


State e Aperfeiçoamento

Fonte: Documentação oficial [React](#) e [Next JS](#)

Adicionando interatividade com o estado

Vamos explorar como o React nos ajuda a adicionar interatividade com **state** and **event handlers**.

Como exemplo, vamos criar um botão de curtir dentro do seu componente `HomePage`. Primeiro, adicione um elemento de botão dentro da instrução `return()`:

```
function HomePage() {  
  
  const names = ['Ada Lovelace', 'Grace Hopper', 'Margaret Hamilton'];  
  
  return (  
    <div>  
      <Header title="Develop. Preview. Ship. 🚀" />  
      <ul>  
        {names.map((name) => (  
          <li key={name}>{name}</li>  
        ))}  
      </ul>  
  
      <button>Like</button>   
    </div>  
  );  
}
```

Ouvindo eventos


Para fazer o botão fazer algo quando clicado, você pode usar o evento `onClick`.

```
<button onClick={}>Like</button>
```

No React, os nomes dos eventos são camelCased. O evento *onClick* é um dos muitos eventos possíveis que você pode usar para responder à interação do usuário. Por exemplo, você pode usar *onChange* para campos de entrada ou *onSubmit* para formulários.

Manipulando Eventos

Você pode definir uma função para "manipular" eventos sempre que eles forem acionados. Crie uma função antes da declaração de retorno chamada handleClick():

```
function HomePage() {  
  const names = ['Ada Lovelace', 'Grace Hopper', 'Margaret Hamilton'];  
    
  function handleClick() {  
    console.log("increment like count")  
  }  
  return (  
    <div>  
      <Header title="Developer Preview Shin" />  
    )  
  }  
}
```

```
function handleClick() {  
    console.log("increment like count")  
}
```

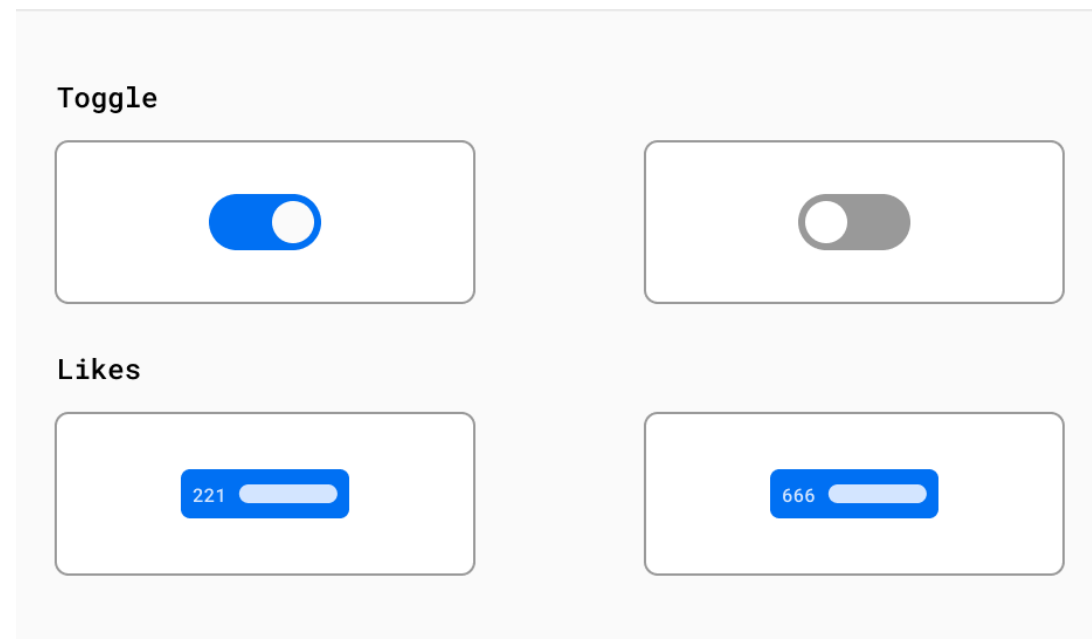
Em seguida, você pode chamar a handleClickfunção quando o onClickevento for acionado:

```
    ))}  
  </ul>  
  
  <button onClick={handleClick}>Like</button>  
  </div>  
);  
}
```


`<button onClick={handleClick}>Like</button>`

State e Hooks


O React tem um conjunto de funções chamadas hooks . Eles permitem adicionar mais recursos, como state, aos seus componentes. Você pode pensar em state como qualquer informação em sua interface que muda ao longo do tempo, geralmente acionada pela interação do usuário.



Você pode usar o estado para armazenar e incrementar o número de vezes que um usuário clicou no botão Curtir. Para utilizar o hook que gerencia o state utilize o comando: `useState()`

```
function HomePage() {  
   React.useState();  
  const names = ['Ada Love
```

O `useState()` retorna um array e você pode acessar e usar esses valores do array dentro do seu componente usando a desestruturação do array:

```
function HomePage() {  
   const [] = React.useState();  
  const names = ['Ada Lovelace', 'C
```


O primeiro item no array é o state value, que você pode nomear qualquer coisa. É recomendável nomeá-lo como algo descritivo.

O segundo item no array é uma função para atualizar o valor. Você pode nomear qualquer função de atualização, mas é comum usar como prefixo “set” seguido do nome da variável de estado que você está atualizando:

```
function HomePage() {
```

```
  1ºconst [likes, 2ºsetLikes] = React.useState();
```

```
  const names = ['Ada Lovelace', 'Grace Hopper', 'Mar
```

```
  const [likes, setLikes] = React.useState();
```

Você também pode aproveitar para inserir o valor inicial do seu state likes: zero.



```
const [likes, setLikes] = React.useState(0);
```

Então, você pode verificar se o estado inicial está funcionando usando a variável de estado dentro do seu componente.

```
<button onClick={handleClick}>Like({likes})</button>
```

A red bracket is drawn under the `{likes}` prop in the JSX code, highlighting the state variable being passed to the component.

Por fim, você pode chamar sua função de atualização de state, `setLikes` em seu componente `HomePage`, vamos adicioná-la dentro da função `handleClick()` que você definiu anteriormente:

```
function HomePage() {  
  const [likes, setLikes] = React.useState(0);  
  
  const names = ['Ada Lovelace', 'Grace Hopper', 'Margaret Hamilton'];  
  
  function handleClick() {  
    → setLikes(likes + 1);  
  }  
  
  return (  

```

```
    setLikes(likes + 1);  
  );  
}
```

Clicar no botão agora chamará a função handleClick, que chama a função setLikes de atualização de state com um único argumento do número atual de curtidas + 1.

Nota: Ao contrário das props que são passadas aos componentes como o primeiro parâmetro de função, o state é iniciado e armazenado dentro de um componente. Você pode passar as informações de estado para componentes filhos como props, mas a lógica para atualizar o estado deve ser mantida dentro do componente onde o estado foi inicialmente criado.

Gerenciando Estado

Esta foi apenas uma introdução do state, e você pode aprender mais sobre como gerenciar o estado e o fluxo de dados em seus aplicativos React. Para saber mais, recomendamos que você consulte as seções [Adicionando interatividade](#) e [Gerenciando state](#) na documentação do React.

Qual é a diferença entre props e state?

- a) Props são informações somente leitura passadas para os componentes. O estado é uma informação que pode mudar ao longo do tempo, geralmente acionada pela interação do usuário.
- b) Props são usados para definir o comportamento do componente. State é usado para estilizar componentes.
- c) Props contêm informações temporárias sobre um componente. O estado contém informações somente leitura.

Qual é a diferença entre props e state?

- a) Props são informações somente leitura passadas para os componentes. O estado é uma informação que pode mudar ao longo do tempo, geralmente acionada pela interação do usuário.
- b) Props são usados para definir o comportamento do componente. State é usado para estilizar componentes.
- c) Props contêm informações temporárias sobre um componente. O estado contém informações somente leitura.

Aprenda mais em:

[State: a memória de um componente](#)

[Conheça seu primeiro hook](#)

[Respondendo a Eventos](#)

Como continuar aprendendo React

Você acabou de conhecer três conceitos essenciais do React: components , props e state . Ter uma base sólida nisso ajudará você a começar a criar aplicativos React. Assim que se sentir mais confiante, confira também estes outros tópicos do React:

- [Como o React lida com renderizações](#) e [como usar refs](#)
- [Como gerenciar o estado](#)
- [Como usar o contexto para dados profundamente aninhados](#)
- [Como usar ganchos da API do React](#), como `useEffect()`

Recursos do React

Ao longo dos anos, muitos cursos, vídeos e artigos foram criados para ajudar os desenvolvedores a aprender React. Embora seja difícil recomendar recursos adequados ao seu estilo de aprendizado, uma referência inestimável é a documentação do React , que contém sandboxes interativos para ajudá-lo a praticar os tópicos.

Quando se trata de aprender React, a melhor maneira de aprender é construir aplicativos . Você pode adotar o React gradualmente usando `<script>` e o que aprendeu até agora para adicionar pequenos componentes a um site existente. No entanto, muitos desenvolvedores descobriram que a experiência do usuário e do desenvolvedor que o React permite é valiosa o suficiente para mergulhar de cabeça e escrever todo o projeto de front-end no React.

De React para Next.js

Embora o React seja excelente na criação de interface do usuário, é necessário algum trabalho para criar essa interface de forma independente em um aplicativo escalável totalmente funcional. A boa notícia é que o Next.js lida com grande parte da instalação e configuração e possui recursos adicionais para ajudá-lo a criar aplicativos React.

[Aqui](#), migraremos o exemplo do React para o Next.js, discutiremos como o Next.js funciona e apresentaremos alguns conceitos de desenvolvimento da Web para ajudá-lo a aprender os recursos mais avançados do Next.js.