Revisão Básica React: useState, map, onClick e mais

1. useState — Guardar e atualizar valores no componente

- `useState` é um **hook** do React que permite criar um **estado interno** no componente.
- Estado é um dado que, quando muda, faz o componente atualizar a tela.
- Você declara assim:

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

function MeuComponente() {
  const [contador, setContador] = useState(0);
  // contador começa em 0
  // setContador é a função para mudar o contador

  function aumentar() {
    setContador(contador + 1); // atualiza o estado
  }

  return (
    <div>
      <p>Valor: {contador}</p>
      <button onClick={aumentar}>Clique para aumentar</button>
    </div>
  );
}
```

- **Explicação**:
  - `[contador, setContador]` é uma "variável + função pra mudar ela".
  - `useState(0)` define o valor inicial (0).
  - Quando chamamos `setContador`, o React redesenha o componente com o novo valor.

---

2. onClick — Reagir a cliques do usuário

- `onClick` é um evento que você coloca em botões ou outros elementos para chamar uma função quando clicar.

```
<button onClick={() => alert("Você clicou!")}>Clique aqui</button>
```

- Você pode passar uma função já definida também:

```
function handleClique() {
  console.log("clicou!");
```

```
}

<button onClick={handleClique}>Clique aqui</button>
```

- Importante: não usar parênteses ao passar a função (ex: `onClick={handleClique}`, não `onClick={handleClique()}`), para não executar imediatamente.

---

3. map — Criar listas de elementos a partir de um array

- `map` é uma função do JavaScript que percorre um array e transforma cada item em algo novo.
- No React, usamos para criar listas de componentes:

```
const frutas = ["Maçã", "Banana", "Laranja"];

function ListaFrutas() {
  return (
    <ul>
      {frutas.map((fruta, index) => (
        <li key={index}>{fruta}</li>
      ))}
    </ul>
  );
}
```

- Cada item precisa de uma `key` única para o React saber qual item é qual (pode usar `index` se não tiver id).
- `map` transforma cada fruta em um `<li>` para mostrar na lista.

---

4. Props — Passar informações para componentes filhos

- Props são os parâmetros que você passa para componentes, como se fossem atributos HTML.

```
function Saudacao({ nome }) {
  return <p>Olá, {nome}!</p>;
}

<Saudacao nome="João" />
```

- O componente `Saudacao` recebe `{ nome }` e usa para mostrar na tela.

---

5. Estado + evento para mudar conteúdo

- Exemplo simples de uma tarefa que pode ser marcada como concluída ou não:

```jsx
import { useState } from "react";

function Tarefa({ texto }) {
  const [concluida, setConcluida] = useState(false);

  function toggleConcluida() {
    setConcluida(!concluida);
  }

  return (
    <div>
      <span style={{ textDecoration: concluida ? "line-through" : "none" }}>
        {texto}
      </span>
      <button onClick={toggleConcluida}>
        {concluida ? "Desmarcar" : "Concluir"}
      </button>
    </div>
  );
}
```

- Explicação: Quando clicamos no botão, o estado `concluida` muda, e o texto recebe ou perde a linha cortada.

---

6. Exemplo completo juntando tudo (to-do list simples)

```jsx
import { useState } from "react";

const tarefasIniciais = [
  { id: 1, texto: "Estudar React" },
  { id: 2, texto: "Fazer exercícios" },
  { id: 3, texto: "Lavar a louça" },
];

function Tarefa({ texto }) {
  const [concluida, setConcluida] = useState(false);

  function toggle() {
    setConcluida(!concluida);
  }

  return (
    <div>
      <span style={{ textDecoration: concluida ? "line-through" : "none" }}>
        {texto}
      </span>
      <button onClick={toggle}>{concluida ? "Desmarcar" : "Concluir"}</button>
    </div>
  );
```

```
}

export default function ListaTarefas() {
  return (
    <div>
      <h2>Lista de Tarefas</h2>
      {tarefasIniciais.map((tarefa) => (
        <Tarefa key={tarefa.id} texto={tarefa.texto} />
      ))}
    </div>
  );
}
```

---

7. Dicas finais

- Sempre que você quiser que a tela atualize, use `useState` e a função que ele dá para mudar o valor.
- Nunca mude o estado diretamente (ex: `contador = 5`), sempre use `setContador(5)`.
- Use `map` para mostrar listas de coisas.
- Passe dados para componentes usando `props`.
- Use `onClick` para responder a cliques.
- Quando usar `map`, coloque `key` para ajudar o React a identificar cada item.
