DESENVOLVIMENTO FULL STACK

React - Props

Caio Eduardo do Prado Ireno

caio.ireno@faculdadeimpacta.com.br

Propriedades

Assim como uma função pode receber argumentos, podemos também passar argumentos aos componentes. Esses são conhecidos como propriedades ou props.

Vamos continuar com o exemplo de componentes da aulas passada

NavButton

```tsx
type NavButtonProps = {
  backgroundColor: string;
  name: string;
};

export const NavButton: React.FC<NavButtonProps> = (NavButtonProps) => {
  return (
    <button
      style={{
        backgroundColor: NavButtonProps.backgroundColor,
      }}
    >
      {NavButtonProps.name}
    </button>
  );
};
```

PropsWithChildren

```tsx
import React from "react";

type NavButtonProps = React.PropsWithChildren<{ backgroundColor: string }>;

export const NavButton: React.FC<NavButtonProps> = (NavButtonProps) => {
  return (
    <button
      style={{
        backgroundColor: NavButtonProps.backgroundColor,
      }}
    >
      {NavButtonProps.children}
    </button>
  );
};
```

Children

O conceito de children no React é uma propriedade especial que representa os elementos filhos (ou conteúdo interno) de um componente

Quando você cria um componente, pode usá-lo de duas maneiras principais:

1. Auto-fechado: **<Componente />**
2. Com elementos filhos (children):

<Componente> O conteúdo ou elementos JSX <Componente>

O conteúdo ou os elementos JSX passados entre as tags do componente são acessíveis através da prop especial chamada **children**.

Exemplo Básico

Layouts de Página

O children é extremamente útil em layouts onde você precisa de uma estrutura geral para todas as páginas do seu aplicativo.

```tsx
import Footer from "./components/Footer";
import Header from "./components/Header";

type LayoutProps = React.PropsWithChildren<{ name?: string }>;

const Layout: React.FC<LayoutProps> = ({ children }) => {
  return (
    <div>
      <Header></Header>
      <main>{children}</main>
      <Footer></Footer>
    </div>
  );
};

export const App = () => {
  return (
    <Layout>
      <h1>Página Inicial</h1>
      <p>Bem-vindo ao meu site!</p>
    </Layout>
  );
};

export default App;
```

Desestruturação

É comum desestruturarmos as propriedades na função.

```tsx
import React from "react";

type NavButtonProps = React.PropsWithChildren<{
  onClick?: () => void;
  backgroundColor: string;
  name: string;
}>;

export const NavButton: React.FC<NavButtonProps> = ({  onClick,  backgroundColor,  name,}) => {
  return (
    <button
      style={{backgroundColor: backgroundColor, }}
      onClick={onClick}
    >
      {name}
    </button>
  );
};
```

ComponentProps

Ao criarmos um componente, podemos perder as funcionalidades de auto-completar do TypeScript. O TypeScript não é capaz de prever quais elementos ou propriedades estão sendo utilizados dentro do nosso componente.

```tsx
import React from "react";

type NavButtonProps = React.ComponentProps<"button"> & {
  name: string;
  backgroundColor: string;
};

export const NavButton = ({
  name,
  backgroundColor,
  ...props
}: NavButtonProps) => {
  return (
    <button style={{ backgroundColor: backgroundColor }} {...props}>
      {name}
    </button>
  );
};
```

Obrigado!