


Ngân Tuyền

Challenge 1 of 2: Fix an event handler

Tại hàng 12 để tham chiếu đến hàm handleClick không được kèm theo ()

```
App.js
Download Reset Fork

2  function handleClick() {
3    let bodyStyle = document.body.style;
4    if (bodyStyle.backgroundColor === 'black') {
5      bodyStyle.backgroundColor = 'white';
6    } else {
7      bodyStyle.backgroundColor = 'black';
8    }
9  }
10
11 return (
12   <button onClick={handleClick}>
13     Toggle the lights
14   </button>
15 );
16 }
17
```

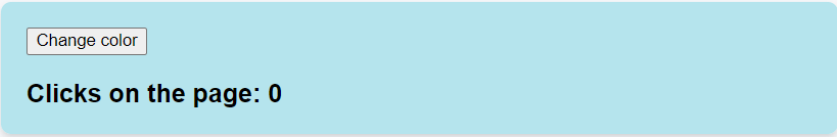


Challenge 2 of 2: Wire up the events

Dùng stopPropagation để ngăn chặn sự lan truyền sự kiện từ phần tử cha sang phần tử con, từ đó giúp thay đổi màu background mà không tăng giá trị clicks

```
ColorSwitch.js
Reset Fork

1  export default function ColorSwitch({
2    onChangeColor
3  }) {
4    return (
5      <button onClick={e => {
6        e.stopPropagation();
7        onChangeColor();
8      }}>
9        Change color
10     </button>
11   );
12 }
```

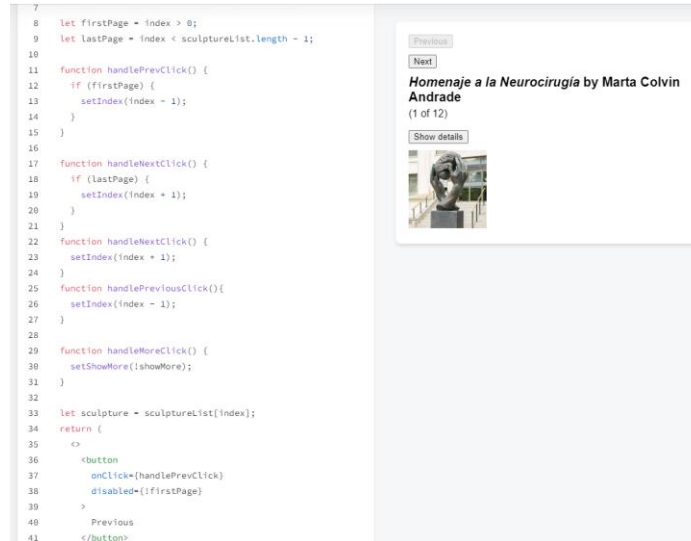


Challenge 1 of 4: Complete the gallery

Đầu tiên set điều kiện để hiện button Next và Previous.

Điều khiển trạng thái button bằng disabled, nếu không là firstPage thì hiện button previous và ngược lại với lastPage thì không hiện Next.

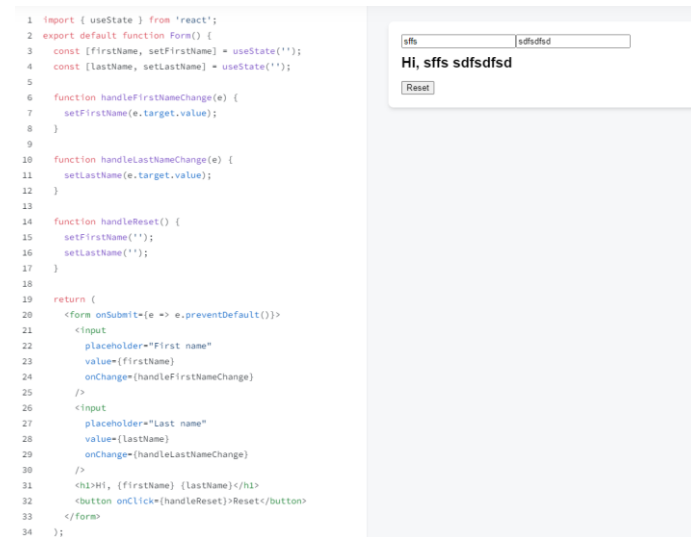
Chuyển trang bằng cách thêm điều kiện firstPage, lastPage vào các function handle.



Challenge 4 of 4: Fix stuck form inputs

Khai báo firstName, lastName bằng useState.

Hàm handleFirstNameChange, handleLastNameChange xử lý sự kiện để cập nhật giá trị khi người dùng thay đổi thông tin trong thẻ input, các giá trị này được lấy từ e.target.value.



Challenge 3 of 4: Fix a crash

Lấy `const [message, setMessage] = useState('');` ra khỏi vòng lặp `if else` để đảm bảo rằng giá trị trạng thái `message` và hàm `setMessage` sẽ được duy trì qua các lần render. Nếu để hàm này trong vòng lặp thì mỗi lần render sẽ làm mất các giá trị trên.

App.js Download Reset Fork

```
1 import { useState } from 'react';
2
3 export default function FeedbackForm() {
4   const [isSent, setIsSent] = useState(false);
5   const [message, setMessage] = useState('');
6   if (isSent) {
7     return <h1>Thank you!</h1>;
8   } else {
9     // eslint-disable-next-line
10
11     return (
12       <form onSubmit={e => {
13         e.preventDefault();
14         alert('Sending: '${message}');
15         setIsSent(true);
16       }}>
17         <textarea
18           placeholder="Message"
19           value={message}
20           onChange={e => setMessage(e.target.value)}
21         />
22         <br />
23         <button type="submit">Send</button>
24       </form>
25     );
26   }
27 }
28
```

Thank you!

Challenge 4 of 4: Remove unnecessary state

Tạo biến `name = prompt('What is your name?')` để lấy giá trị từ hộp thoại và `alert('Hello, ${name}!')`; sẽ lấy giá trị vừa nhập được để hiển thị.

App.js Download Reset Fork

```
1 import { useState } from 'react';
2
3 export default function FeedbackForm() {
4   const [name, setName] = useState('');
5
6   function handleClick() {
7     const name = prompt('What is your name?');
8     alert('Hello, ${name}!');
9   }
10
11   return (
12     <button onClick={handleClick}>
13       Greet
14     </button>
15   );
16 }
17
18
```

Greet

Show less

Challenge 1 of 3: Fix incorrect state updates

Xử lý `handleLastNameChange` và `handlePlusClick` tương tự như `handleFirstNameChange` bằng cách thêm `...player` vào `setPlayer` để cập nhật lại các thuộc tính đã thay đổi.

```
10 function handlePlusClick(e) {
11   setPlayer({
12     ...player,
13     score: player.score + 1,
14   });
15 }
16 function handleFirstNameChange(e) {
17   setPlayer({
18     ...player,
19     firstName: e.target.value,
20   });
21 }
22 function handleLastNameChange(e) {
23   setPlayer({
24     ...player,
25     lastName: e.target.value
26   });
27 }
28 return (
29   <>
30     <label>
31       Score: <b>{player.score}</b>
32       { ' ' }
33       <button onClick={handlePlusClick}>
34         +1
35       </button>
36     </label>
37     <label>
38       First name:
39       <input
40         value={player.firstName}
41         onChange={handleFirstNameChange}
42       />
43     </label>
```

Score: 11

First name:

Last name:

Challenge 2 of 3: Find and fix the mutation

Dùng `setShape` để cập nhật lại vị trí của box khi người dùng di chuyển, gán `x: shape.position.x += dx` và `y: shape.position.y += dy` nhằm chỉ để thay đổi vị trí của box mà không phụ thuộc vào `initialPosition` nhằm vẫn giữ nguyên vị trí của background.

```
10 export default function Canvas() {
11   const [shape, setShape] = useState({
12     color: 'orange',
13     position: initialPosition
14   });
15
16   function handleMove(dx, dy) {
17     setShape([
18       ...shape,
19       position: {
20         x: shape.position.x + dx,
21         y: shape.position.y + dy,
22       }
23     ]);
24   }
25
26   function handleColorChange(e) {
27     setShape([
28       ...shape,
29       color: e.target.value
30     ]);
31   }
32
33   return (
34     <>
35       <select
36         value={shape.color}
37         onChange={handleColorChange}
38       >
39         <option value="orange">orange</option>
40         <option value="lightpink">lightpink</option>
41       </select>
42     </>
```

aliceblue

Drag me!

Challenge 3 of 3: Update an object with Immer

Tương tự bài trên dùng useImmer để khởi tạo color và positin cho box, handleMove sử dụng draft để cập nhật vị trí box khi người dùng di chuyển.

```
10
11 export default function Canvas() {
12   const [shape, updateShape] = useImmer({
13     color: 'orange',
14     position: initialPosition
15   });
16
17
18   function handleMove(dx, dy) {
19     updateShape(draft => {
20       draft.position.x += dx;
21       draft.position.y += dy;
22     });
23   }
24
25   function handleColorChange(e) {
26     updateShape(draft => {
27       draft.color = e.target.value;
28     });
29   }
30 }
```

