Aula 2 – Atomic Design, Props e Estados no React

Nesta aula, vamos explorar os conceitos de atomic design, props e state no React. Estes são fundamentais para o desenvolvimento de componentes dinâmicos e reutilizáveis, e podem ser usados tanto em componentes de classe quanto em componentes funcionais.

Atomic Design é uma metodologia de desenvolvimento de interfaces proposta por Brad Frost. Ela é baseada na ideia de construir interfaces como se fossem compostas por "átomos", criando sistemas reutilizáveis e escaláveis. A metodologia organiza os componentes em cinco níveis hierárquicos:

1. **Átomos (Atoms)**: Os elementos mais básicos da interface, como botões, inputs, e ícones. Eles não têm dependências e são altamente reutilizáveis.

2. **Moléculas (Molecules)**: Combinações de átomos que formam uma unidade funcional. Exemplo: um campo de busca com um input e um botão.

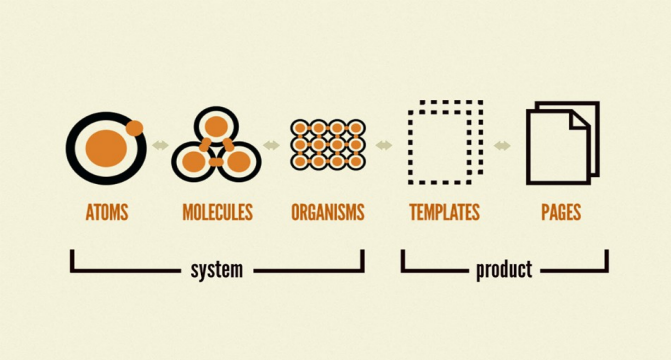
3. **Organismos (Organisms)**: Combinações de átomos e moléculas que criam blocos de interface maiores e mais complexos. Exemplo: um cabeçalho que contém logo, barra de navegação, e botão de login.

4. **Templates**: Estruturas de layout que organizam os organismos e definem como eles se comportam em uma página. Geralmente usam dados genéricos.

5. **Páginas (Pages)**: Instâncias concretas dos templates, com dados reais.

Vantagens do Atomic Design

- **Reutilização:** Facilita a reutilização de componentes em diferentes partes do projeto.
- **Consistência:** Garante que a interface tenha um visual e comportamento unificados.
- **Manutenção:** Reduz a complexidade e facilita a manutenção ao separar responsabilidades.



Ao longo do curso, veremos como o Atomic Design vai ser explorado dentro do React.

Props

Props (abreviação de 'propriedades') são usadas para passar dados de um componente pai para um componente filho. Elas são imutáveis, tornando os componentes previsíveis e reutilizáveis."). são uma forma de passar dados de um componente pai para um componente filho em React. Elas permitem que você personalize o comportamento e a aparência de um componente. As props são importantes por tais motivos: Reutilização: Props permitem que você reutilize o mesmo componente com diferentes dados.

Personalização: Você pode personalizar a saída de um componente dependendo dos dados que você passa.

Separação de Responsabilidades: Mantém a lógica de um componente separada de seus dados, facilitando a manutenção e a compreensão do código.

Na pasta da aula 2 no componente SaudacaoComProps podemos ver um exemplo de um componente utilizando as props, que no caso é usada para receber o nome do usuário.

```
// App.js
import React from "react";
import SaudacaoComProps from "./components/Aula 2/SaudacaoComProps";

Codeium: Refactor | Explain | Generate JSDoc | X
const App = () => {
  return (
    <>
      <SaudacaoComProps nome="Fabio" />
    </>
  );
};

export default App;
```

PropTypes

PropTypes são uma maneira de verificar os tipos de propriedades (props) nos componentes do React. ajudam a garantir que os componentes recebam os dados corretamente. É útil para evitar bugs e melhorar a robustez do código.

Na pasta da aula 2 no componente ExemploPropTypes podemos ver um exemplo de um componente utilizando as PropTypes.

Estados

Estado é uma maneira de armazenar dados que podem mudar ao longo do tempo em um componente React. Quando o estado de um componente é atualizado, o React re-renderiza automaticamente o componente para refletir as mudanças.

Interatividade: O estado permite que os componentes sejam dinâmicos e interativos, respondendo a ações do usuário.

Gerenciamento de Dados: O estado é útil para armazenar dados que mudam, como entradas de formulários, resultados de cálculos, etc.

Reatividade: Quando o estado muda, o React atualiza automaticamente a interface do usuário, tornando a construção de UIs dinâmicas muito mais fácil.

Na pasta da aula 2 nos componentes Contador e ContadorClasse podemos ver exemplos de uso de estados em componentes funcionais e de classe.