Lista de Exercícios: React Básico (JSX, Componentes, Props, Estado, Eventos, etc.)

A seguir está uma lista de exercícios teóricos e práticos de React, organizada de forma progressiva pelos tópicos principais: **JSX, Componentes Funcionais, Props, Estado com** `useState`**, Manipulação de Eventos, Renderização Condicional, Renderização de Listas, Comunicação entre Componentes, Formulários Controlados e Introdução ao** `useEffect`. Os exercícios variam em formato (múltipla escolha, completar código, dissertativo e implementação de código).

JSX

Exercício 1 — Múltipla escolha
Qual das seguintes afirmativas sobre JSX está incorreta?

A) O JSX permite escrever código parecido com HTML dentro do JavaScript, facilitando a criação de componentes de interface.

B) Para aplicar classes CSS em elementos JSX, utiliza-se o atributo **class** como no HTML, por exemplo: `<div class="minha-classe">`.

C) O JSX precisa ser transformado em JavaScript puro (por exemplo, usando Babel) antes de ser executado pelo navegador.

D) Podemos incorporar expressões JavaScript em um JSX colocando-as entre chaves `{ }`, por exemplo: `<p>{1 + 1}</p>`.

Exercício 2 — Identifique e corrija o erro
O código JSX abaixo contém um erro. Identifique o erro e explique como corrigi-lo.

```
function Saudacao() {
  return (
    <div class="saudacao">
      <h1>Olá, mundo!</h1>
    </div>
  );
}
```

Dica: Atenção à forma correta de aplicar classes CSS em JSX e à sintaxe geral do JSX.

Componentes Funcionais

Exercício 3 — Implementação
Crie um componente funcional React chamado **BoasVindas** que retorna um elemento `<h2>` com o texto **"Bem-vindo!"**. Escreva o código desse componente.

Exercício 4 — Dissertativo
Em React, os componentes devem ter nomes começando com letra maiúscula. Explique por que existe essa regra de nomenclatura. O que acontece se você definir um componente com letra inicial minúscula?

Props (Propriedades)

Exercício 5 — Completar código
Considere o componente funcional **Saudacao** abaixo, que espera receber uma prop **nome** e exibe uma saudação. Complete o código de uso do componente **Saudacao** para passar a prop **nome** com o valor **"João"** corretamente.

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

function App() {
  return (
    <div>
      {/* Complete abaixo para passar a prop "nome" com valor "João" */}
      <Saudacao _____________ />
    </div>
  );
}
```

Exercício 6 — Dissertativo
É correto um componente React modificar o valor de suas próprias props? Por exemplo, se um componente recebe uma prop **titulo**, ele poderia alterar esse valor internamente? Justifique sua resposta explicando a natureza das props e como deve ser feita a passagem de dados correta entre componentes.

Estado com `useState`

Exercício 7 — Múltipla escolha
Qual das alternativas a seguir declara corretamente um estado chamado **contador** com valor inicial **0** usando o Hook `useState`?

A) `const contador = useState(0);`

B) `const [contador, setContador] = useState(0);`

C) `let contador = 0; useState(contador);`

D) `const [contador, setContador] = useState();`

Exercício 8 — Implementação
Implemente um componente funcional **Contador** que: - Exiba um título ou parágrafo mostrando o valor atual do contador. - Comece com o valor **0** utilizando um estado (`useState`). - Tenha um botão com o texto **"Incrementar"** que, ao ser clicado, acrescente **+1** ao valor do contador (atualizando o estado).

Escreva o código JSX/React completo do componente **Contador** implementando a funcionalidade descrita.

Manipulação de Eventos

Exercício 9 — Múltipla escolha
Suponha que você tenha uma função chamada **handleClick** definida em um componente React. Qual é a maneira correta de associar essa função ao evento de clique (`onClick`) de um botão?

A) `<button onclick="handleClick()">Clique aqui</button>`

B) `<button onClick={handleClick()}>Clique aqui</button>`

C) `<button onClick={handleClick}>Clique aqui</button>`

D) `<button onClick="handleClick">Clique aqui</button>`

Exercício 10 — Implementação
Crie um componente funcional **BotaoAlerta** que renderize um `<button>` com o texto **"Mostrar Alerta"**. Quando esse botão for clicado, deve disparar um alerta (`alert`) no navegador com a mensagem **"Botão clicado!"**. Implemente o componente mostrando o JSX e a função de evento necessária.

Renderização Condicional

Exercício 11 — Implementação curta
Escreva um trecho de JSX que exiba um parágrafo `<p>Bem-vindo de volta!</p>` somente se uma variável booleana **logado** for verdadeira (`true`). Se **logado** for falso, nada deve ser renderizado por esse trecho.

Exercício 12 — Implementação
Crie um componente funcional **SaudacaoUsuario** que recebe uma prop **nome**. O componente deve renderizar: - `<h2>Olá, {nome}!</h2>` caso a prop **nome** esteja definida e **não** seja uma string vazia. - `<h2>Por favor, faça login.</h2>` caso a prop **nome** esteja ausente, seja `undefined` ou seja uma string vazia.

Mostre também um exemplo de uso do componente **SaudacaoUsuario** chamando-o uma vez com a prop **nome** definida e outra vez **sem** passar a prop, para demonstrar as duas situações de renderização condicional.

Renderização de Listas

Exercício 13 — Múltipla escolha
Por que é importante fornecer a prop especial **key** ao renderizar uma lista de elementos em React?

A) Porque o React usa as keys para identificar e acompanhar cada elemento da lista, otimizando as atualizações e evitando problemas ao reordenar ou modificar itens.

B) Porque sem uma key, os elementos da lista não serão renderizados no DOM.

C) Porque a prop key define automaticamente a posição de cada item na lista na ordem de renderização.

D) Porque é através da key que podemos aplicar estilos CSS únicos a cada item da lista.

Exercício 14 — Implementação
Escreva um componente funcional **ListaFrutas** que recebe via props um array de strings chamado **frutas**. O componente deve renderizar uma lista não ordenada (`<ul>`) e, dentro dela, criar um item `<li>` para cada fruta no array, exibindo o nome da fruta. Lembre-se de utilizar uma **key** única para cada item ao usar a iteração (por exemplo, com `.map`).

Forneça um exemplo de uso do componente **ListaFrutas** passando um array de exemplo (por exemplo, `['Maçã', 'Banana', 'Laranja']`).

Comunicação entre Componentes

Exercício 15 — Dissertativo com exemplo
Como você pode passar dados de um componente **pai** para um componente **filho** em React? Descreva brevemente o mecanismo e forneça um exemplo simples (por exemplo, passando um texto de saudação de um componente Pai para um componente Filho via props).

Exercício 16 — Dissertativo/Implementação
Suponha que você tenha dois componentes, **Pai** e **Filho**. O componente Pai mantém um **estado** (por exemplo, um contador ou uma mensagem) e deseja ser **notificado** quando algo acontecer no Filho (por exemplo, um clique de botão) para então atualizar seu próprio estado. Descreva ou demonstre como implementar essa comunicação do **Filho → Pai** usando props: isto é, como o Pai pode **passar uma função** para o Filho via props e o Filho **chamar essa função** para enviar informações ou notificações de volta para o Pai.

Formulários Controlados

Exercício 17 — Múltipla escolha
O que melhor descreve um **formulário controlado** em React?

A) Um formulário cujo estado dos campos é mantido pelo DOM do navegador, em vez de variáveis de estado do React.

B) Um formulário em que cada campo de entrada (input) tem seu valor **controlado pelo estado** do componente React, geralmente atualizado a cada evento `onChange`.

C) Um formulário que utiliza **refs** para ler os valores dos campos de entrada somente no envio, ao invés de armazená-los no estado.

D) Um formulário em que os inputs não possuem o atributo `value`, permitindo que o usuário digite livremente sem qualquer intervenção do React.

Exercício 18 — Implementação
Crie um componente funcional **Espelhador** que contém um campo de texto (`<input type="text">`) e um parágrafo `<p>`. O valor digitado no campo de texto deve aparecer **simultaneamente** (em tempo real) no parágrafo abaixo dele. Utilize estado (`useState`) para armazenar o texto do input e faça a ligação adequada entre o **valor** do input e o **estado** (torne o input um campo controlado com `onChange`).

Introdução ao `useEffect`

Exercício 19 — Múltipla escolha
Qual é a finalidade principal do Hook `useEffect` no React?

A) Permitir manipular diretamente o DOM de um elemento JSX, substituindo a necessidade de usar seletores como `document.getElementById`.

B) Executar **efeitos colaterais** em componentes funcionais, como buscar dados em uma API, assinar eventos do navegador ou atualizar algo fora do React após a renderização do componente.

C) Declarar um estado derivado que recalcula automaticamente com base em outros estados, evitando a necessidade de usar vários Hooks `useState`.

D) Garantir que o componente funcional seja re-renderizado automaticamente sempre que suas props mudarem.

Exercício 20 — Implementação
Suponha um componente **ContadorComTitulo** com um estado de contador e um botão para incrementá-lo. Utilize o Hook `useEffect` para **atualizar o título da página** do navegador sempre que o valor do contador for atualizado. Por exemplo, quando o valor do contador for **5**, o título (`document.title`) deve ser **"Contador: 5"**. Escreva o trecho de código dentro do componente que implementa esse comportamento.