STATE

REACT & NEXT.JS OFFICAL DOCS STUDY

DongMin Kim

CONTENTS

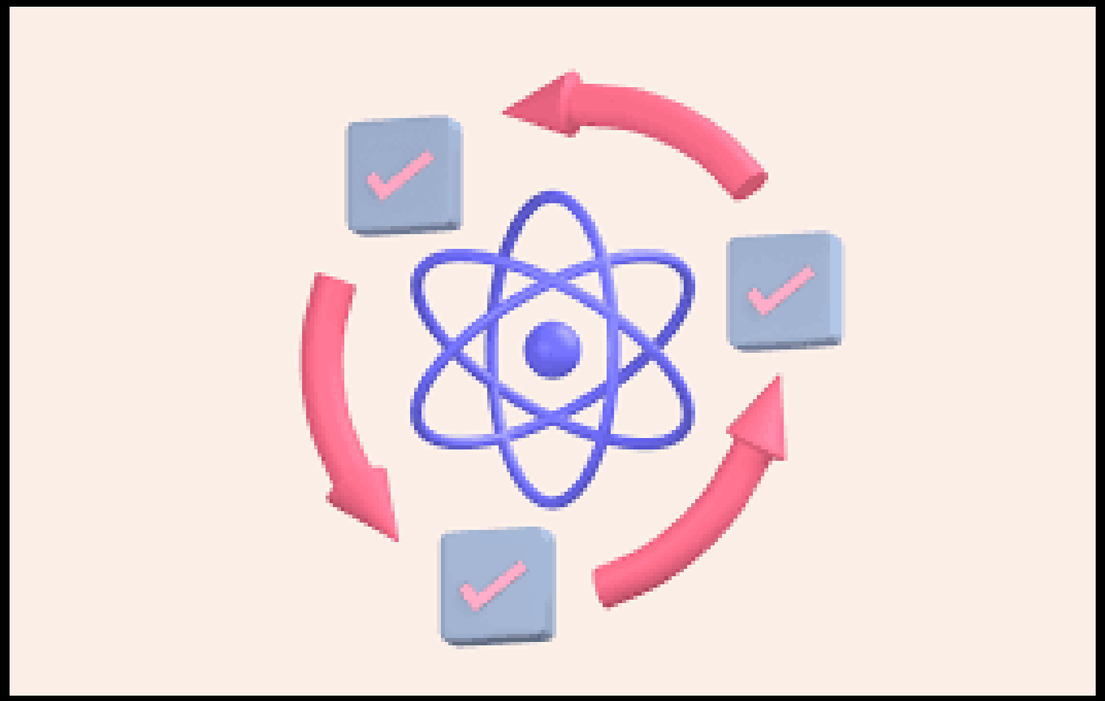
- **State**
  - Data Binding
  - Rerendering
  - Asynchronous update
  - Isolated / Private
- **Official Docs**

GOALS

- **What is State?**
- **Characteristics of State**
- **Caveats of State**

State

React is smart



JavaScript

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <div id="content">Count :</div>
    <button id="btn">Click me</button>
  </body>
</html>
```

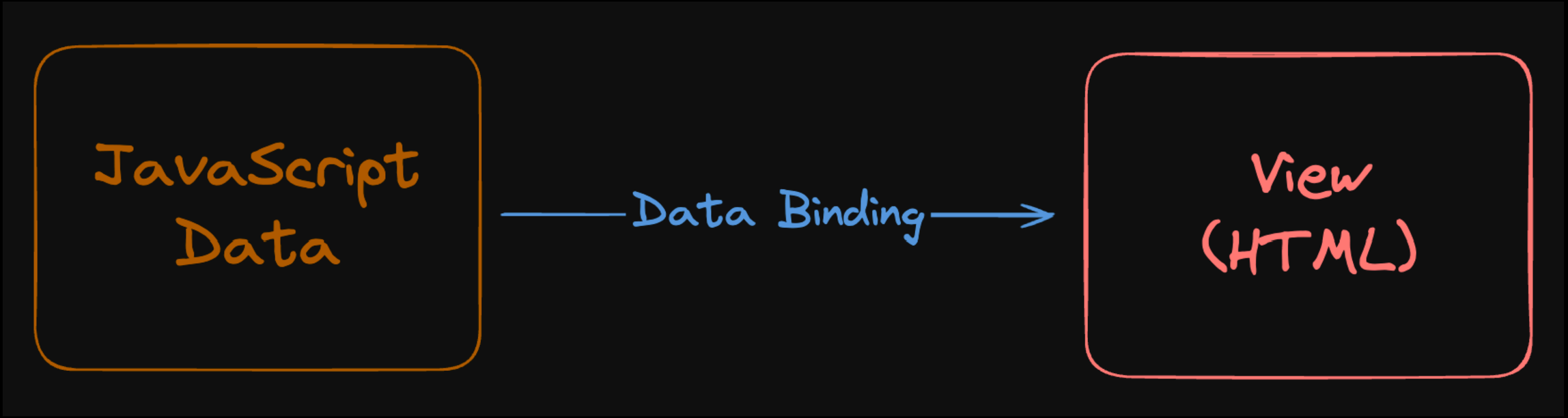
JavaScript

```
<script>
  let count = 0;

  function handleClick() {
    count++;

    rerender();
  }

  function rerender() {
    const content = document.getElementById("content");

    content.innerHTML = `Count : ${count}`;
  }

  document.getElementById("btn").addEventListener("click", handleClick);
</script>
```

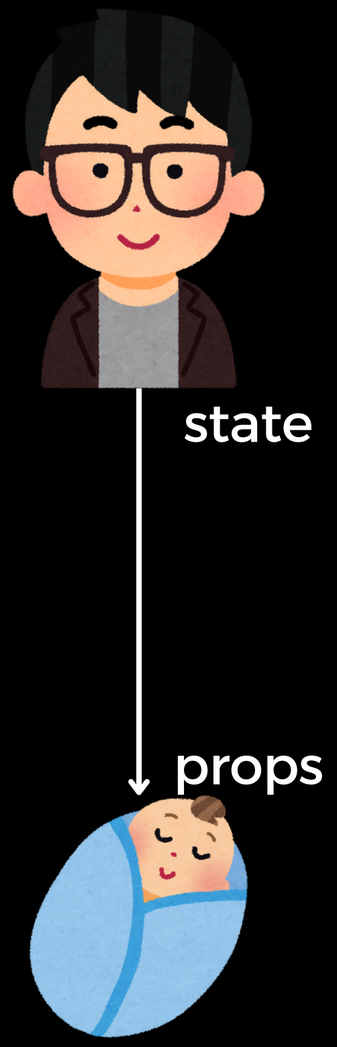
React

```
1  const [count, setCount] = useState(0);
2
3  const handleClick = () => {
4    setCount(count + 1);
5  };
```

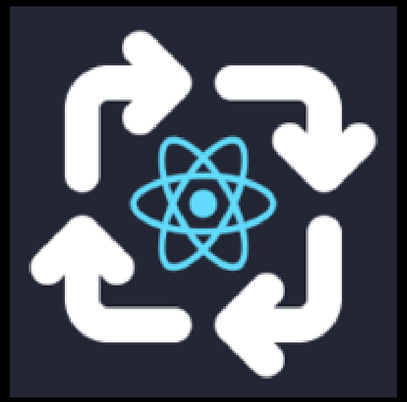
Data Binding



Data Binding



State



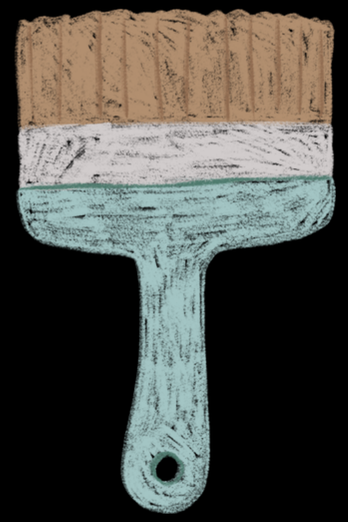
Rerendering

Asynchrounous update

Isolated / Private

Rerendering

```
const [count, setCount] = useState(0);

const handleClick = () => {
  setCount(count + 1);
};
```



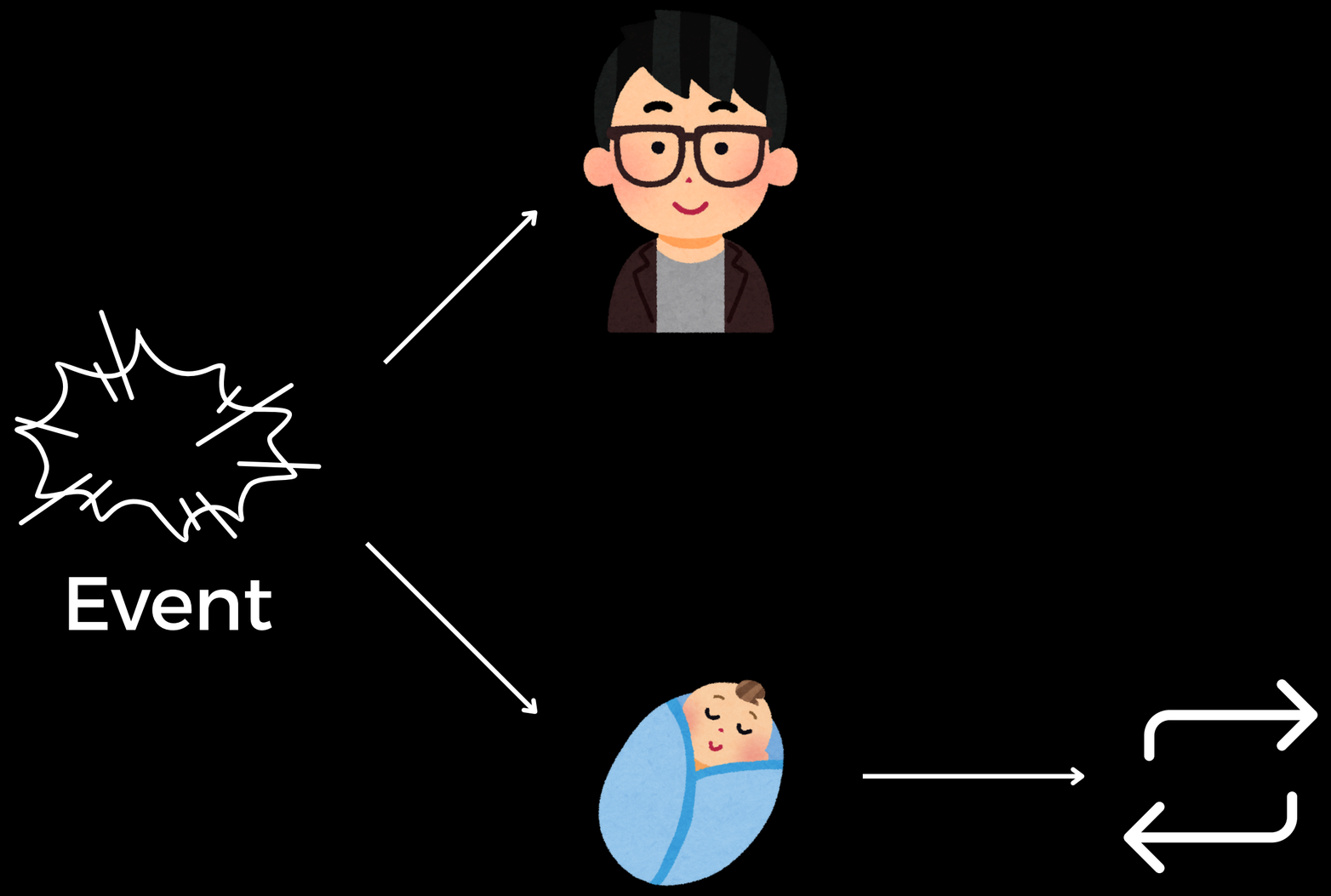
DON'T FORGET

Asynchronous Update

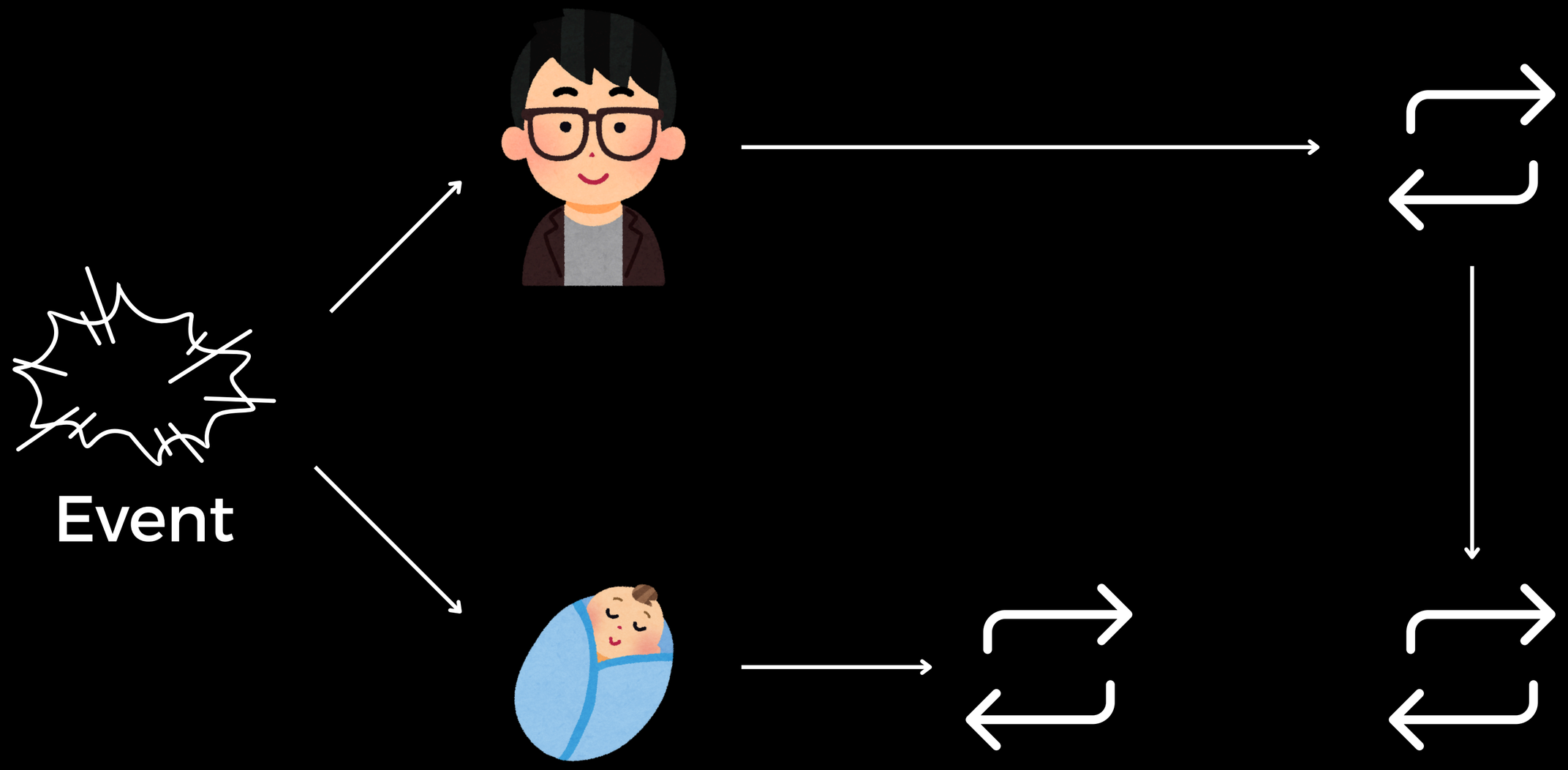
```
import { useState } from "react";

function App() {
  const [limit, setLimit] = useState(0);
  const [count, setCount] = useState(0);

  const handleClick = () => {
    setLimit(limit + 1);
    if (limit < 3) {
      setCount(count + 1);
    }
  };

  return (
    <div>
      <div>count : {count}</div>
      <button onClick={handleClick}>increment</button>
    </div>
  );
}

export default App;
```

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

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

  const handleClick = () => {
    setCount(count + 1);
    setCount(count + 1);
    setCount(count + 1);
    setCount(count + 1);
  };

  return (
    <div>
      <div>count : {count}</div>
      <button onClick={handleClick}>increment</button>
    </div>
  );
}

export default App;
```

Asynchronous Update



Asynchronous Update

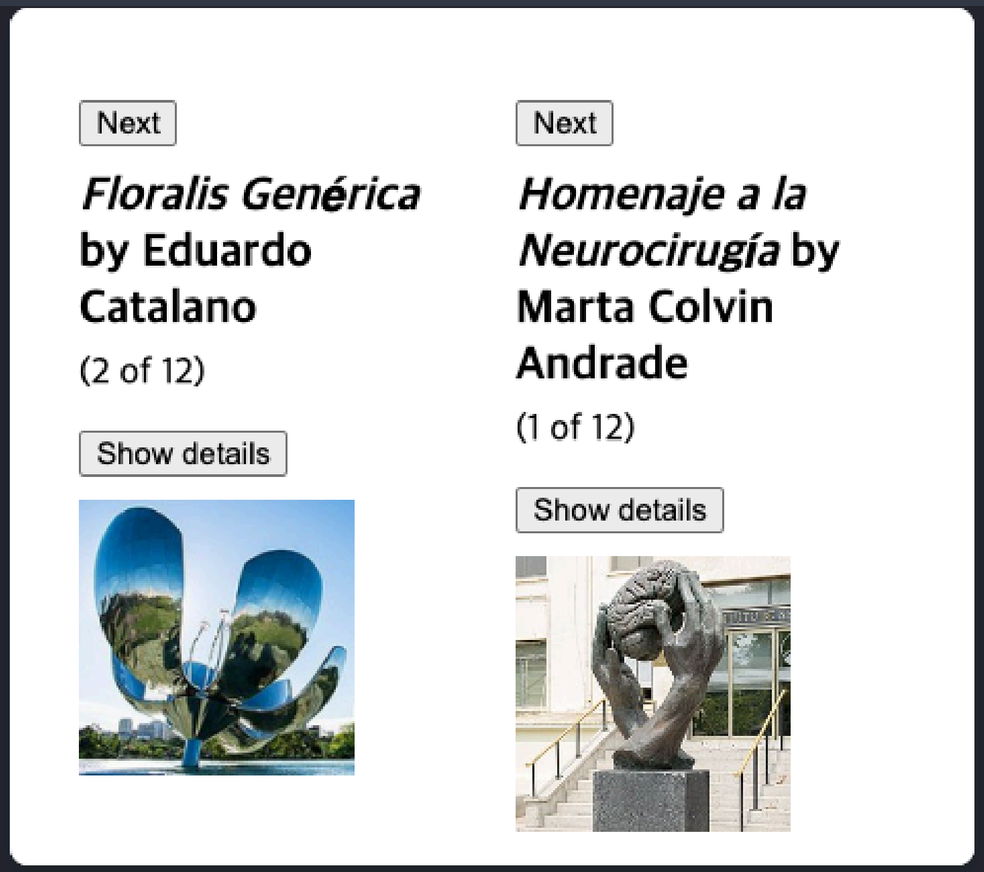


Isolated / Private

App.js Gallery.js data.js

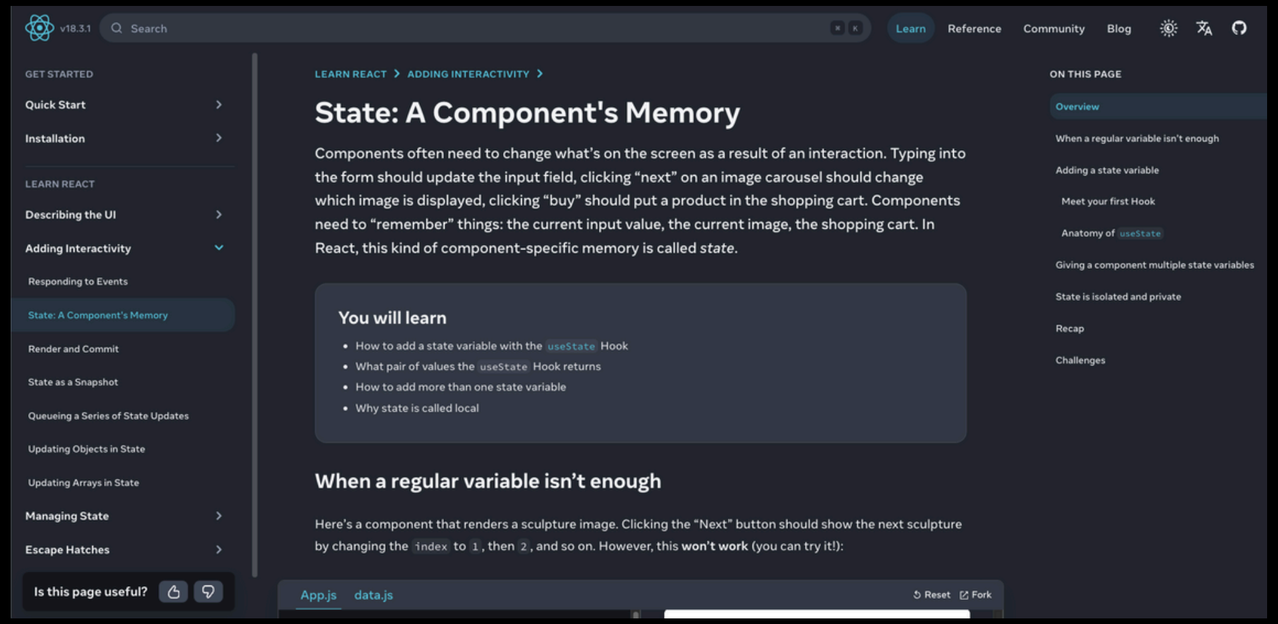
Reset Fork

```
1  import Gallery from './Gallery.js';
2
3  export default function Page() {
4    return (
5      <div className="Page">
6        <Gallery />
7        <Gallery />
8      </div>
9    );
10 }
11
12
```



Official Docs

State



☒Rules of Hooks

Pitfall

Hooks—functions starting with `use`—can only be called at the top level of your components or your own Hooks. You can't call Hooks inside conditions, loops, or other nested functions. Hooks are functions, but it's helpful to think of them as unconditional declarations about your component's needs. You "use" React features at the top of your component similar to how you "import" modules at the top of your file.

☒Rules of Hooks

Only Call Hooks at the Top Level

Don't call Hooks inside loops, conditions, or nested functions. Instead, always use Hooks at the top level of your React function, before any early returns. By following this rule, you ensure that Hooks are called in the same order each time a component renders. That's what allows React to correctly preserve the state of Hooks between multiple `useState` and `useEffect` calls. (If you're curious, we'll explain this in depth below.)

Only Call Hooks from React Functions

Don't call Hooks from regular JavaScript functions. Instead, you can:

- ✅ Call Hooks from React function components.
- ✅ Call Hooks from custom Hooks (we'll learn about them on the next page).

By following this rule, you ensure that all stateful logic in a component is clearly visible from its source code.

⊠Rules of Hooks

DEEP DIVE

How does React know which state to return?

Hide Details

You might have noticed that the `useState` call does not receive any information about *which* state variable it refers to. There is no “identifier” that is passed to `useState`, so how does it know which of the state variables to return? Does it rely on some magic like parsing your functions? The answer is no.

Instead, to enable their concise syntax, Hooks **rely on a stable call order on every render of the same component.** This works well in practice because if you follow the rule above (“only call Hooks at the top level”), Hooks will always be called in the same order. Additionally, a linter plugin catches most mistakes.

Internally, React holds an array of state pairs for every component. It also maintains the current pair index, which is set to `0` before rendering. Each time you call `useState`, React gives you the next state pair and increments the index. You can read more about this mechanism in React Hooks: Not Magic, Just Arrays.

⊠Rules of Hooks

```
1  const App = () => {
2    const [name, setName] = useState("John Doe");
3    const [age, setAge] = useState(25);
4    const [address, setAddress] = useState("USA");
5    const [isStudent, setIsStudent] = useState(true);
6
7    return <div>...</div>;
8  };
9
10 export default App;
```

```
[
    0: {...}
    1: {...}
    2: {...}
    3: {...}
]
```

useState()

- In Strict Mode, React will **call your initializer function twice** in order to help you find accidental impurities. This is development-only behavior and does not affect production. If your initializer function is pure (as it should be), this should not affect the behavior. The result from one of the calls will be ignored.

- If the new value you provide is identical to the current `state`, as determined by an `Object.is` comparison, React will **skip re-rendering the component and its children.** This is an optimization. Although in some cases React may still need to call your component before skipping the children, it shouldn't affect your code.

useState()

Updating objects and arrays in state

You can put objects and arrays into state. In React, state is considered read-only, so **you should *replace* it rather than *mutate* your existing objects**. For example, if you have a `form` object in state, don't mutate it:

```
// 🚩 Don't mutate an object in state like this:
form.firstName = 'Taylor';
```

Instead, replace the whole object by creating a new one:

```
// ✅ Replace state with a new object
setForm({
  ...form,
  firstName: 'Taylor'
});
```

Read updating objects in state and updating arrays in state to learn more.

useState()

Avoiding recreating the initial state

React saves the initial state once and ignores it on the next renders.

```
function TodoList() {
  const [todos, setTodos] = useState(createInitialTodos());
  // ...
```

Although the result of `createInitialTodos()` is only used for the initial render, you're still calling this function on every render. This can be wasteful if it's creating large arrays or performing expensive calculations.

To solve this, you may **pass it as an *initializer* function** to `useState` instead:

```
function TodoList() {
  const [todos, setTodos] = useState(createInitialTodos);
  // ...
```

Notice that you're passing `createInitialTodos`, which is the *function itself*, and not `createInitialTodos()`, which is the result of calling it. If you pass a function to `useState`, React will only call it during initialization.

React may call your initializers twice in development to verify that they are pure.

useState()

I'm trying to set state to a function, but it gets called instead

You can't put a function into state like this:

```
const [fn, setFn] = useState(someFunction);

function handleClick() {
  setFn(someOtherFunction);
}
```

Because you're passing a function, React assumes that `someFunction` is an initializer function, and that `someOtherFunction` is an updater function, so it tries to call them and store the result. To actually *store* a function, you have to put `() =>` before them in both cases. Then React will store the functions you pass.

```
const [fn, setFn] = useState(() => someFunction);

function handleClick() {
  setFn(() => someOtherFunction);
}
```

THANK YOU

QnA & Discussion

Retrospect

Please write in free format and upload.

NICKNAME.MD

in 11_15_State/Retrospect

https://github.com/gdsc-konkuk/24-25-study-react-nextjs-docs/tree/main/11_15_State