Things I learnt from the New React Docs

Debbie O’Brien
@debs_obrien

React Conf

Debbie O’Brien

Head Developer Advocate at Bit

Member of the React 18 Working Group

@debs_obrien

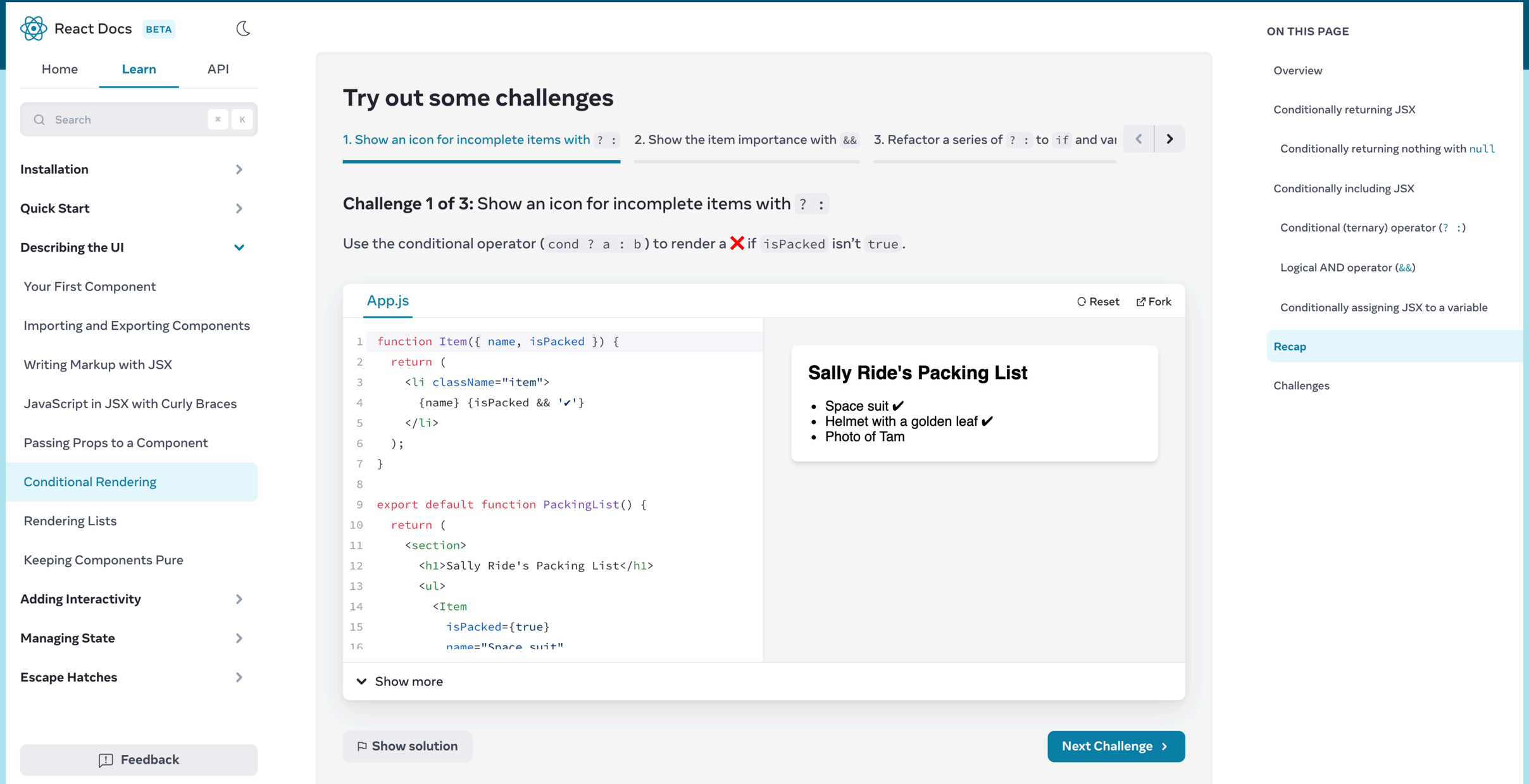
debbie.codes

```
};

export const ThemeContext = React.createContext(
  themes.dark // default value
);
```

themed-button.js

```
import {ThemeContext} from './theme-context';

class ThemedButton extends React.Component {
  render() {
    let props = this.props;
    let theme = this.context;
    return (
      <button
        {...props}
        style={{backgroundColor: theme.background}}
      />
    );
  }
}
ThemedButton.contextType = ThemeContext;

export default ThemedButton;
```


The new React Docs



React Component

A FUNCTION WITH A CAPITAL LETTER

Step 2: Define the function

With `function Profile() { }` you define a JavaScript function with the name `Profile`.

Gotcha

React components are regular JavaScript functions, but **their names must start with a capital letter** or they won't work!

bit.ly/react-first-component

JSX

WITH CURLY BRACES YOU ARE ENTERING INTO JAVASCRIPT LAND

```
<ul style={
  {
    backgroundColor: 'black',
    color: 'pink'
  }
}>
```

The next time you see `{{` and `}}` in JSX, know that it's nothing more than an object inside the JSX curlies!

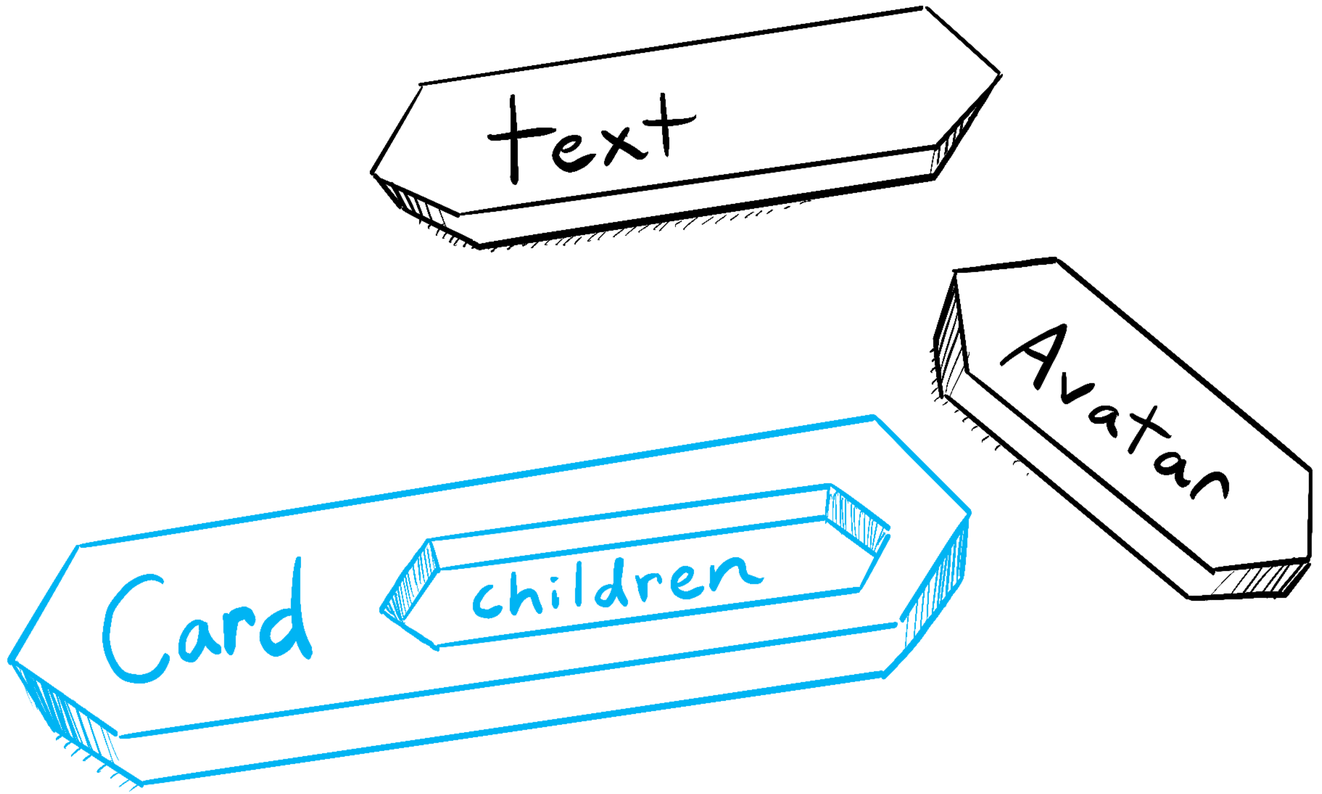
bit.ly/jsx-curly-braces

Children

A prop that can be filled in by its parent

Children

{children} A HOLE THAT CAN BE FILLED IN BY IT'S PARENT



bit.ly/jsx-as-children

App.js Avatar.js utils.js

Reset Fork

```
import Avatar from './Avatar.js';

function Card({ children }) {
  return (
    <div className="card">
      {children}
    </div>
  );
}

export default function Profile() {
  return (
    <Card>
      <Avatar
        size={100}
        person={{
          name: 'Katsuko Saruhashi',
          imageId: 'YfeOqp2'
        }}
      />
    </Card>
  );
}

```

Show less

Context

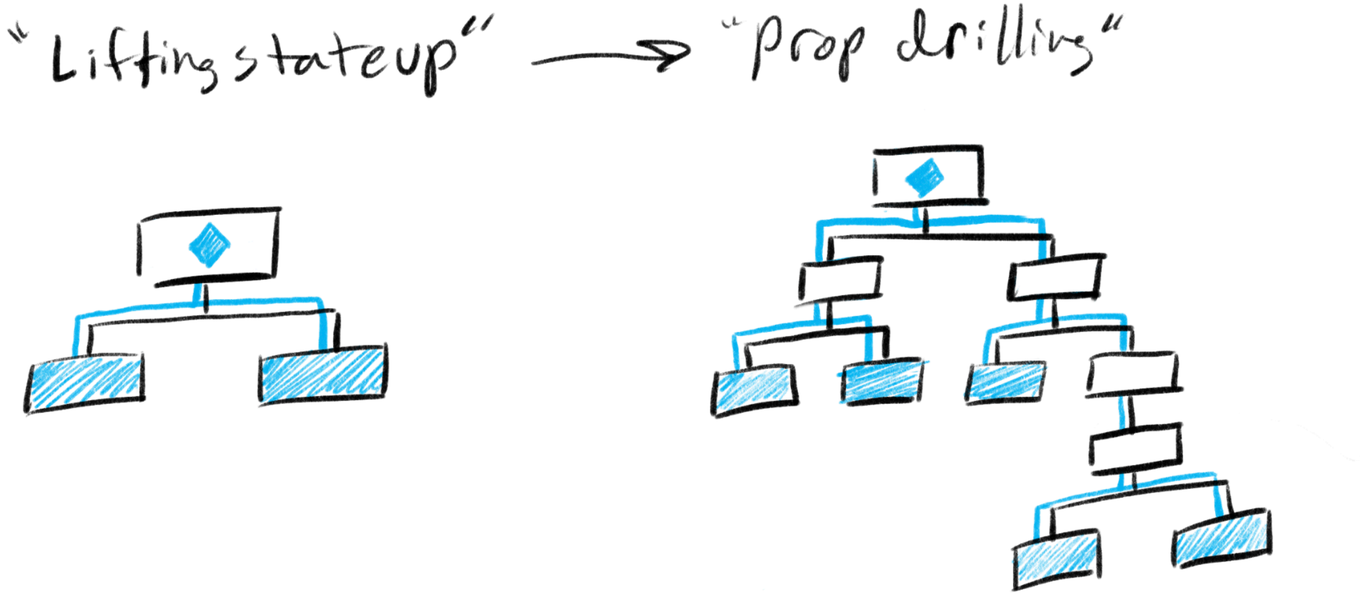
An alternative to passing props

Context

LETS A PARENT COMPONENT PROVIDE DATA TO THE ENTIRE TREE BELOW IT

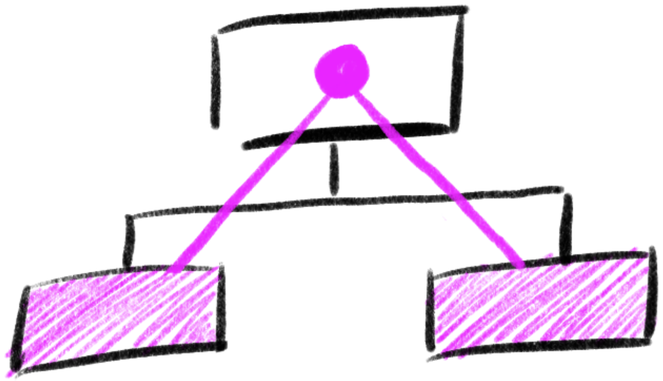
The problem with passing props

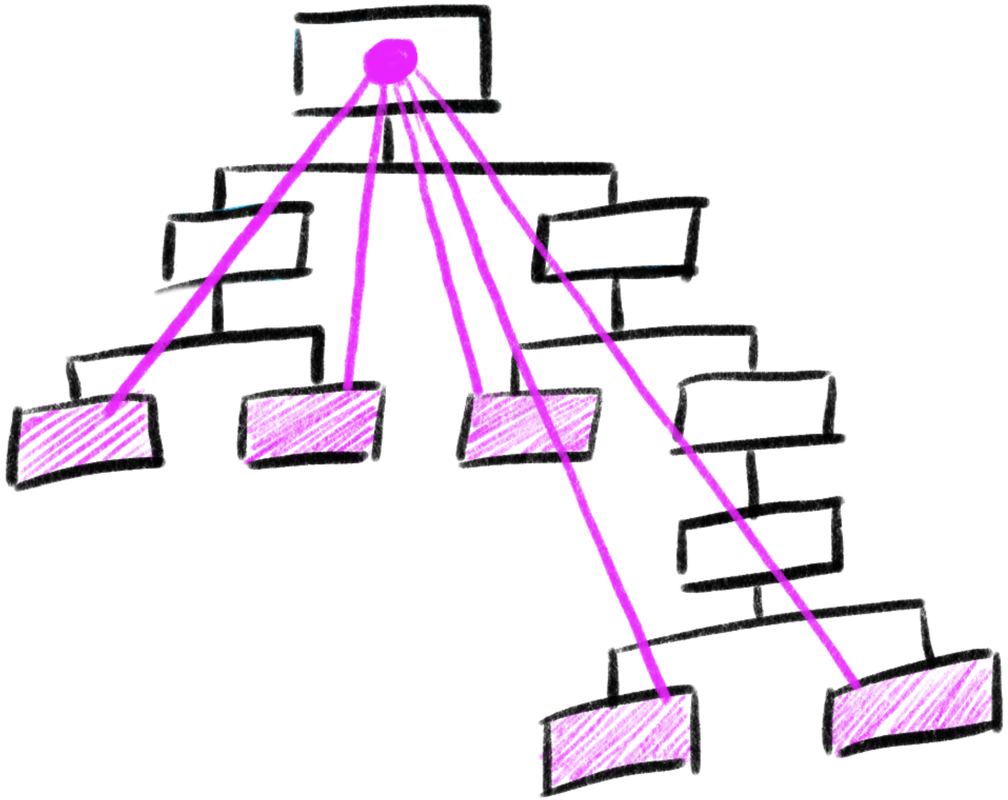
Passing props is a great way to explicitly pipe data through your UI tree to the components that use it. But it can become verbose and inconvenient when you need to pass some prop deeply through the tree, or if many components need the same prop. The nearest common ancestor could be far removed from the components that need data, and lifting state up that high can lead to a situation sometimes called “prop drilling.”



Context

LETS A PARENT COMPONENT PROVIDE DATA TO THE ENTIRE TREE BELOW IT





bit.ly/use-context

Hooks

Special functions that start with “use” and are only available while React is rendering

`useState()` Hook

STATE: A COMPONENT'S MEMORY

1 Declare a state variable

Call `useState` and pass the initial state to it. React will store the state that you passed, and give it back to you.

2 Update state on interaction

To change it, call the state setter function with the next state value. React will put that value into state instead.

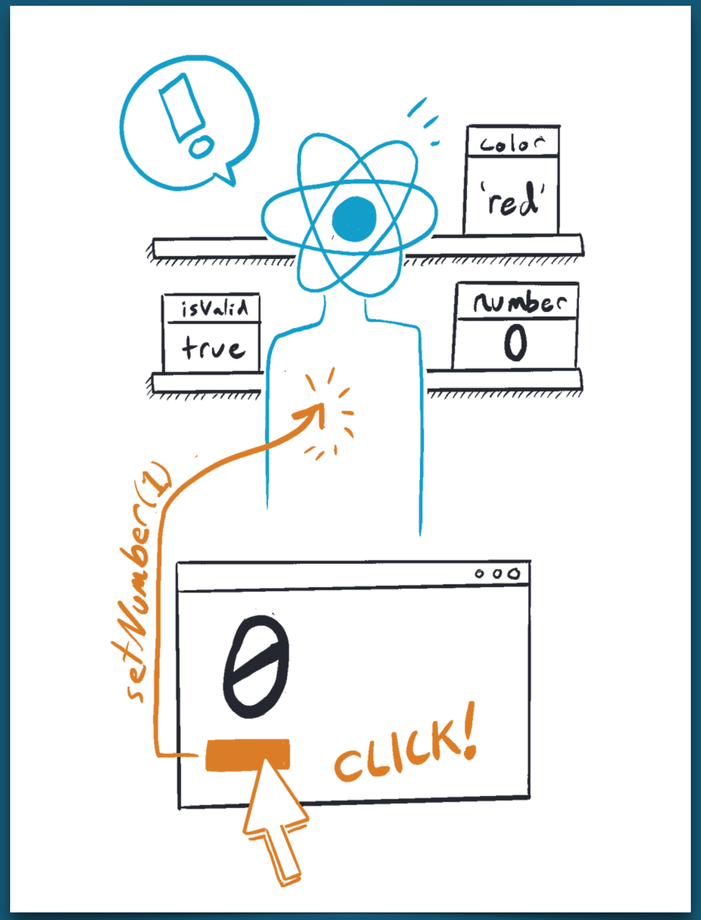
3 Render state in the UI

Use the state for rendering by putting it into the JSX.

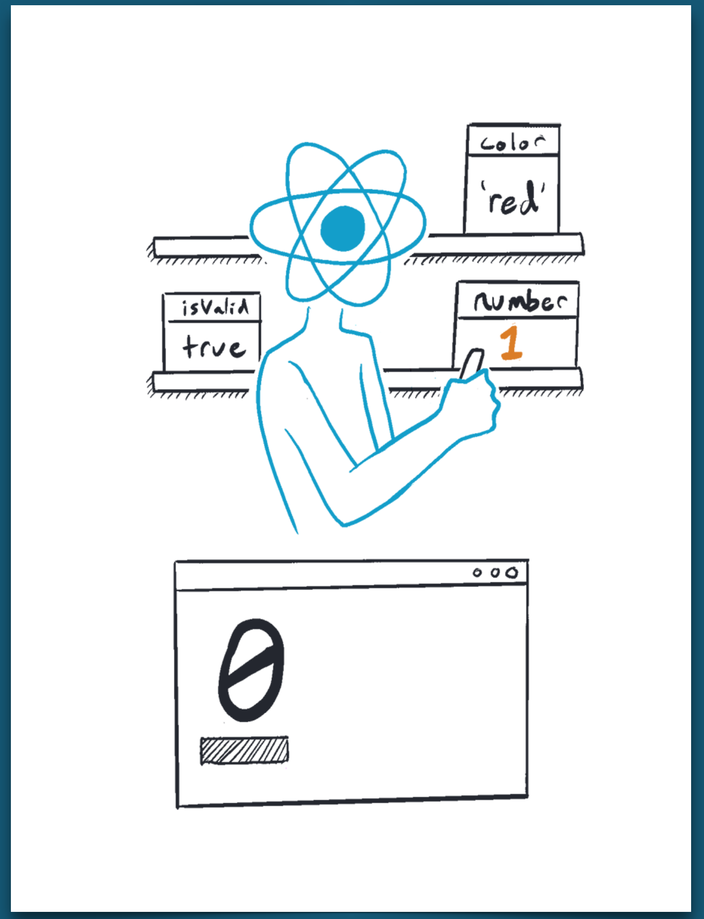
```
import { useState } from 'react';

function Counter() {
  1 const [count, setCount] = useState(0);

  function handleClick() {
    2 setCount(count + 1);
  }

  return (
    <button onClick={handleClick}>
      You pressed me {3 count} times
    </button>
  );
}
```

bit.ly/use-state

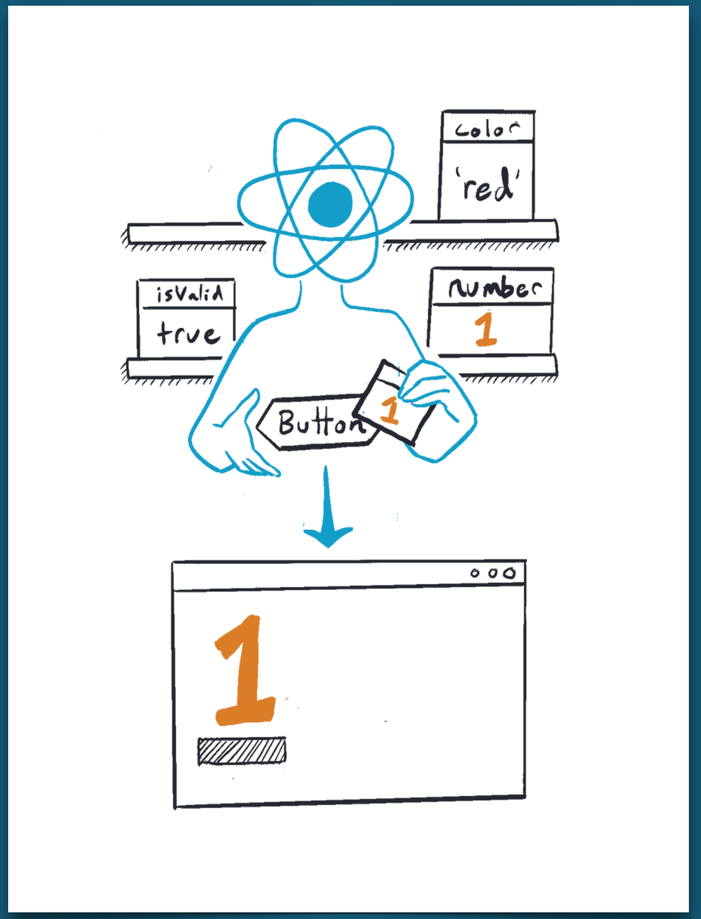
Updating State Triggers a Re-Render



React gets
a state update



React updates
the state value



React passes a snapshot
of the state value into the
component

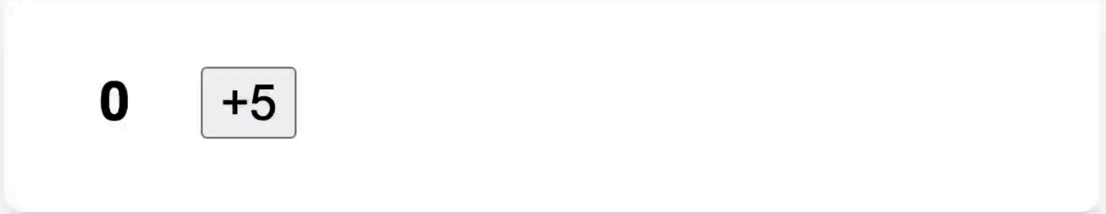
State over time

WHAT NUMBER WILL THE ALERT BE?

App.js

Download Reset Fork

```
1  import { useState } from 'react';
2
3  export default function Counter() {
4    const [number, setNumber] = useState(0);
5
6    return (
7      <>
8        <h1>{number}</h1>
9        <button onClick={() => {
10         setNumber(number + 5);
11         alert(number);
12       }}>+5</button>
13     </>
14   )
15 }
16
```



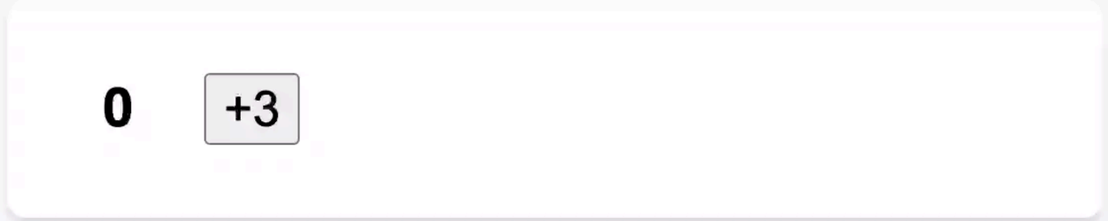
React batches state updates

REACT PROCESSES STATE UPDATES AFTER EVENT HANDLERS HAVE FINISHED RUNNING

App.js

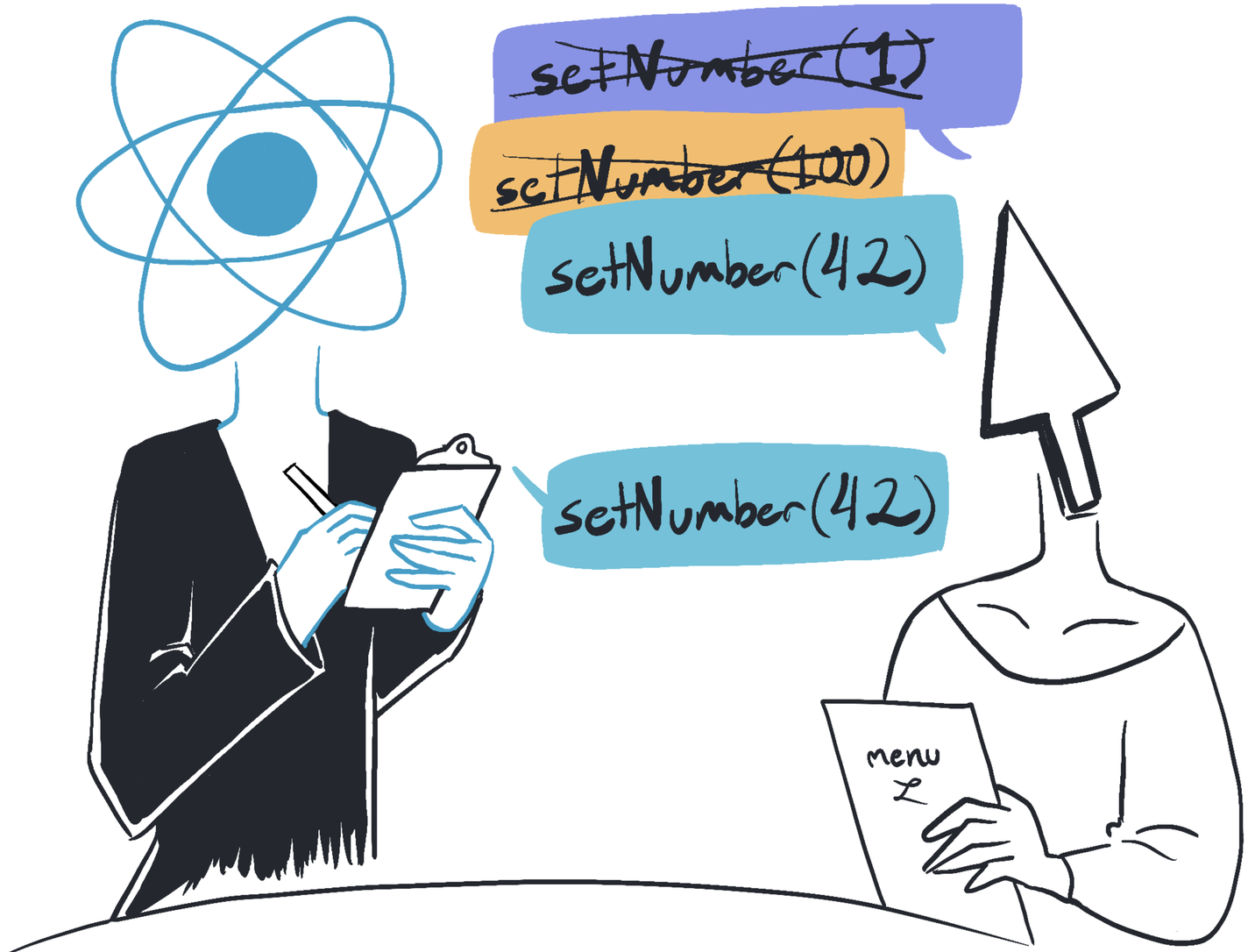
Download Reset Fork

```
1  import { useState } from 'react';
2
3  export default function Counter() {
4    const [number, setNumber] = useState(0);
5
6    return (
7      <>
8        <h1>{number}</h1>
9        <button onClick={() => {
10         setNumber(number + 1);
11         setNumber(number + 1);
12         setNumber(number + 1);
13       }}>+3</button>
14     </>
15   )
16 }
```



React batches state updates

REACT PROCESSES STATE UPDATES AFTER EVENT HANDLERS HAVE FINISHED RUNNING



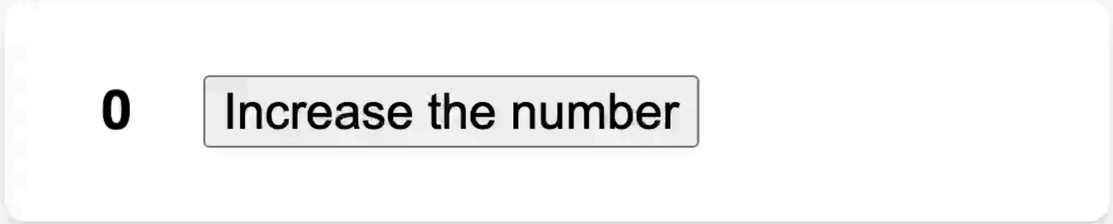
Replacing state after updating it

ADDS “REPLACE WITH 42” TO THE QUEUE, IGNORING WHAT’S ALREADY QUEUED

App.js

Download Reset Fork

```
1  import { useState } from 'react';
2
3  export default function Counter() {
4    const [number, setNumber] = useState(0);
5
6    return (
7      <>
8        <h1>{number}</h1>
9        <button onClick={() => {
10         setNumber(number + 99);
11         setNumber(42);
12       }}>Increase the number</button>
13     </>
14   )
15 }
16
```



Updater Function

UPDATE THE STATE MULTIPLE TIMES DURING THE SAME EVENT

1 **Setting state with next value**

Usually, when you set state, you replace it.

2 **Setting state with an updater**

But you can pass an updater function to transform it.

```
function handleClick() {
  setCount(1 123);

  setCount(2 c => c + 1); // Result: 124
}
```

bit.ly/updater-function

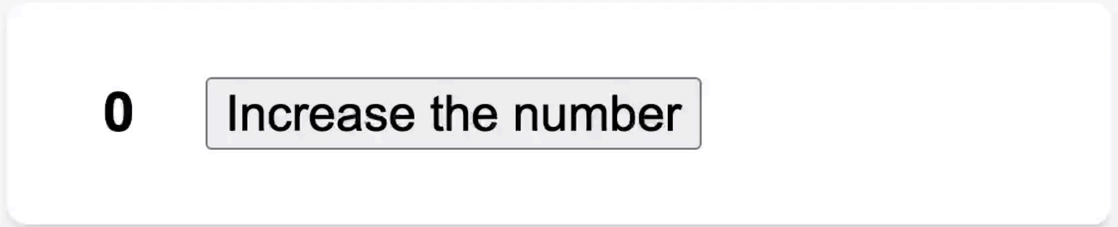
Updater Function

UPDATE THE STATE MULTIPLE TIMES DURING THE SAME EVENT

App.js

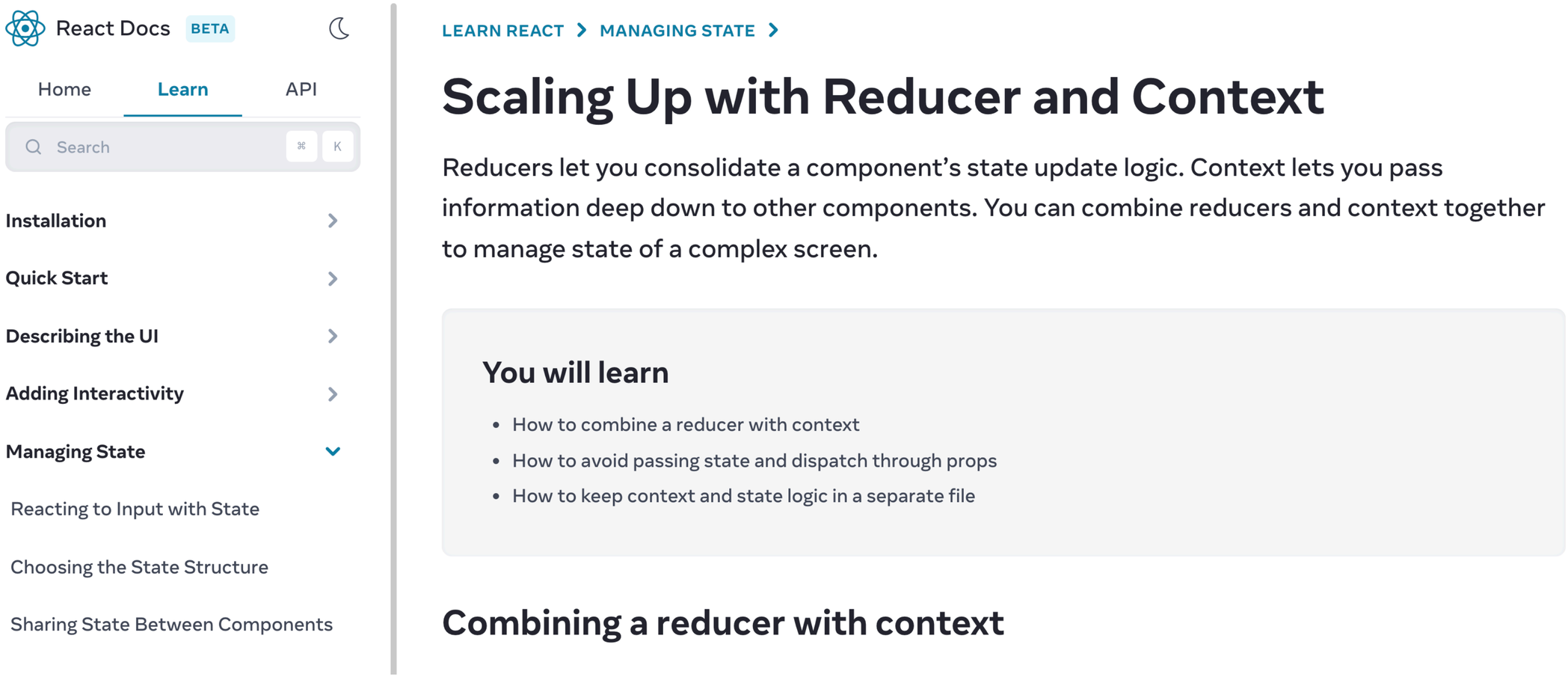
Download Reset Fork

```
1  import { useState } from 'react';
2
3  export default function Counter() {
4    const [number, setNumber] = useState(0);
5
6    return (
7      <>
8        <h1>{number}</h1>
9        <button onClick={() => {
10         setNumber(number + 5);
11         setNumber(n => n + 1);
12       }}>Increase the number</button>
13     </>
14   )
15 }
16
```

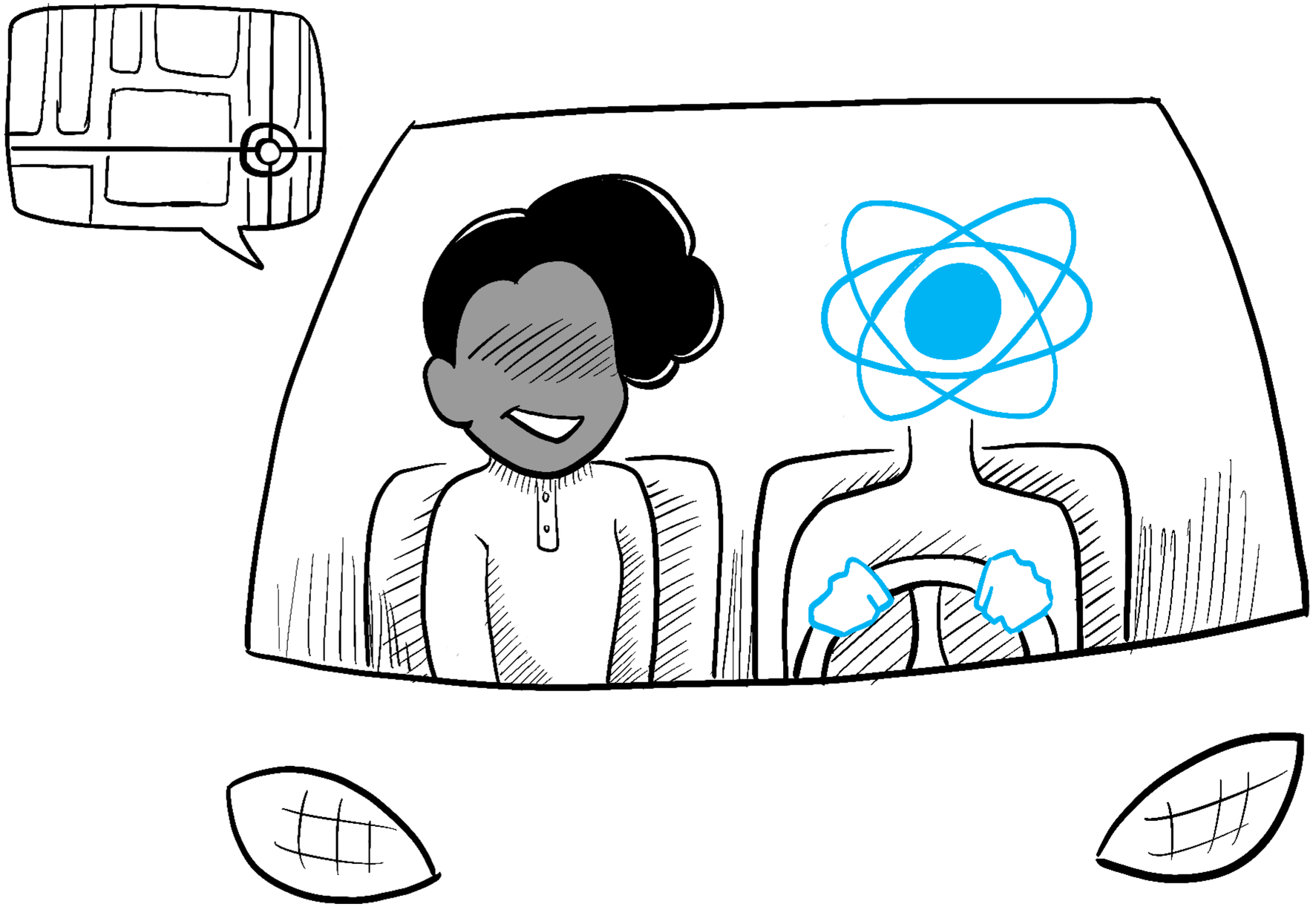


Reducer and Context for State Management

COMBINE CONTEXT WITH USEREDUCER YOU ESSENTIALLY END UP WITH A STATE MANAGEMENT SOLUTION



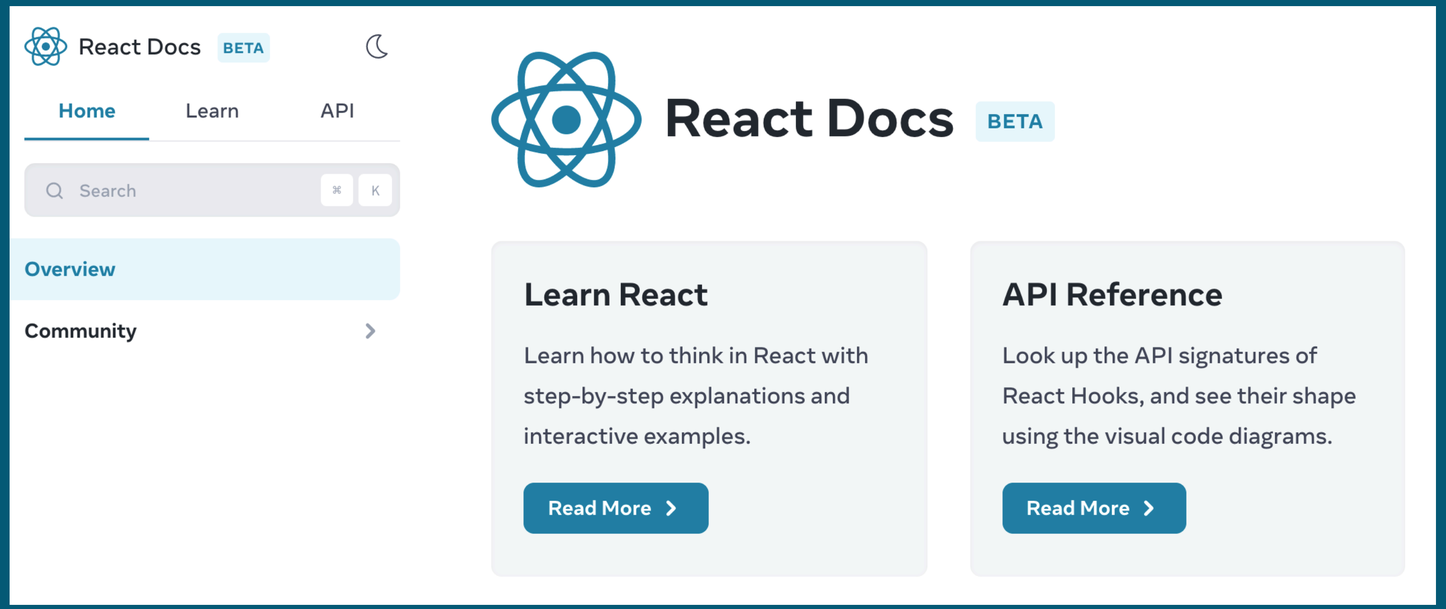
https://bit.ly/context-reducer



There has never been a better time to learn

Thank You

beta.reactjs.org



React Conf

@debs_obrien

debbie.codes