State Management

State Management

State Management

"data"

State Management

"data"

1. **Storing** the data
2. **Accessing** the data
3. **Modifying** the data

State Management

"data"

1. **Storing** the data
2. **Accessing** the data
3. **Modifying** the data

useState

State Management

"data"

1. **Storing** the data
2. **Accessing** the data
3. **Modifying** the data

useState

useReducer

State Management

"**data**"

1. **Storing** the data
2. **Accessing** the data
3. **Modifying** the data

useState

useReducer

useRef

State Management

"data"

1. **Storing** the data
2. **Accessing** the data
3. **Modifying** the data

useState **useReducer** **useRef**

useState

useState

Local data-management hook

useState

Local data-management hook

Formats of writing useState?

```
const data = useState(1)
```

```
const state = data[0]
```

```
const setState = data[1]
```

useState

Local data-management hook

Formats of writing useState?

```
const data = useState(1)
const state = data[0]
const setState = data[1]
```

OR

useState

Local data-management hook

Formats of writing useState?

```
const data = useState(1)
const state = data[0]
const setState = data[1]
```

OR

```
const [state, setState] = useState(1)
```

useState

Local data-management hook

```
const [state, setState] = useState(1)
```

useState

Local data-management hook

```
const [state, setState] = useState(1)
```

variable to access data

useState

Local data-management hook

function to update the data

```
const [state, setState] = useState(1)
```

variable to access data

useState

Local data-management hook

function to update the data

```
const [state, setState] = useState(1)
```

variable to access data

initial state to store the data

useState

Local data-management hook

function to **update** the data

```
const [state, setState] = useState(1)
```

variable to **access** data

initial state to **store** the data

useState

Local data-management hook

function to **update** the data

```
const [state, setState] = useState(1)
```

variable to **access** data

initial state to **store** the data

The initial state can store, both "**primitive**" and "**non-primitive**" data-types

useState

Ways of initialising an useState hook

useState

Ways of initialising an useState hook

1. **Passing the initial data as an argument**

useState

Ways of initialising an useState hook

1. **Passing the initial data as an argument**

```
const [state, setState] = useState(1)
```

useState

Ways of initialising an useState hook

2. **Passing the variable containing the initial data**

```
const data = number || 2
```

```
const [state, setState] = useState(data)
```

useState

Ways of initialising an useState hook

3. **Passing the data received from the props**

```
const Component = ({count}) => {
  const [state, setState] = useState(count)
  return <div>{count}</div>
}
```

useState

Ways of initialising an useState hook

4. **Return value of an initialising function**

```
const [state, setState] = useState(() => {
 return 23;
})
```

useReducer

useReducer

→ data-management hook

useReducer

$\searrow$ data-management hook

```
const [state, setState] = useReducer(() => {}, 30)
```

useReducer

$\xrightarrow{}$ data-management hook

```
const [state, dispatch] = useReducer(() => {}, 30)
```

useReducer

data-management hook

```
const [state, dispatch] = useReducer(() => {}, 30)
```

variable to access data

useReducer

data-management hook

function to dispatch action obj to the reducer function

```
const [state, dispatch] = useReducer(() => {}, 30)
```

variable to access data

useReducer

data-management hook

function to dispatch action obj to the reducer function

```
const [state, dispatch] = useReducer(() => {}, 30)
```

variable to access data

reducer function, to modify the data

useReducer

data-management hook

function to dispatch action obj to the reducer function

```
const [state, dispatch] = useReducer(() => {}, 30)
```

variable to access data

reducer function, to modify the data

initial state to store the data

useReducer

data-management hook

function to dispatch action obj to the reducer function

```
const [state, dispatch] = useReducer(() => {}, 30)
```

variable to **access** data

reducer function, to **modify** the data

initial state to **store** the data

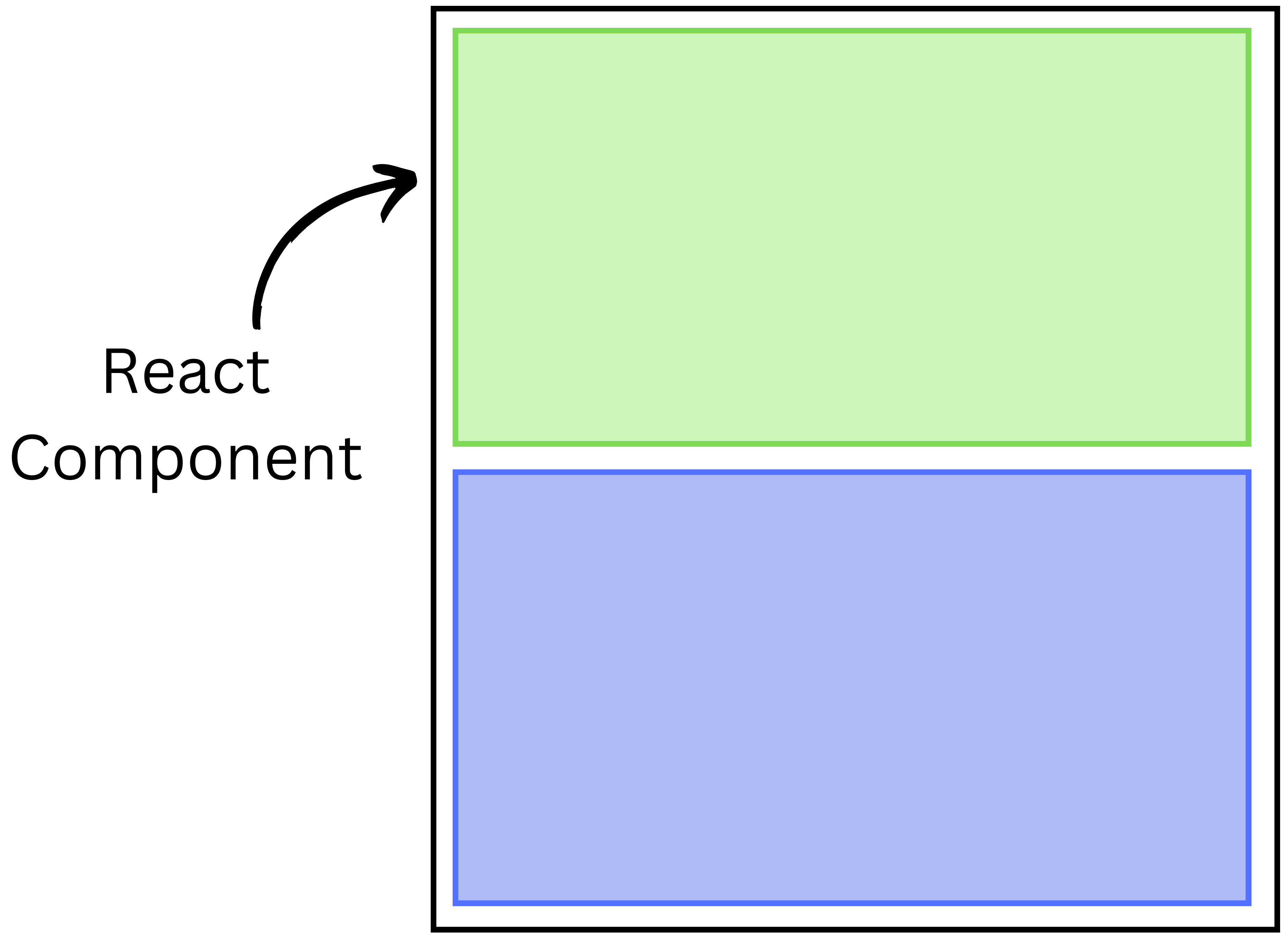
useReducer

$\searrow$ data-management hook

Why should we use **useReducer** to manage state when we already have **useState**?

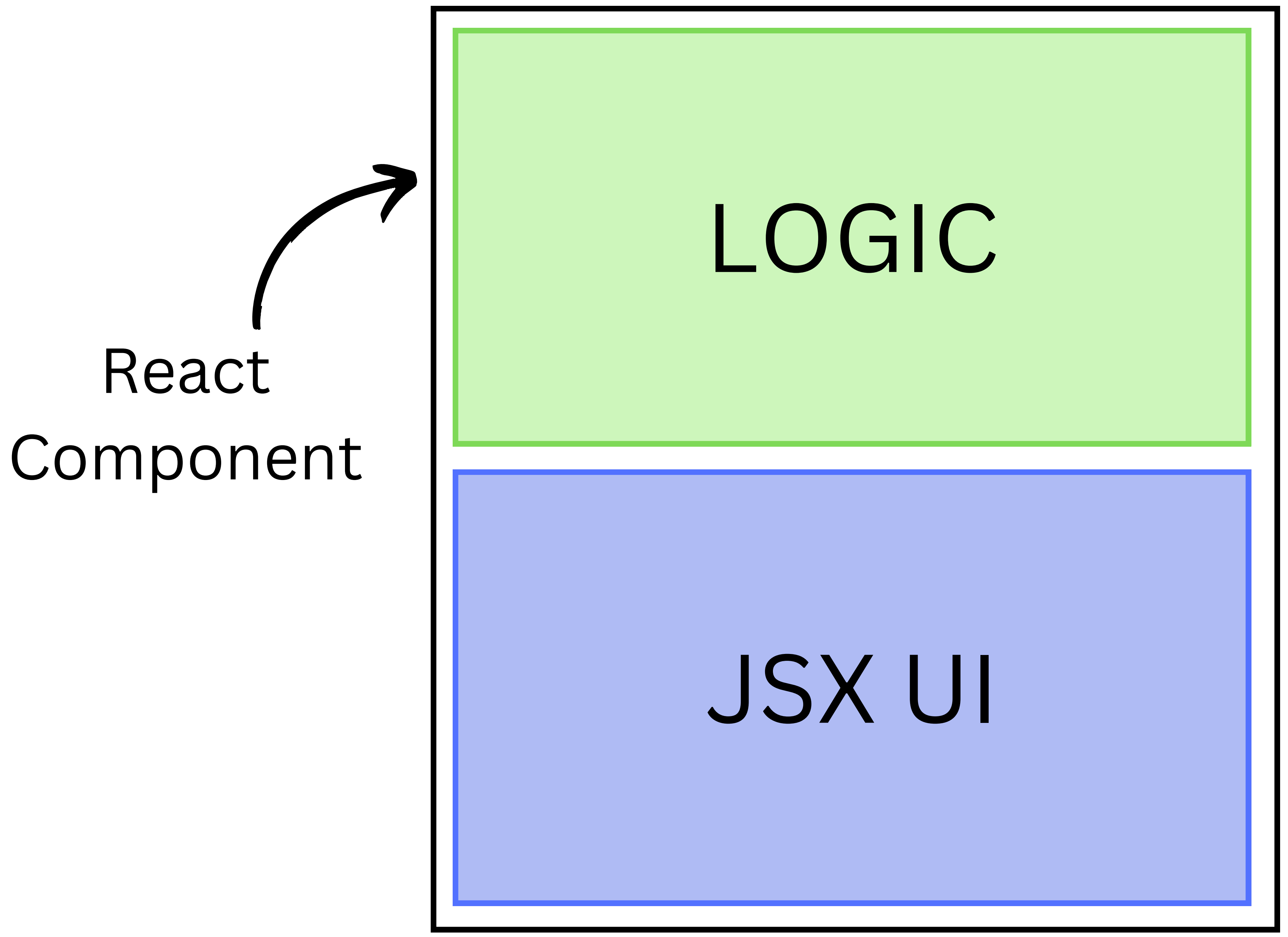
useReducer

Why should we use **useReducer** to manage state when we already have **useState**?



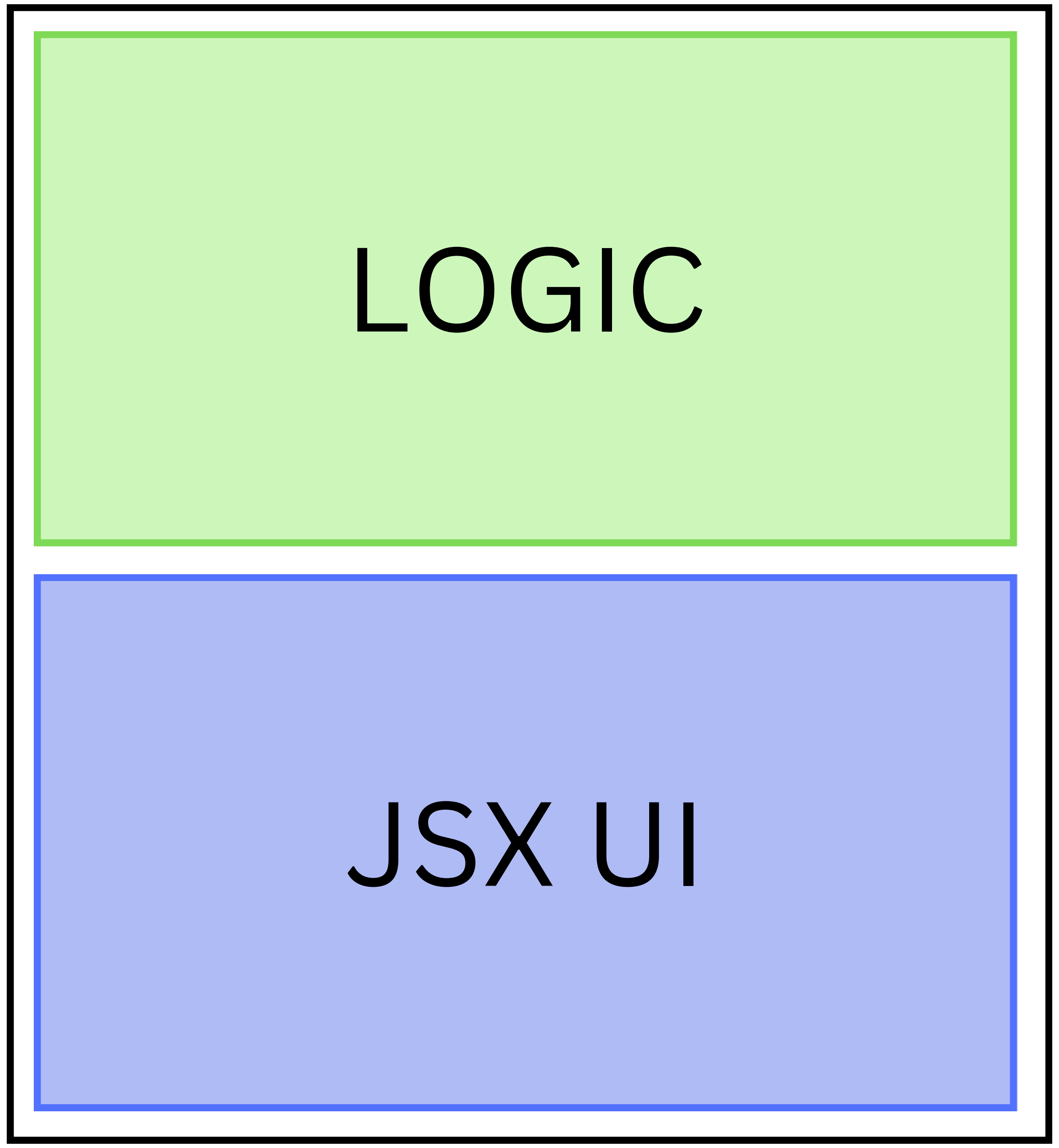
useReducer

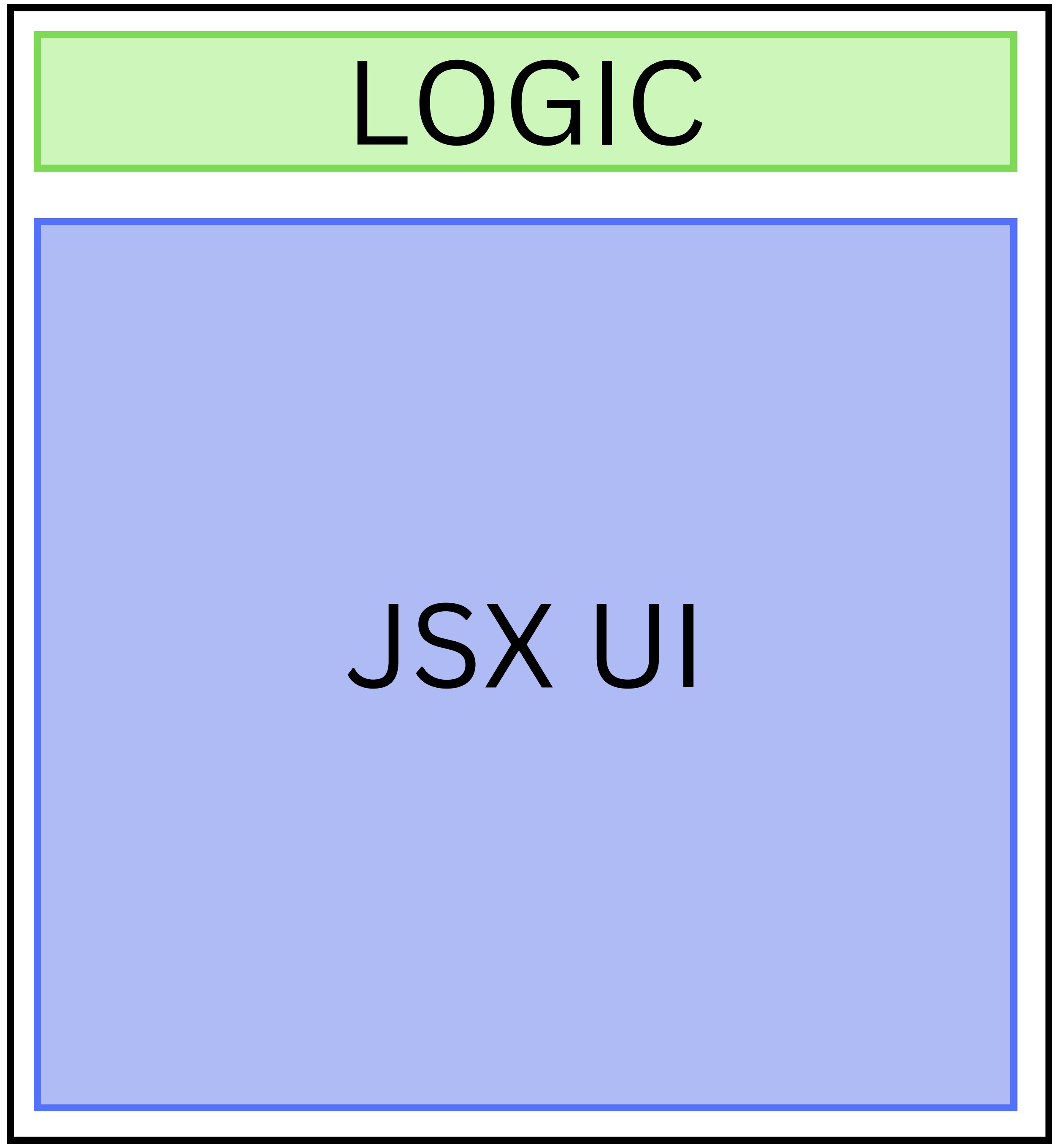
Why should we use **useReducer** to manage state when we already have **useState**?



useReducer

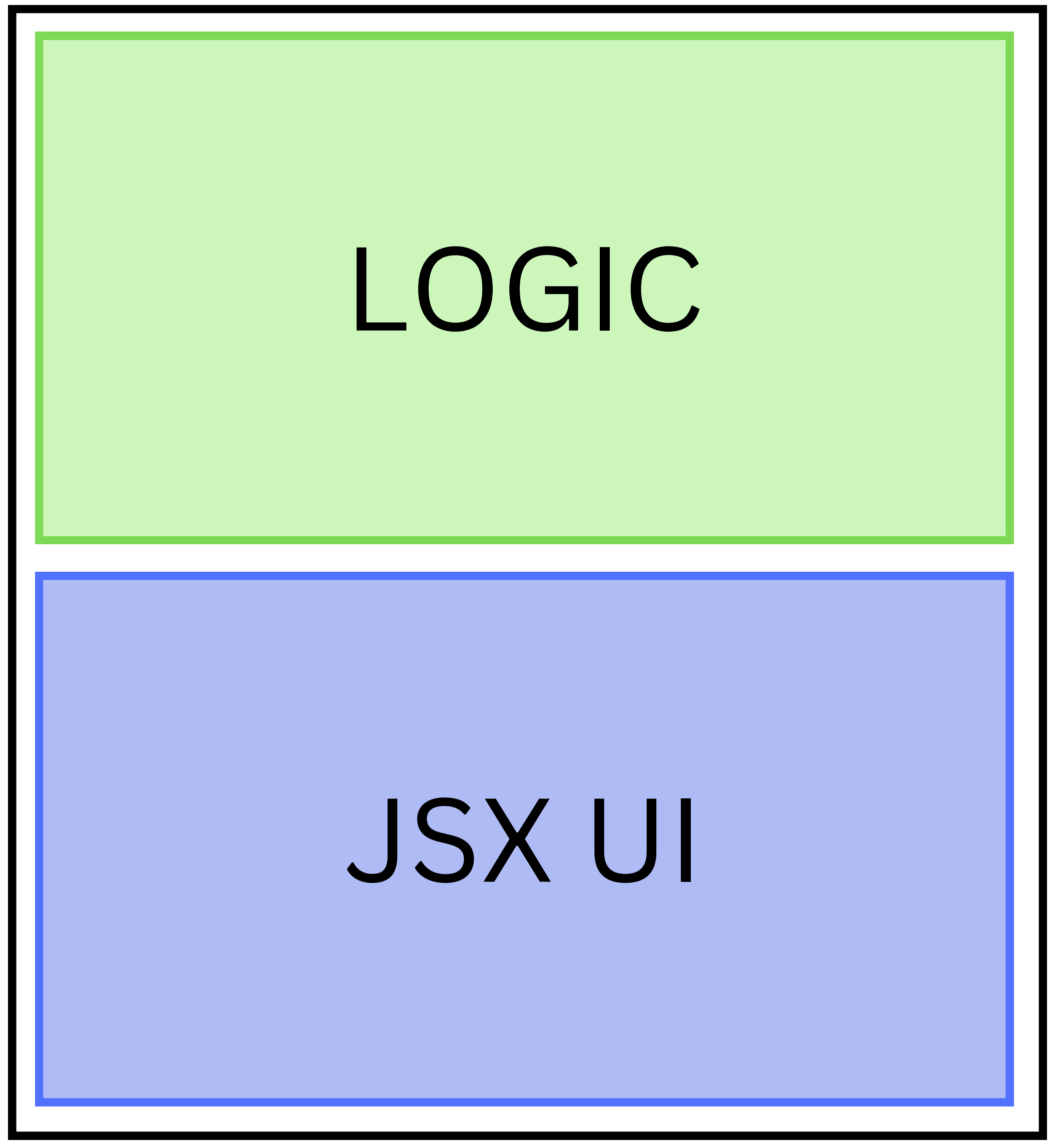
Why should we use **useReducer** to manage state when we already have **useState**?



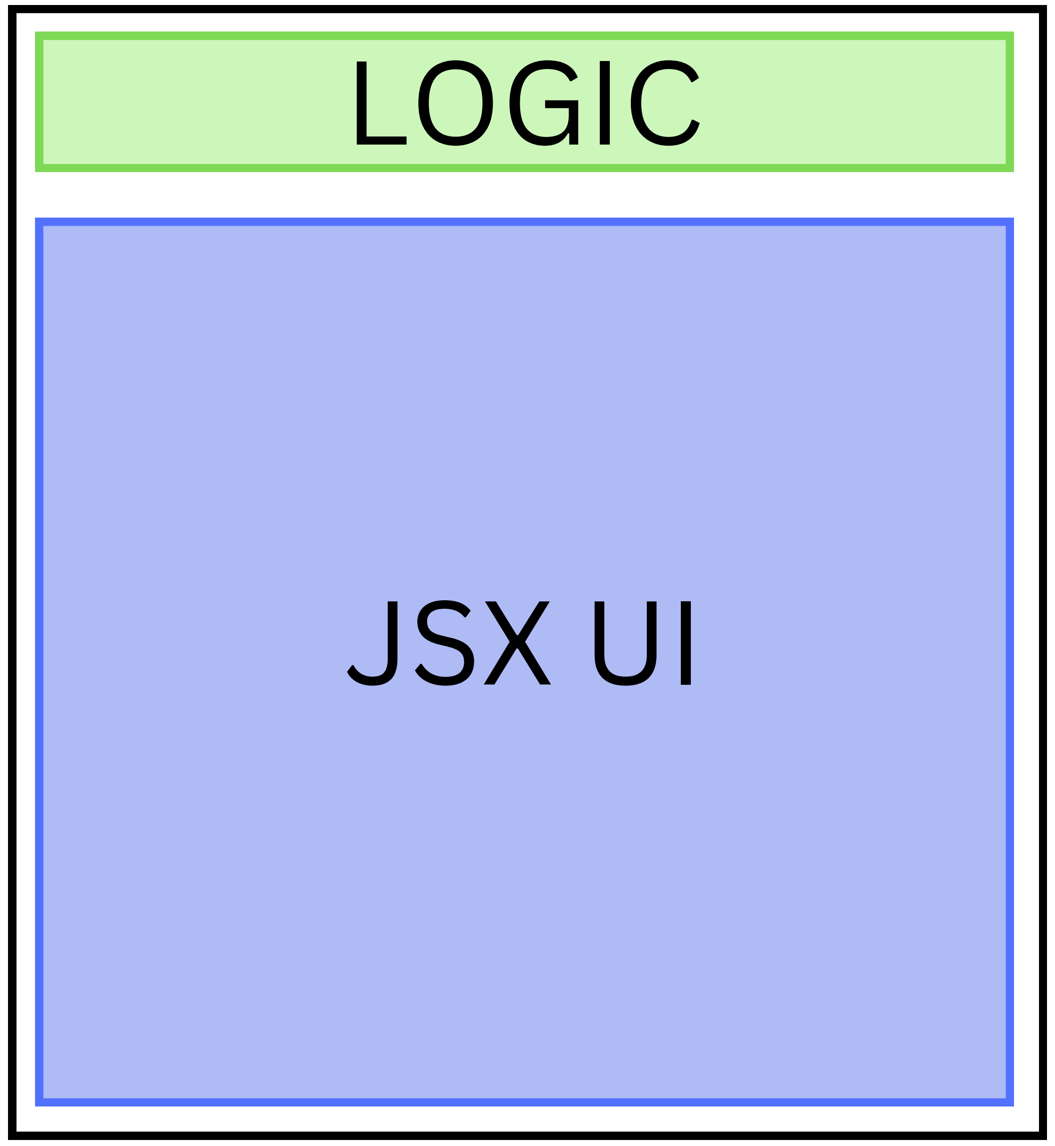


useReducer

Why should we use **useReducer** to manage state when we already have **useState**?



useState



useReducer

useReducer

Why should we use **useReducer** to manage state when we already have **useState**?

How to handle 10 input boxes in a form?

.
.
.

useReducer

Why should we use **useReducer** to manage state when we already have **useState**?

How to handle 10 input boxes in a form?

```
useState()
useState()
useState()
    .
    .
    .
useState()
```

useReducer

Why should we use **useReducer** to manage state when we already have **useState**?

How to handle 10 input boxes in a form?

```
useState()
useState()
useState()
.
.
.
useState()
```

OR

useReducer

Why should we use **useReducer** to manage state when we already have **useState**?

How to handle 10 input boxes in a form?

```
useState()
useState()
useState()
.
.
.
useState()
```

OR

```
useReducer()
```

useReducer

Why should we use **useReducer** to manage state when we already have **useState**?

```
setState( logic )
```

useState

useReducer

Why should we use **useReducer** to manage state when we already have **useState**?

outsourcing the logic handling

$$\begin{tikzcd}
\texttt{setState(logic)} \arrow[r, bend left] & \texttt{reducer function}
\end{tikzcd}$$

useState

useReducer

useRef

useRef

$\hookrightarrow$ data-management hook

useRef

$\searrow$ data-management hook

```
const ref = useRef(intialData)
```

useRef

data-management hook

```
const ref = useRef(intialData)
```

```
{current: initialData}
```

useRef

$\searrow$ data-management hook

```
const ref = useRef(intialData)
```

Access the data: `ref.current`

useRef

→ data-management hook

```
const ref = useRef(intialData)
```

Access the data: `ref.current`

Modify the data: `ref.current = 'new data'`