github.com/Instituto-Federal-PR/vite-react-app-aula01
```

2. Criando Custom Hooks - Favoritos

- Criar o diretório ***"scr/hooks"***
- Criar o arquivo ***"src/hooks/useFavorites.ts"***

Arquivo: "src/hooks/useFavorites.ts"

TypeScript
```typescript
import { useState, useEffect } from 'react';
import type { Movie } from '@/types';

export function useFavorites() {

  const [favorites, setFavorites] = useState<Movie[]>(() => {
    // Garante que roda no ambiente do navegador
    if (typeof window !== 'undefined') {
      const stored = localStorage.getItem('favorites');
      if (stored) {
        try {
          return JSON.parse(stored);
        } catch (error) {
          console.error('Erro ao carregar favoritos:', error);
        }
      }
    }
    return []; // se não houver nada salvo
  });

  useEffect(() => {
    // Salva quando favoritos muda
```

```
    localStorage.setItem('favorites', JSON.stringify(favorites));
  }, [favorites]);

  const addFavorite = (movie: Movie) => {
    setFavorites((prev) => {
      if (!prev.find((m) => m.id === movie.id)) {
        return [...prev, movie];
      }
      return prev;
    });
  };

  const removeFavorite = (movieId: number) => {
    setFavorites((prev) => prev.filter((m) => m.id !== movieId));
  };

  const isFavorite = (movieId: number) => {
    return favorites.some((m) => m.id === movieId);
  };

  return {
    favorites,
    addFavorite,
    removeFavorite,
    isFavorite,
  };
}
```

3. Criando o Context - Favoritos

- Criar o diretório ***"scr/contexts"***
- Criar o arquivo ***"src/contexts/FavoritesContext.ts"***

Arquivo: "src/contexts/FavoritesContext.ts"

TypeScript

```
import React, { createContext, useContext, ReactNode } from 'react';
import { useFavorites } from '@/hooks/useFavorites';
import type { Movie } from '@/types';

interface FavoritesContextType {
```

```
  favorites: Movie[];
  addFavorite: (movie: Movie) => void;
  removeFavorite: (movieId: number) => void;
  isFavorite: (movieId: number) => boolean;
}

const FavoritesContext = createContext<FavoritesContextType |
undefined>(undefined);

export function FavoritesProvider({ children }: { children: ReactNode
}) {
  const favorites = useFavorites();

  return (
    <FavoritesContext.Provider value={favorites}>
      {children}
    </FavoritesContext.Provider>
  );
}

export function useFavoritesContext() {
  const context = useContext(FavoritesContext);
  if (!context) {
    throw new Error('useFavoritesContext deve ser usado dentro de
FavoritesProvider');
  }
  return context;
}
```

Shell

```
docker compose exec app npm install wouter
```

4. Instalando o componente Badge / Shadcn

No terminal de comando
(dentro da pasta do projeto)

```shell
docker compose exec app npx shadcn@latest add badge
```

5. Criando a página de Detalhes do Filme

- Criar o arquivo ***"src/pages/MovieDetails.tsx"***

Arquivo: "src/pages/MovieDetails.tsx"

```typescript
import { useParams, useLocation } from 'wouter';
import { ArrowLeft, Heart, Play } from 'lucide-react';
import Header from '@/components/Header';
import { Button } from '@/components/ui/button';
import { Card } from '@/components/ui/card';
import { Badge } from '@/components/ui/badge';
import { useFavoritesContext } from '@/contexts/FavoritesContext';
import type { Movie } from '@/types';
import moviesData from '@/data/movies.json';

export default function MovieDetails() {

  const params = useParams<{ id: string }>();
  const [, navigate] = useLocation();
  const { isFavorite, addFavorite, removeFavorite } = useFavoritesContext();

  const movieId = parseInt(params?.id || '0');
  const movie = moviesData.find((m) => m.id === movieId) as Movie | undefined;

  if (!movie) {
    return (
      <div className="min-h-screen bg-black flex flex-col items-center
justify-center">
        <Header />
        <div className="text-center">
          <h1 className="text-4xl font-black text-white mb-4">Filme não
encontrado</h1>
          <Button
            onClick={() => navigate('/')}
            className="flex items-center gap-2 px-6 py-3 bg-red-600 text-white
font-bold rounded-lg hover:bg-red-700 transition-all duration-200 mx-auto"
          >
            <ArrowLeft size={20} />
            Voltar para Home
```

```
          </Button>
        </div>
      </div>
    );
  }

  const favorite = isFavorite(movie.id);

  const handleFavorite = () => {
    if (favorite) {
      removeFavorite(movie.id);
    } else {
      addFavorite(movie);
    }
  };

  return (
    <div className="min-h-screen bg-black">
      <Header />

      <div className="relative w-full h-96 bg-neutral-900 overflow-hidden
pt-16">
        <img
          src={movie.fundo}
          alt={movie.titulo}
          className="w-full h-full object-cover"
        />
        <div className="absolute inset-0 bg-gradient-to-b from-transparent
via-black/50 to-black" />
      </div>

      <section className="container pb-20 -mt-24 relative z-10">
        <Button
          onClick={() => navigate('/')}
          variant="ghost"
          size="icon"
          className="absolute top-0.5 left-1 bg-black/50 hover:bg-black
text-white rounded-full transition-all duration-200 z-10"
          aria-label="Voltar"
        >
          <ArrowLeft size={24} />
        </Button>

        <Card className="bg-neutral-900 border-neutral-800 rounded-lg p-8">
          <div className="grid grid-cols-1 md:grid-cols-3 gap-8">
            <div className="flex justify-center md:justify-start">
```

```
              <img
                src={movie.imagem}
                alt={movie.titulo}
                className="w-72 h-96 object-cover rounded-lg border
border-neutral-700"
              />
            </div>

            <div className="md:col-span-2">
              <h1 className="text-5xl font-black text-white mb-4">
                {movie.titulo}
              </h1>

              <div className="flex flex-wrap gap-4 mb-6">
                <div className="flex items-center gap-2">
                  <span className="text-2xl font-bold text-red-600">★
{movie.rating}</span>
                  <span className="text-neutral-400">/10</span>
                </div>
                <div className="text-neutral-400">
                  <span className="font-semibold">{movie.ano}</span>
                </div>
                <div className="text-neutral-400">
                  <span className="font-semibold">{movie.duracao}
minutos</span>
                </div>
              </div>

              <div className="mb-6">
                <p className="text-neutral-400 mb-2">Diretor</p>
                <p className="text-xl font-semibold
text-white">{movie.diretor}</p>
              </div>

              <div className="mb-6">
                <p className="text-neutral-400 mb-2">Gêneros</p>
                <div className="flex flex-wrap gap-2">
                  {movie.genero.map((g) => (
                    <Badge
                      key={g}
                      variant="secondary"
                      className="px-3 py-1 bg-neutral-800 text-neutral-200
rounded-full text-sm"
                    >
                      {g}
                    </Badge>
```

```
                ))}
              </div>
            </div>

            <div className="mb-8">
              <p className="text-neutral-400 mb-2">Sinopse</p>
              <p className="text-lg text-neutral-300 leading-relaxed">
                {movie.descricao}
              </p>
            </div>

            <div className="flex gap-4">
              <Button
                className="flex items-center gap-2 px-8 py-3 bg-red-600
text-white font-bold rounded-lg hover:bg-red-700 transition-all duration-200
hover:shadow-lg hover:shadow-red-600/50 active:scale-95"
              >
                <Play size={20} />
                Assistir Agora
              </Button>
              <Button
                onClick={handleFavorite}
                className={`flex items-center gap-2 px-8 py-3 font-bold
rounded-lg transition-all duration-200 active:scale-95 ${
                  favorite
                    ? 'bg-red-600 text-white hover:bg-red-700'
                    : 'bg-neutral-800 text-white hover:bg-neutral-700'
                }`}
              >
                <Heart size={20} fill={favorite ? 'currentColor' : 'none'} />
                {favorite ? 'Remover' : 'Favoritar'}
              </Button>
            </div>
          </div>
        </div>
      </Card>
    </section>
  </div>
 );
}
```

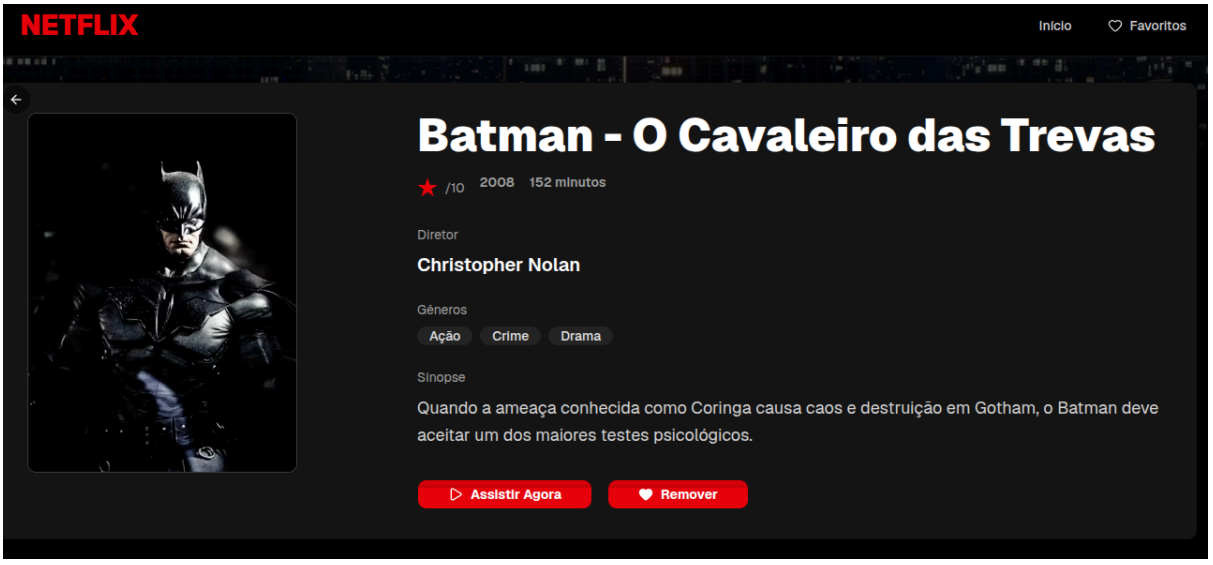
6. Adicionando as Rotas para as Páginas (Wouter)

Arquivo: “src/App.tsx”

```typescript
import './App.css'
import { Route, Switch } from "wouter";
import Home from '@/pages/Home';
import MovieDetails from '@/pages/MovieDetails';
import { FavoritesProvider } from "./contexts/FavoritesContext";

function Router() {
  return (
    <Switch>
      <Route path={"/"} component={Home} />
      <Route path={"/filme/:id"} component={MovieDetails} />
    </Switch>
  );
}
function App() {

  return (
    <FavoritesProvider>
      <Router />
    </FavoritesProvider>
  )
}

export default App
```

7. Testando a Navegação: Home => Detalhes do Filme

Acessando a aplicação no navegador
(http://localhost:12007/filme/2)



8. Alterando Componente Card - Adicionando Funcionalidade de Favoritos

- No arquivo ***"src/components/Card.tsx"***

Arquivo: "src/components/Card.tsx"

```typescript
import { Link } from 'wouter';
import { Heart } from 'lucide-react';
import { Card } from '@/components/ui/card';
import { Button } from '@/components/ui/button';
import type { Movie } from '@/types';
// Linha Adicionada
import { useFavoritesContext } from '@/contexts/FavoritesContext';

interface Props {
  movie: Movie;
}

export default function MovieCard({ movie }: Props) {

    // Linhas Adicionadas
    const { isFavorite, addFavorite, removeFavorite } = useFavoritesContext();
    const favorite = isFavorite(movie.id);

    const handleFavorite = (e: React.MouseEvent) => {
        // Linhas Acionadas
        e.preventDefault();
        if (favorite) {
            removeFavorite(movie.id);
        }
        else {
            addFavorite(movie);
        }
    }
    ... // Restante do Código (Aula passada)
}
```

9. Criando a Página de Filmes Favoritos

- No arquivo ***"src/pages/Favorites.tsx"***

Arquivo: "src/pages/Favorites.tsx"

TypeScript

```
import { Link } from 'wouter';
import { ArrowLeft } from 'lucide-react';
import Header from '@/components/Header';
import Card from '@/components/Card';
import { Button } from '@/components/ui/button';
import { useFavoritesContext } from '@/contexts/FavoritesContext';


export default function Favorites() {

    const { favorites } = useFavoritesContext();

    return (
        <div className="min-h-screen bg-black">
            <Header />

            <section className="container pt-24 pb-20">
                <div className="mb-12 ms-4">
                    <Link href="/">
                        <a>
                            <Button
                                variant="ghost"
                                className="flex items-center gap-2 text-red-600
hover:text-red-500 hover:bg-transparent transition-colors duration-200 mb-6
font-semibold"
                            >
                                <ArrowLeft size={20} />
                                Voltar
                            </Button>
                        </a>
                    </Link>
                    <h1 className="text-5xl md:text-6xl font-black text-white">
                        Meus Favoritos
                    </h1>
                    <p className="text-neutral-400 mt-2">
                        {favorites.length} - filme{favorites.length > 1 ? 's' :
''} salvo{favorites.length > 1 ? 's' : ''}
                    </p>
                </div>

                {favorites.length > 0 ? (
                    <div className="grid grid-cols-1 sm:grid-cols-2
lg:grid-cols-3 xl:grid-cols-4 gap-6">
                        {favorites.map((movie) => (
                            <Card key={movie.id} movie={movie} />
                        ))}
```

```
                    </div>
                ) : (
                    <div className="flex flex-col items-center justify-center
py-32">
                        <div className="text-center">
                        <p className="text-2xl font-bold text-neutral-400
mb-4">
                            Nenhum filme favorito ainda
                        </p>
                        <p className="text-neutral-500 mb-8">
                            Comece a adicionar filmes aos seus favoritos
                        </p>
                        <Link href="/">
                            <a>
                            <Button className="inline-flex items-center gap-2
px-6 py-3 bg-red-600 text-white font-bold rounded-lg hover:bg-red-700
transition-all duration-200">
                                Explorar Filmes
                            </Button>
                            </a>
                        </Link>
                        </div>
                    </div>
                )}
            </section>
        </div>
    );
}
```

10. Adicionando a Rota - Página de Favoritos

Arquivo: “src/App.tsx”

TypeScript

```
import './App.css'
import { Route, Switch } from "wouter";
import Home from '@/pages/Home';
import MovieDetails from '@/pages/MovieDetails';
import Favorites from '@/pages/Favorites';
import { FavoritesProvider } from "./contexts/FavoritesContext";

function Router() {
  return (
    <Switch>
```

```
      <Route path={"/"} component={Home} />
      <Route path={"/filme/:id"} component={MovieDetails} />
      <Route path={"/favoritos"} component={Favorites} />
    </Switch>
  );
}
function App() {

  return (
    <FavoritesProvider>
      <Router />
    </FavoritesProvider>
  )
}

export default App
```

11. Testando a Navegação: Home => Lista de Favoritos

Acessando a aplicação no navegador
(http://localhost:12007/)