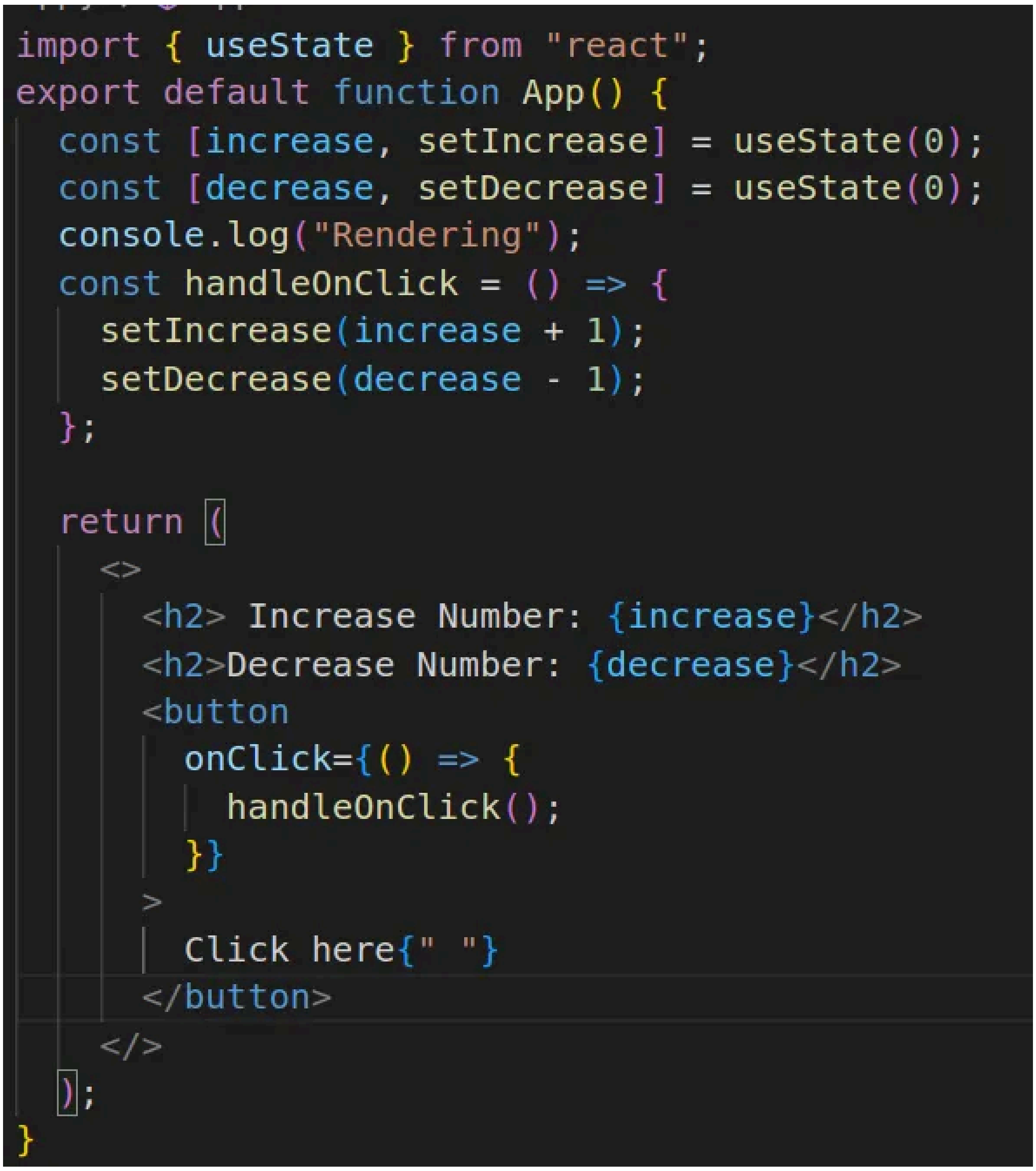
Automatic Batching in React 18: How it works

Rashmi Bharti · Follow
3 min read · Nov 24, 2022

React uses **Batches** to **groups multiple state updates into a single re-render** for better performance.

Batching in React 17

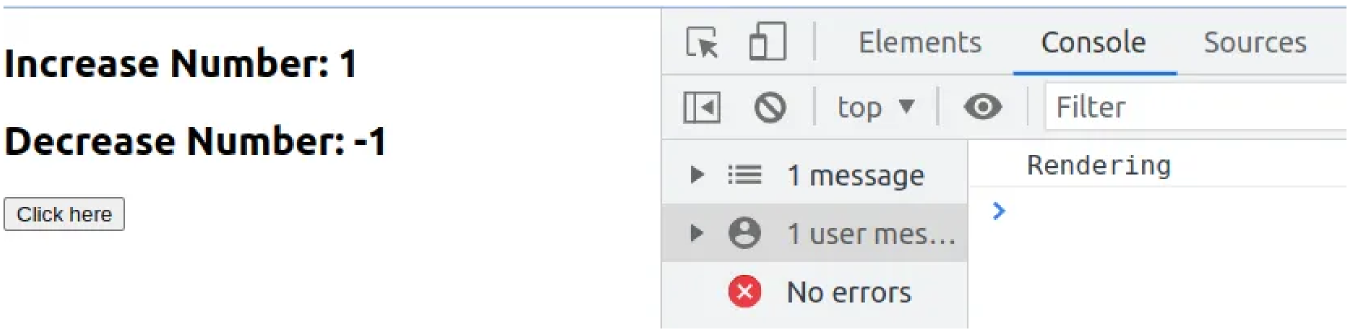
In React 17 and prior versions, let us understand how batches works.Let's consider the following example of making a state update in a React component:



```
import { useState } from "react";
export default function App() {
  const [increase, setIncrease] = useState(0);
  const [decrease, setDecrease] = useState(0);
  console.log("Rendering");
  const handleOnClick = () => {
    setIncrease(increase + 1);
    setDecrease(decrease - 1);
  };

  return (
    <>
      <h2> Increase Number: {increase}</h2>
      <h2>Decrease Number: {decrease}</h2>
      <button
        onClick={() => {
          handleOnClick();
        }}
      >
        Click here{" "}
      </button>
    </>
  );
}
```

In the above example, the `handleOnClick()` function will be called when a user clicks on the button. It will execute two-state updates on each click.

If you observe the browser console, you will see that the **"Rendering"** message is logged only once for both state updates.



Increase Number: 1

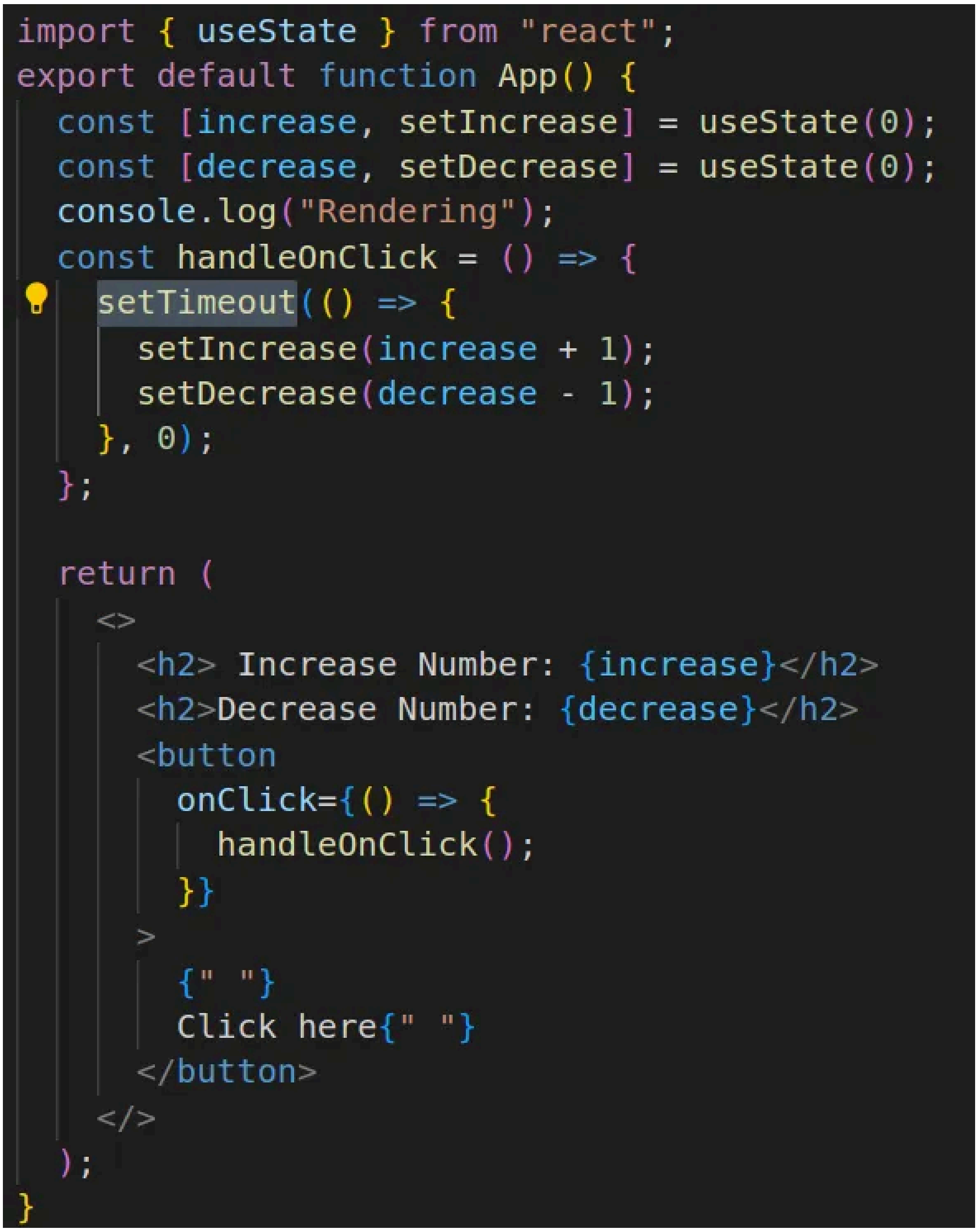
Decrease Number: -1

```
Rendering
```

Now you have seen that React batched both state updates and re-rendered the component only once.

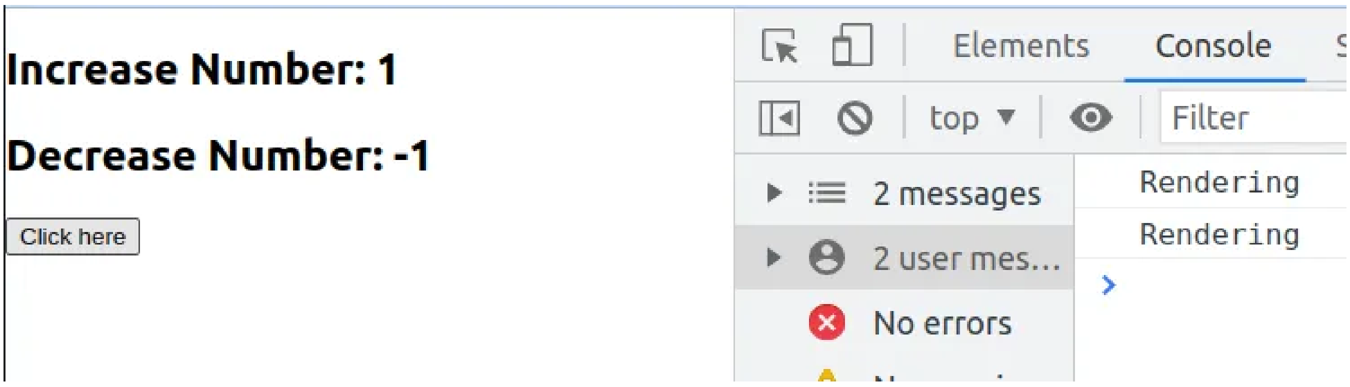
But, what if we execute state updates in a context that is not associated with the browser?

For example, consider a **setTimeout()** call that asynchronously loads data:



```
import { useState } from "react";
export default function App() {
  const [increase, setIncrease] = useState(0);
  const [decrease, setDecrease] = useState(0);
  console.log("Rendering");
  const handleOnClick = () => {
    setTimeout(() => {
      setIncrease(increase + 1);
      setDecrease(decrease - 1);
    }, 0);
  };

  return (
    <>
      <h2> Increase Number: {increase}</h2>
      <h2>Decrease Number: {decrease}</h2>
      <button
        onClick={() => {
          handleOnClick();
        }}
      >
        {" "}
        Click here{" "}
      </button>
    </>
  );
}
```

In the above example, the state update occurs in the callback of **setTimeout()** function.


Increase Number: 1

Decrease Number: -1

```
Rendering
Rendering
```

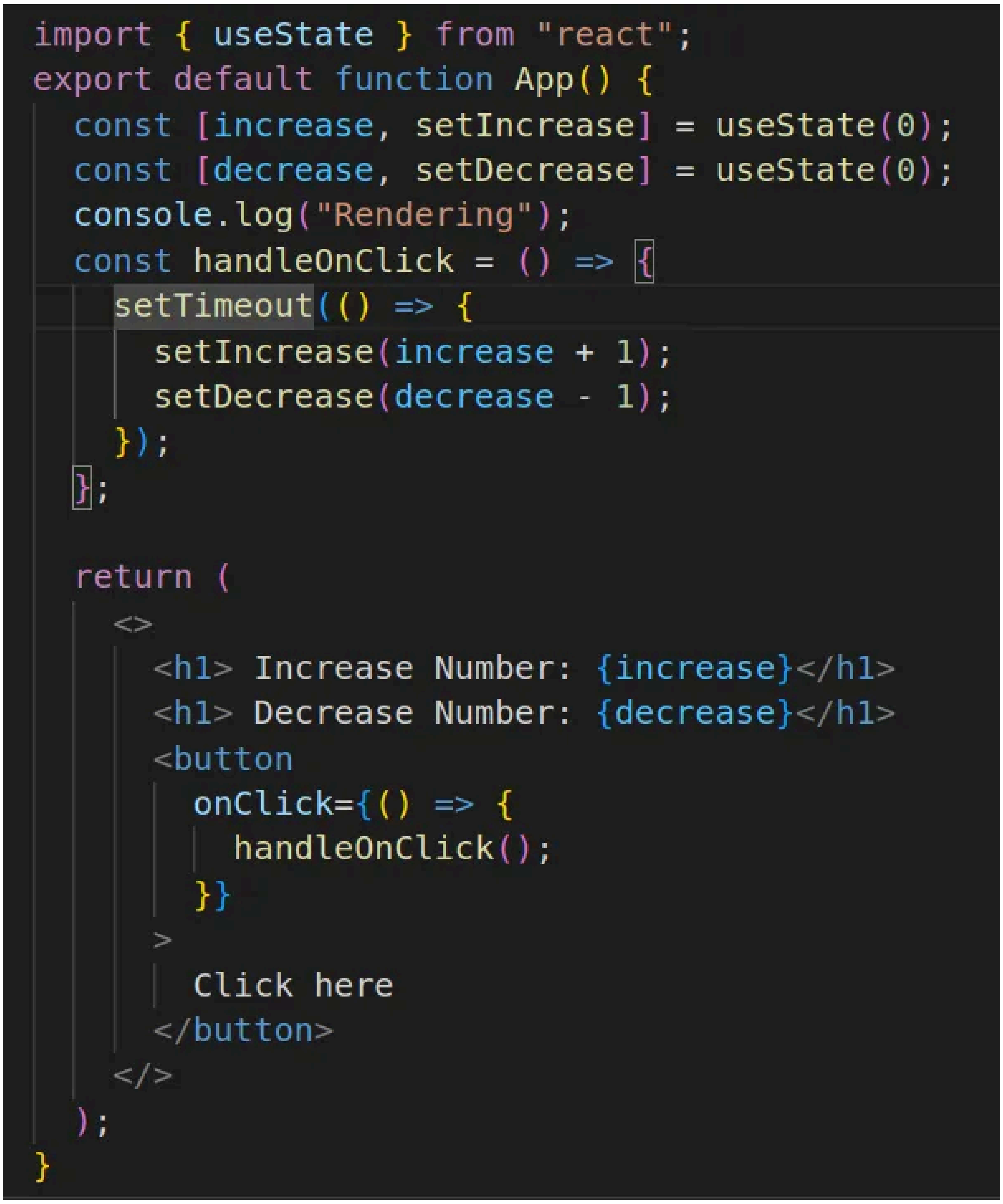
If you observe the browser console after executing this example, you will see 2 messages. This indicates that two separate re-renders occur for each state update.

This behaviour can cause severe performance issues related to application speed. That's why React introduced **Automatic Batching**.

Using Automatic Batching:

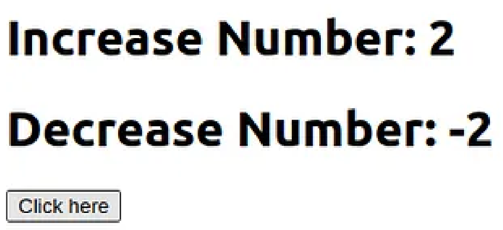
In React 18 ensures that state updates invoked from any location will be batched by default. This will batch state updates, including native event handlers, asynchronous operations, timeouts, and intervals.

Let's consider the following example to understand how **automatic batching works.**



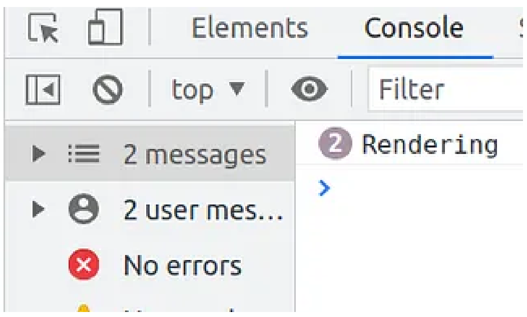
```
import { useState } from "react";
export default function App() {
  const [increase, setIncrease] = useState(0);
  const [decrease, setDecrease] = useState(0);
  console.log("Rendering");
  const handleOnClick = () => {
    setTimeout(() => {
      setIncrease(increase + 1);
      setDecrease(decrease - 1);
    });
  };

  return (
    <>
      <h1> Increase Number: {increase}</h1>
      <h1> Decrease Number: {decrease}</h1>
      <button
        onClick={() => {
          handleOnClick();
        }}
      >
        Click here
      </button>
    </>
  );
}
```

Above example result is:-



Increase Number: 2

Decrease Number: -2



```
2 Rendering
```

The above example shows an optimal re-rendering process where `handleOnClick()` function will be called when a user clicks on the button.It will execute two-state updates on each click and give only one re-render as React batches all state updates regardless of the invocation location.

How to stop Automatic Batching

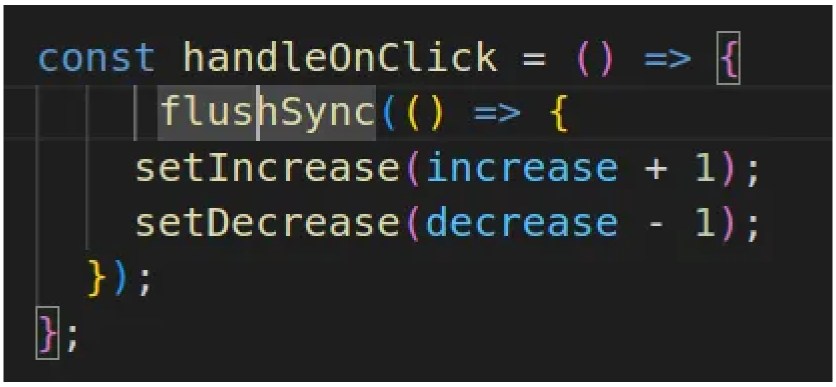
Automatic batching is indeed an amazing feature. But, there can be situations where we need to prevent this from happening. For that, React provides a method named `flushSync()` in `react-dom` that allows us to trigger a re-render for a specific state update.

How does flushSync() works

To use this, simply import it from `react-dom` using the syntax:

```
import { flushSync } from  'react-dom';
```

And then, call the method inside an event handler and place your state update inside the body of `flushSync()` .



```
const handleOnClick = () => {
    flushSync(() => {
    setIncrease(increase + 1);
    setDecrease(decrease - 1);
  });
};
```

When the event is invoked, React will update the DOM once at `flushSync()` and update the DOM again at `setIncrease(increase + 1)` avoiding the batching.

Conclusion

React 18 will batch the state updates for us no matter if it is in a simple function containing multiple state updates or a web API and interfaces like setTimeout, fetch or a promise containing multiple state updates.

Feel free to share any thoughts or opinions or questions, I will be happy to help you as I can, Thanks for reading...!

If you are interested in software development, you can connect with mobcoder (https://mobcoder.com/contact)

#mobcoder #softwareDevelopment #reactjs #automaticBatching #javascript #batching

R

Follow

Written by Rashmi Bharti

3 Followers