The Built in React Hooks useState, useEffect and useReducer

Peter Kellner

Developer, Consultant and Author

ReactAtScale.com @pkellner linkedin.com/in/peterkellner99

Three React Hooks Covered in This Module

useState

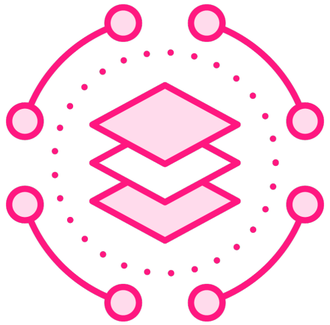
useEffect

useReducer

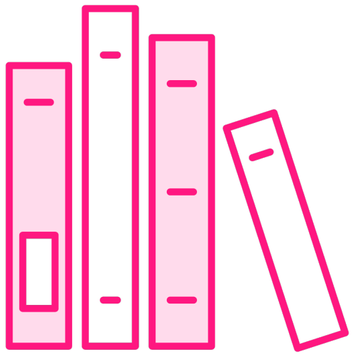
The 15 Built in React Hooks

The 15 Built in React Hooks

Basic Hooks



Additional Hooks

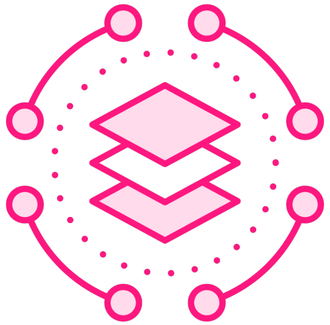


Library Hooks

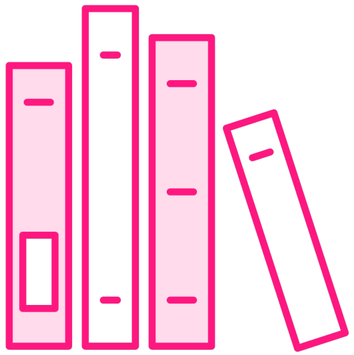
The 15 Built in React Hooks

Basic Hooks

useState, useEffect,
useContext

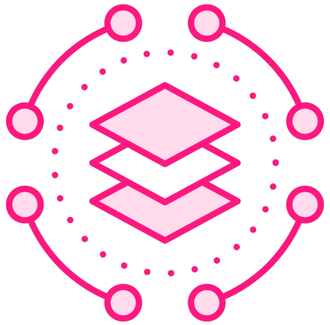


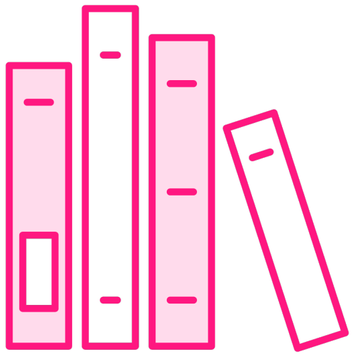
Additional Hooks



Library Hooks

The 15 Built in React Hooks





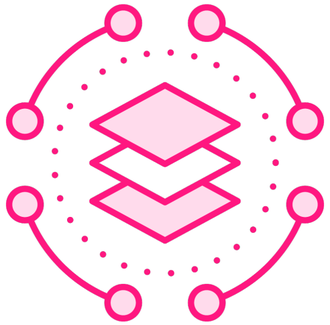
Basic Hooks

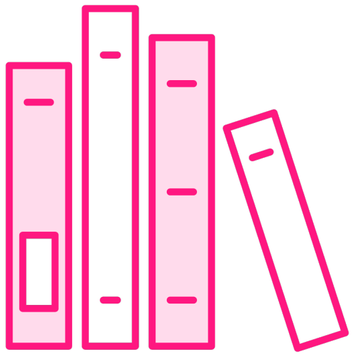
useState, useEffect,
useContext

Additional Hooks

Library Hooks

The 15 Built in React Hooks





Basic Hooks

Additional Hooks

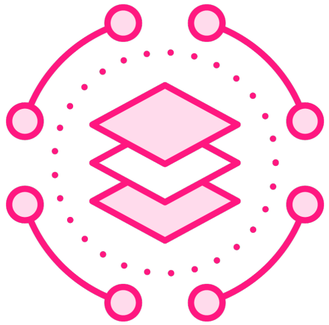
Library Hooks

useState, useEffect,
useContext

The 15 Built in React Hooks

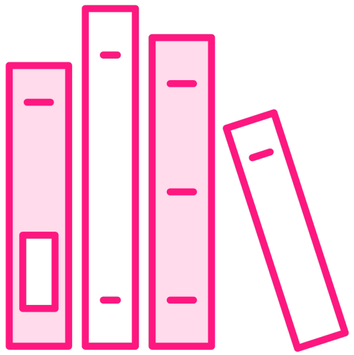
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId

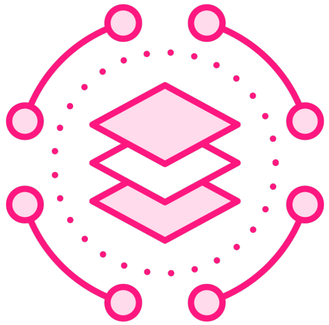


Library Hooks

The 15 Built in React Hooks

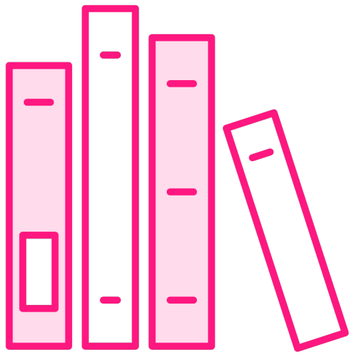
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId

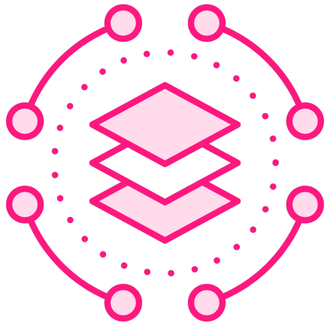


Library Hooks

The 15 Built in React Hooks

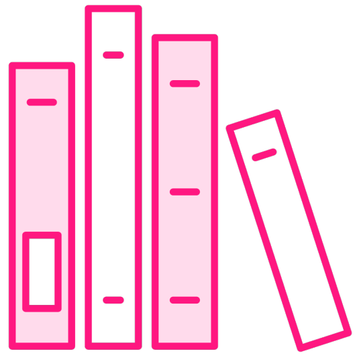
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId

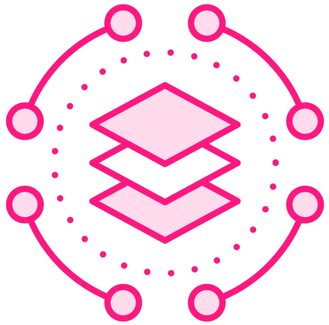


Library Hooks

The 15 Built in React Hooks

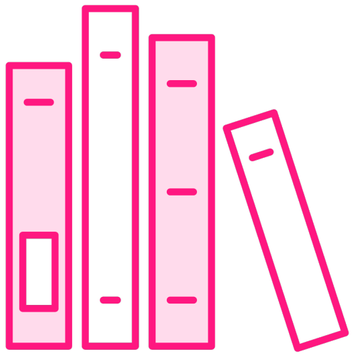
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId

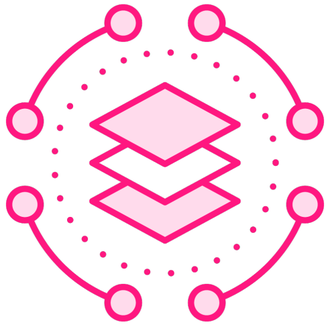


Library Hooks

The 15 Built in React Hooks

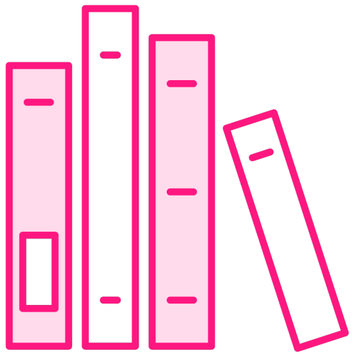
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId

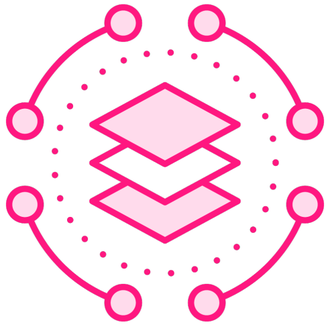


Library Hooks

The 15 Built in React Hooks

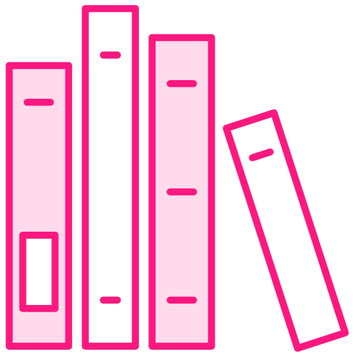
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId

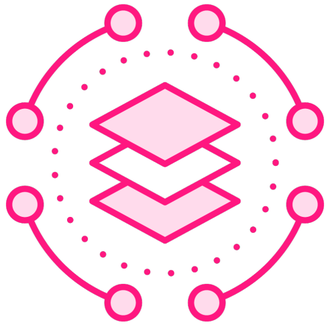


Library Hooks

The 15 Built in React Hooks

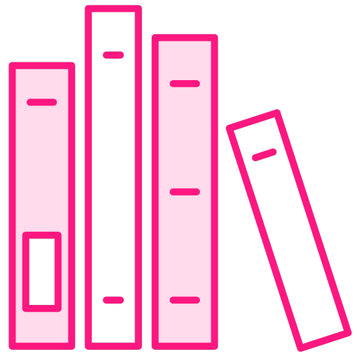
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId



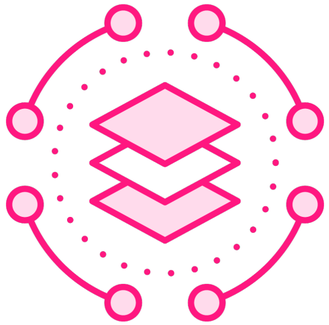
Library Hooks

useSyncExternalStore, useInsertionEffect

The 15 Built in React Hooks

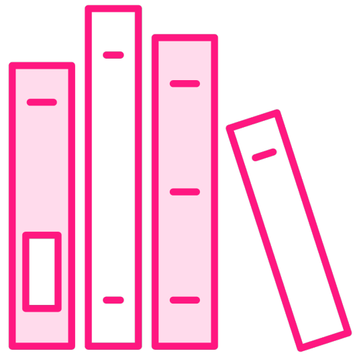
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId



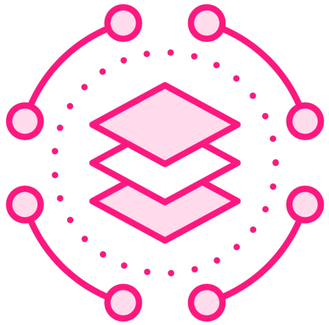
Library Hooks

useSyncExternalStore, useInsertionEffect

The 15 Built in React Hooks

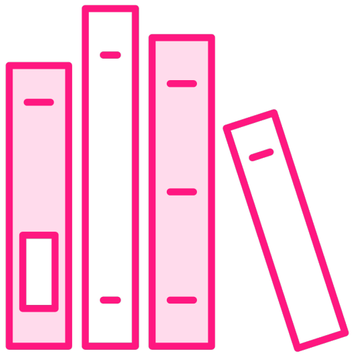
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId



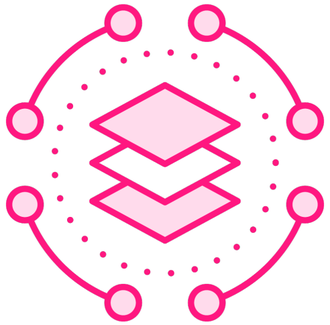
Library Hooks

useSyncExternalStore, useInsertionEffect

The 15 Built in React Hooks

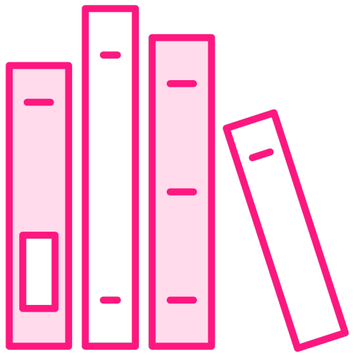
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId



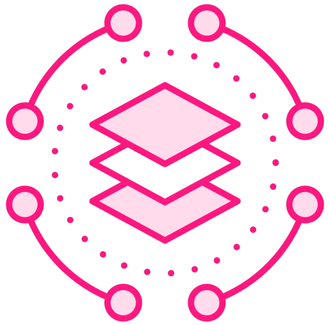
Library Hooks

useSyncExternalStore, useInsertionEffect

The 15 Built in React Hooks

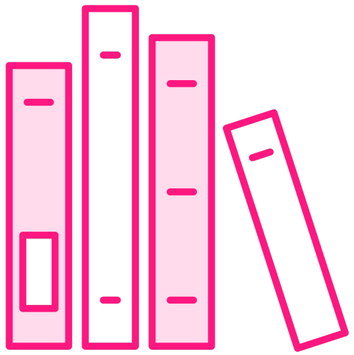
Basic Hooks

useState, useEffect, useContext



Additional Hooks

useReducer, useCallback, useMemo, useRef, useImperativeHandle, useLayoutEffect, useDebugValue, useDeferredValue, useTransition, useId



Library Hooks

useSyncExternalStore, useInsertionEffect

Basic Hooks

useState and useEffect Working Together to Manage State

Review of useState and useEffect

/pages/demo-review.js

```
import { useState } from "react";
export default function Demo() {
  const [cnt, setCnt] = useState(100);

  function incrementCnt() {
    setCnt(cnt + 1);
  }

  return <button onClick={incrementCnt}>
    {cnt}</button>;
}
```

```jsx
import { useState } from "react";
export default function Demo() {
  const [cnt, setCnt] = useState(100);

  function incrementCnt() {
    setCnt((previousState) => {
      return previousState + 1;
    });
  }

  return <button onClick={incrementCnt}>
    {cnt}</button>;
}
```

```jsx
import { useState, useEffect } from "react";
export default function Demo() {
  const [cnt, setCnt] = useState(100);

  function incrementCnt() {
    setCnt((previousState) => {
      return previousState + 1;
    });
  }

  return <button onClick={incrementCnt}>
    {cnt}</button>;
}
```

Benefits of Passing a Function into State Setter

Helpful when new state is based on a calculation from old state

Multiple useState calls will get batched

Guarantee previous state value is from before render

Benefits of Passing a Function into State Setter

Helpful when new state is based on a calculation from old state

Multiple useState calls will get batched

Guarantee previous state value is from before render

```jsx
import { useState, useEffect } from "react";
export default function Demo() {
  const [cnt, setCnt] = useState(100);

  useEffect(() => {});

  function incrementCnt() {
    setCnt((previousState) => {
      return previousState + 1;
    });
  }

  return <button onClick={incrementCnt}>
    {cnt}</button>;
}
```

```jsx
import { useState, useEffect } from "react";
export default function Demo() {
  const [cnt, setCnt] = useState(100);

  useEffect(() => {
    console.log
      ("example: subscribe to events");
  });

  function incrementCnt() {
    setCnt((previousState) => {
      return previousState + 1;
    });
  }

  return <button onClick={incrementCnt}>
    {cnt}</button>;
}
```

```jsx
import { useState, useEffect } from "react";
export default function Demo() {
  const [cnt, setCnt] = useState(100);

  useEffect(() => {
    console.log
      ("example: subscribe to events");
    return () => console.log("unsubscribe");
  });

  function incrementCnt() {
    setCnt((previousState) => {
      return previousState + 1;
    });
  }

  return <button onClick={incrementCnt}>
    {cnt}</button>;
}
```

```jsx
import { useState, useEffect } from "react";
export default function Demo() {
  const [cnt, setCnt] = useState(100);

  useEffect(() => {
    console.log
      ("example: subscribe to events");
    return () => console.log("unsubscribe");
  } ,[]);

  function incrementCnt() {
    setCnt((previousState) => {
      return previousState + 1;
    });
  }

  return <button onClick={incrementCnt}>
    {cnt}</button>;
}
```

END OF Review of useState and useEffect

/pages/demo-review.js

The Built in React Hook useReducer

Explaining reducers is hard

The reducer pattern is an odd pattern

It has been around a long time because it’s powerful

Next, the useReducer parameters and returns explained

The Built in React Hook useReducer

Explaining reducers is hard

The reducer pattern is an odd pattern

It has been around a long time because it’s powerful

Next, the useReducer parameters and returns explained

The Built in React Hook useReducer

Explaining reducers is hard

The reducer pattern is an odd pattern

It has been around a long time because it's powerful

Next, the useReducer parameters and returns explained

The Built in React Hook useReducer

Explaining reducers is hard

The reducer pattern is an odd pattern

It has been around a long time because it’s powerful

Next, the useReducer parameters and returns explained

The Built in React Hook useReducer

Explaining reducers is hard

The reducer pattern is an odd pattern

It has been around a long time because it’s powerful

Next, the useReducer parameters and returns explained

The Built in React Hook useReducer

```
const [state, dispatch] =
  useReducer(reducer, initialArg, init);
```

The Built in React Hook useReducer

```
const [state, dispatch] =
  useReducer(reducer, initialArg, init);
```

reducer: A function that takes in 2 parameters, state and action and returns a new state.

initialArg: The starting value of state.

init: A function that returns the starting state

The Built in React Hook useReducer

```
const [state, dispatch] =
  useReducer(reducer, initialArg, init);
```

reducer: A function that takes in 2 parameters, state and action and returns a new state.

initialArg: The starting value of state.

init: A function that returns the starting state

state: The returned local component state managed by the reducer

dispatch: The function that gets called whose parameters are passed into the reducer as the second parameter.

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState } from "react";
export default function demo() {
  const [cnt, setCnt] = useState(10);

  return <button onClick={
    () => setCnt(cnt + 1)}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useReducer } from "react";

function reducer(state, action) {
switch (action.type) {
case "increment":
return state + action.incrementValue;
default:
return action;
}
}

export default function demo() {
const [state, dispatch] =
useReducer(reducer, 10);
return (
<button
onClick={() =>
dispatch({
type: "increment",
incrementValue: 1,
})
}
```

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

export default function demo() {
  const [cnt, setCnt] =
    useState(10);

  return (
    <button onClick={
            () => setCnt(cnt + 1)}>
      {cnt}
    </button>
  )
}
```

```
import { useReducer } from "react";

function reducer(state, action) {
  switch (action.type) {
    case "increment":
      return state +
                action.incrementValue;
    default:
      return action;
  }
}

export default function demo() {
  const [state, dispatch] =
    useReducer(reducer, 10);
    return <button
      onClick={() =>
         dispatch({
           type: "increment",
           incrementValue: 1,
         })}>{state}</button>
}
```

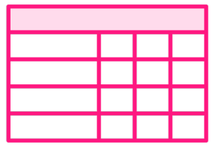
Title and Two Tabs

/pages/demo-no-reducer.js

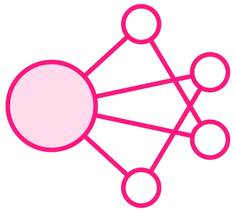
/pages/demo-with-reducer.js

What We Know About useReducer

How to call useReducer



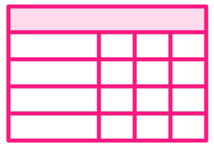
How to use the returned state



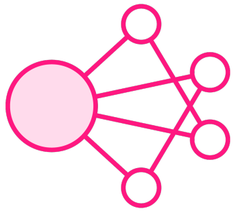
How to use the reducer's dispatch method

What We Know About useReducer

How to call useReducer



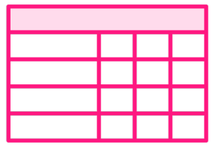
How to use the returned state



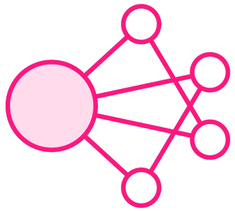
How to use the reducer's dispatch method

What We Know About useReducer

How to call useReducer



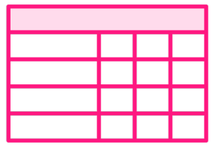
How to use the returned state



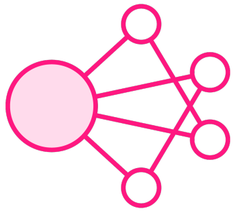
How to use the reducer's dispatch method

What We Know About useReducer

How to call useReducer



How to use the returned state



How to use the reducer’s dispatch method

There should be a reducer function that lets useReducer work exactly like useState

The code coming up is a little tricky and includes more advanced JavaScript then you've seen so far.

It's OK not to understand it all and just follow along.

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState } from "react";
export default function demo() {
  const [cnt, setCnt] = useState(10);

  return <button onClick={
    () => setCnt(cnt + 1)}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState, useReducer } from "react";
export default function demo() {
  const [cnt, setCnt] = useState(10);

  return <button onClick={
    () => setCnt(cnt + 1)}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState, useReducer } from "react";
export default function demo() {
  const [cnt, setCnt] =
    useReducer((_, action) => action, 10);

  return <button onClick={
    () => setCnt(cnt + 1)}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState, useReducer } from "react";
export default function demo() {
  const [cnt, setCnt] =
    useReducer((_, action) => {
      return action
    }, 10);

  return <button onClick={
    () => setCnt(cnt + 1)}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState, useReducer } from "react";
export default function demo() {
  const [cnt, setCnt] =
    useReducer((state, action) => {
      return action
    }, 10);

  return <button onClick={
    () => setCnt(cnt + 1)}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState, useReducer } from "react";
export default function demo() {
  const [cnt, setCnt] =
    useReducer((state, action) => {
      switch (action.type) {
        case "increment":
        default:
      }

    }, 10);

  return <button onClick={
    () => setCnt(cnt + 1)}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState, useReducer } from "react";
export default function demo() {
  const [cnt, setCnt] =
    useReducer((state, action) => {
      switch (action.type) {
        case "increment":
          return state + action.incrementValue;
        default:
          return action;
      }

    }, 10);

  return <button onClick={
    () => setCnt(cnt + 1)}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState, useReducer } from "react";
export default function demo() {
  const [cnt, setCnt] =
    useReducer((state, action) => {
      switch (action.type) {
        case "increment":
          return state + action.incrementValue;
        default:
          return action;
      }

    }, 10);

  return <button onClick={
    () => setCnt({
            type: "increment",
            incrementValue: 1,
          })}>{cnt}</button>
}
```

Learn useReducer Basics

/pages/demo-reducer.js

```
import { useState, useReducer } from "react";
export default function demo() {
  const [cnt, dispatch] =
    useReducer((state, action) => {
      switch (action.type) {
        case "increment":
          return state + action.incrementValue;
        default:
          return action;
      }

    }, 10);

  return <button onClick={
    () => dispatch({
            type: "increment",
            incrementValue: 1,
          })}>{cnt}</button>
}
```

Demo

Replace useState code in conference app speaker list component with useReducer

Up Next:

The Built-in React Hooks useRef, useContext, useMemo and useCallback