State Management

React & Next.js Official Docs Study

DongMin Kim

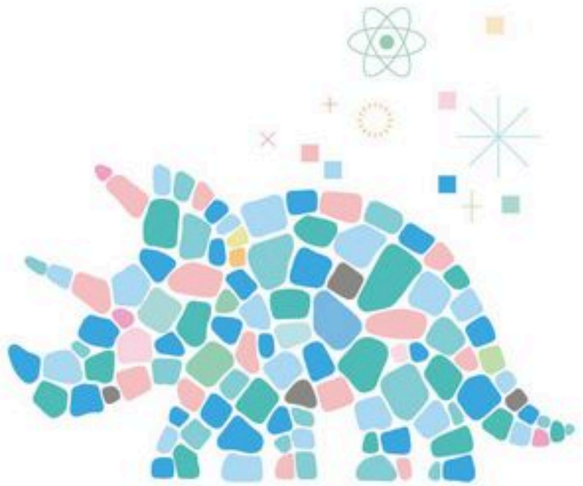
Contents

* What is state?

* How to implement Global State

오픈소스 & 웹 시리즈 119

예제 코드 제공



리액트 훅을 활용한

마이크로 상태 관리

리액트 상태 관리의 기본 개념부터
동작 원리와 문제 해결, 렌더링 최적화 기법까지

다이시 카토 지음 / 이선협, 김지은 옮김

위키북스

Daishi Kato
dai-shi

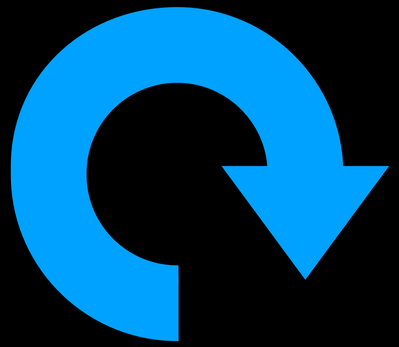
Follow ♡ Sponsor

React library author, maintaining three state management libraries, Zustand🐻, Jotai👻, Valtio🧙, and React framework, Waku⛩.

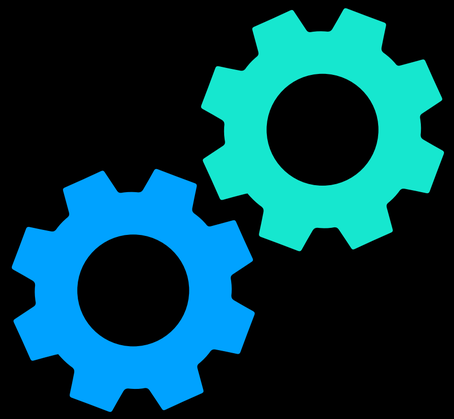
7.1k followers · **1** following

State

Data that affects the behavior of components and the UI



Trigger rerendering



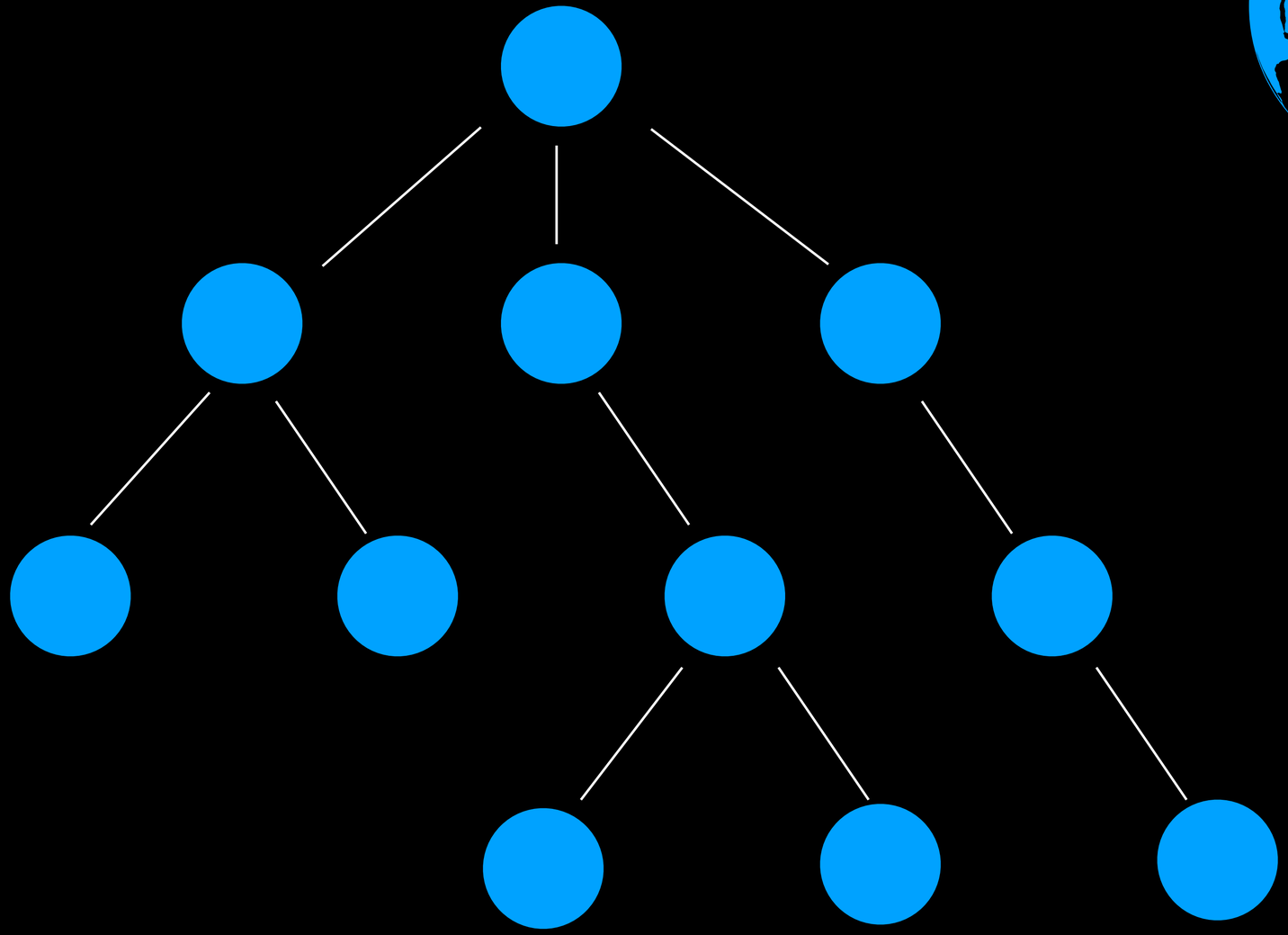
Isolated

Immutability

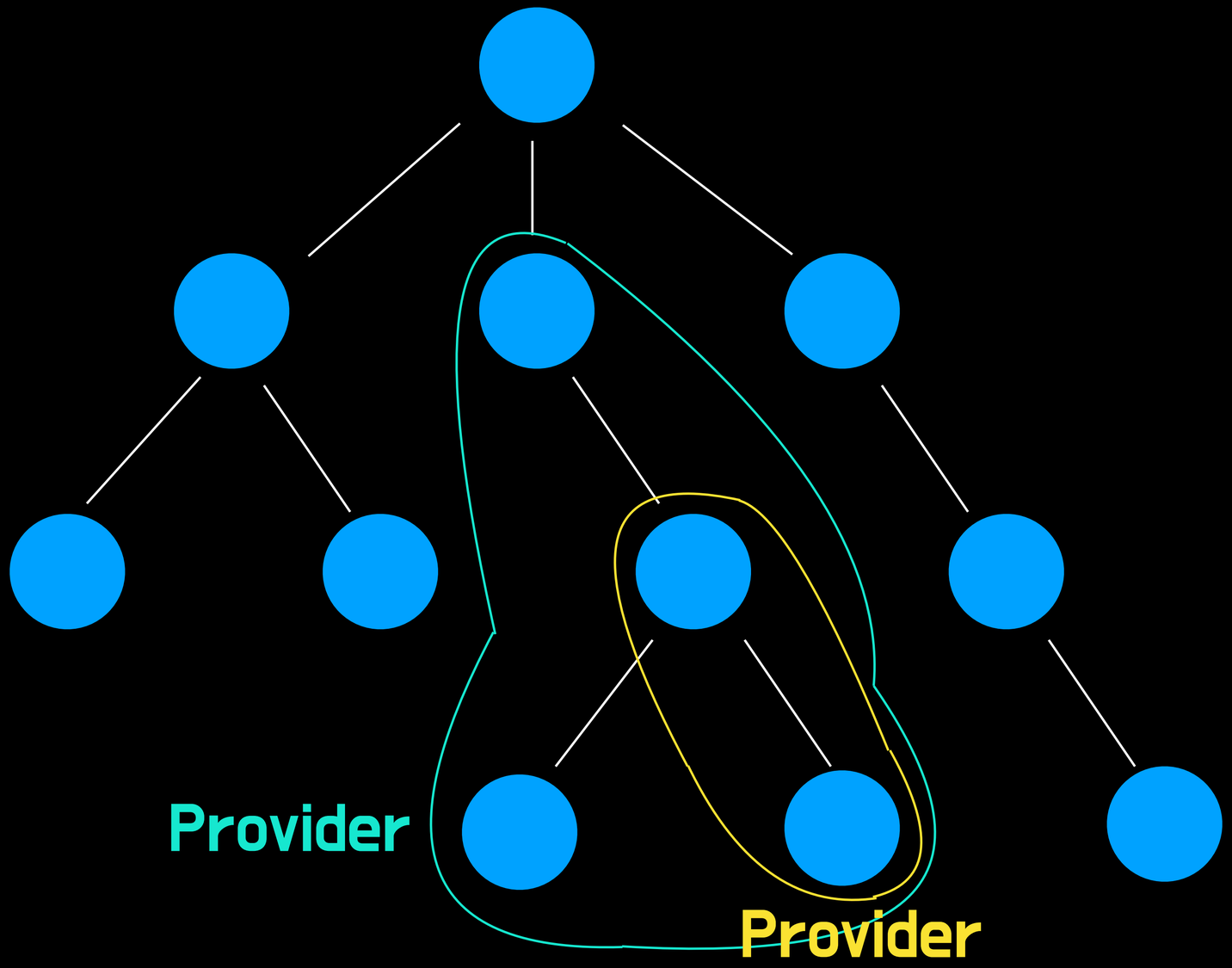
State

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

export function Component() {
  const [state, setState] = useState("");

  return (
    <div>
      <h1>{state}</h1>
      <button onClick={() => setState("Hello")}>Set state</button>
    </div>
  );
}

```

State



Context



Context

```
1  const CountContext = createContext({ count1: 0, count2: 0 });
2
3  const Counter1 = () => {
4    const { count1 } = useContext(CountContext);
5    ...
6  }
7
8  const Counter2 = () => {
9    const { count2 } = useContext(CountContext);
10   ...
11 }
12
13 const App = () => {
14   const [count1, setCount1] = useState(0);
15   const [count2, setCount2] = useState(0);
16
17   return (
18     <CountContext.Provider value={{ count1, count2 }}>
19       ...
```

Context

```
const Count1Context = createContext([0, () => {}]);

const Count2Context = createContext([0, () => {}]);

const Count1Provider = ({ children }) => {
  const [count1, setCount1] = useState(0);
  return (
    <Count1Context.Provider value={[count1, setCount1]}>
      {children}
    </Count1Context.Provider>
  );
};

const Count2Provider = ({ children }) => {
  const [count2, setCount2] = useState(0);
  return (
    <Count2Context.Provider value={[count2, setCount2]}>
      {children}
    </Count2Context.Provider>
  );
};

const App = () => (
    <Count1Provider>
      <Count2Provider>
    ...
```

```
const Count1 = () => {
  const [count1, setCount1] = useContext(Count1Context);
  ...

const Count2 = () => {
  const [count2, setCount2] = useContext(Count2Context);
  ...

```

Context

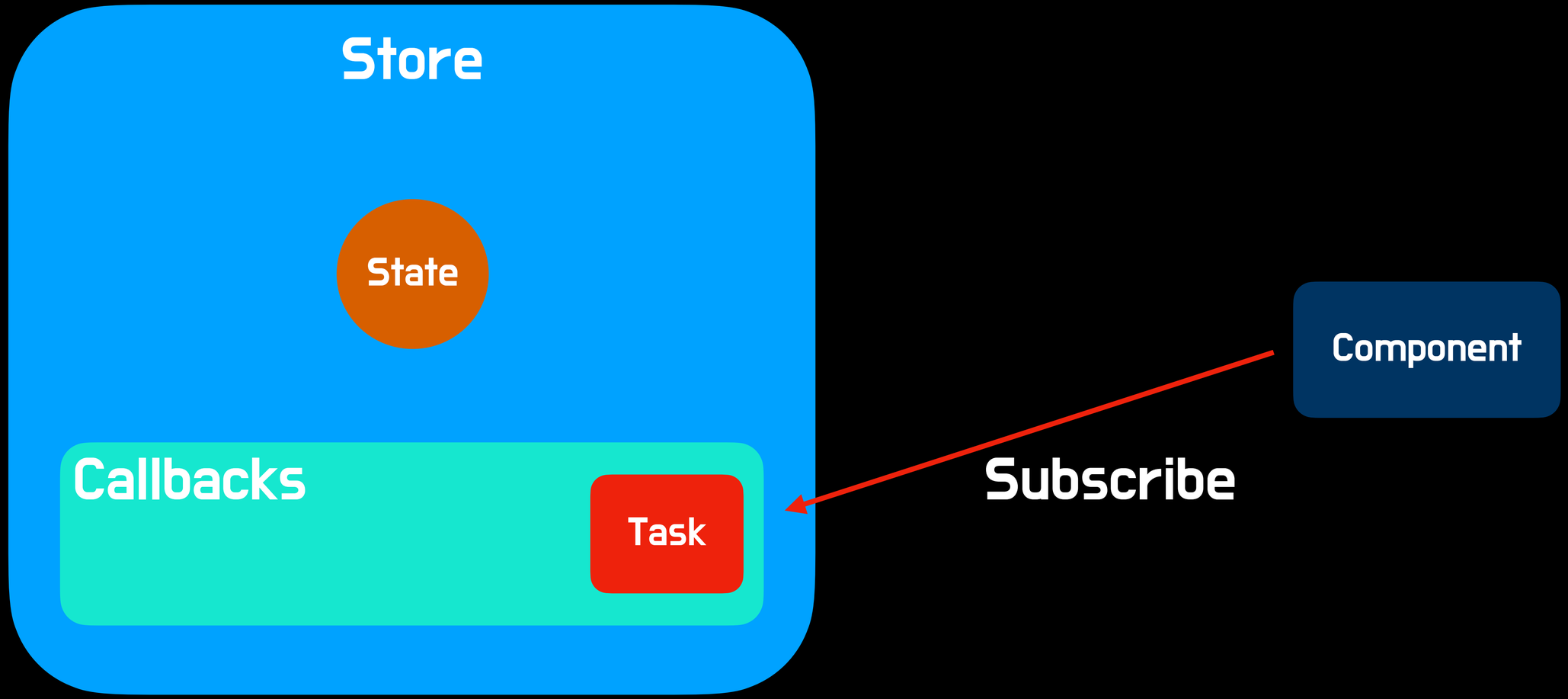
```jsx
const Count1Context = createContext(0);
const Count2Context = createContext(0);
const DispatchContext = createContext(() => {});

const Provider = ({ children }) => {
  const [state, dispatch] = useReducer(
    ( prev, action ) => {
      if (action.type === "INC1") {
        return { ...prev, count1: prev.count1 + 1 };
      }
      if (action.type === "INC2") {
        return { ...prev, count2: prev.count2 + 1 };
      }
      throw new Error("no matching action");
    },
    {
      count1: 0,
      count2: 0,
    }
  );

  return (
    <DispatcherContext.Provider value={dispatch}>
      <Count1Context.Provider value={state.count1}>
        <Count2Context.Provider value={state.count2}>
          ...
```

```jsx
const Counter1 = () => {
  const count1 = useContext(Count1Context);
  const dispatch = useContext(DispatchContext);
  ...

const Counter2 = () => {
  const count2 = useContext(Count2Context);
  const dispatch = useContext(DispatchContext);
  ...
```

Context

```
const Count1Context = createContext(null);

const Count1Provider = ({ children }) => {
  <Count1Context.Provider value={useState(0)}>
    {children}
  </Count1Context.Provider>
};

const useCount1 = () => {
  const value = useContext(Count1Context);
  if (value === null) throw new Error("Provider missing");
  return value;
};

const Count1 = () => {
  const [count1, setCount1] = useCount1();
  ...
}

const App = () => (
  <Count1Provider>
  ...
```

Subscribe



Subscribe

```ts
const createStore = <T extends unknown>(initialState: T): Store<T> => {
  let state = initialState;
  const callbacks = new Set<() => void>();
  const getState = () => state;
  const setState = (nextState: T | ((prev: T) => T)) => {
    state =
      typeof nextState === "function"
        ? (nextState as (prev: T) => T)(state)
        : nextState;
    callbacks.forEach((callback) => callback());
  };
  const subscribe = (callback: () => void) => {
    callbacks.add(callback);
    return () => callbacks.delete(callback);
  };
  return { getState, setState, subscribe };
};

const store = createStore({ count: 0 });
```

Subscribe

```
1  const useStore = (store) => {
2    const [state, setState] = useState(store.getState());
3
4    useEffect(() => {
5      const unsubscribe = store.subscribe(() => {
6        setState(store.getState());
7      });
8      setState(store.getState());
9      return unsubscribe;
10   }, [store]);
11
12   return [state, store.setState];
13 };
```

Subscribe

```
const Component1 = () => {
  const [state, setState] = useStore(store);
  const inc = () => {
    setState((prev) => ({ ...prev, count: prev.count + 1 }));
  };

  return (
    <div>
      <h1>Component1</h1>
      <p>Count: {state.count}</p>
      <button onClick={inc}>Increment</button>
    </div>
  );
};

function App() {
  return (
    <>
      <Component1 />
      ...
```

Subscribe

```tsx
const useStoreSelector = <T, S>(
  store: Store<T>
  selector: (state: T) => S
) => {
  const [state, setState] = useState(() => selector(store.getState()));

  useEffect(() => {
    const unsubscribe = store.subscribe(() => {
      setState(selector(store.getState()));
    });
    setState(selector(store.getState()));
    return unsubscribe;
  }, [store, selector]);

  return state;
}
```

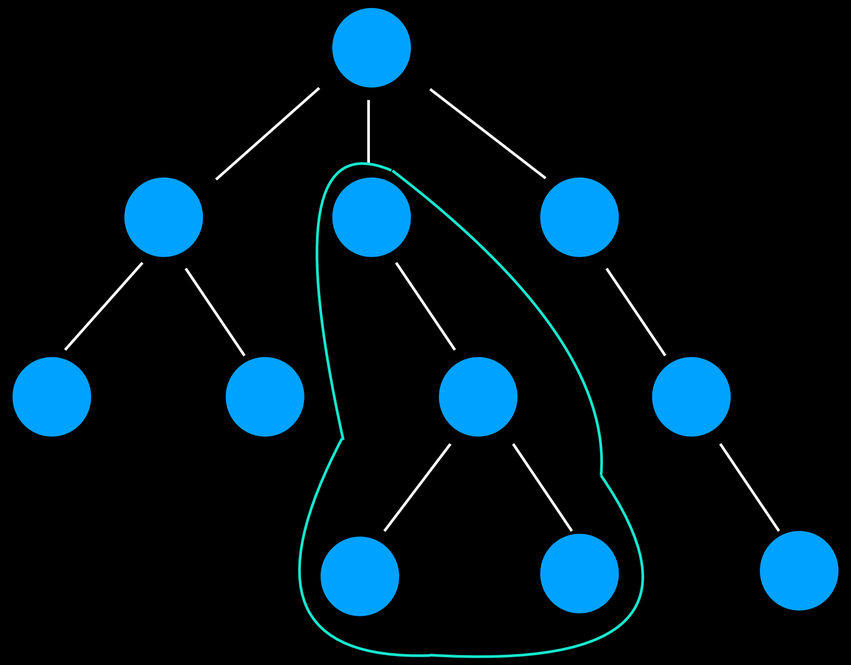
Subscribe

```
1  const Component1 = () => {
2    const state = useStoreSelector(
3      store,
4      useCallback((state) => state.count1, []),
5    );
6
7    const inc = () => {
8      store.setState((prev) => ({
9        ...prev,
10       count1: prev.count1 + 1,
11     }));
12 ...
```

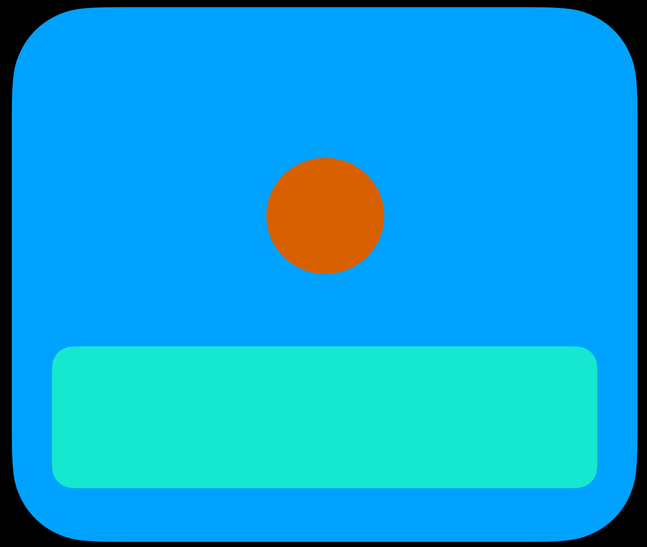
Subscribe

```
1  const Component1 = () => {
2    const state = useSyncExternalStore(
3      store.subscribe,
4      useCallback(() => store.getState().count, [])
5    );
6
7    const inc = () => {
8      store.setState((prev) => ({ ...prev, count: prev.count + 1 }));
9    };
10
11   return (
12     <div>
13       <h1>Component1</h1>
14       <p>Count: {state}</p>
15       <button onClick={inc}>Increment</button>
16     </div>
17   );
18 };
```

Context + Subscribe

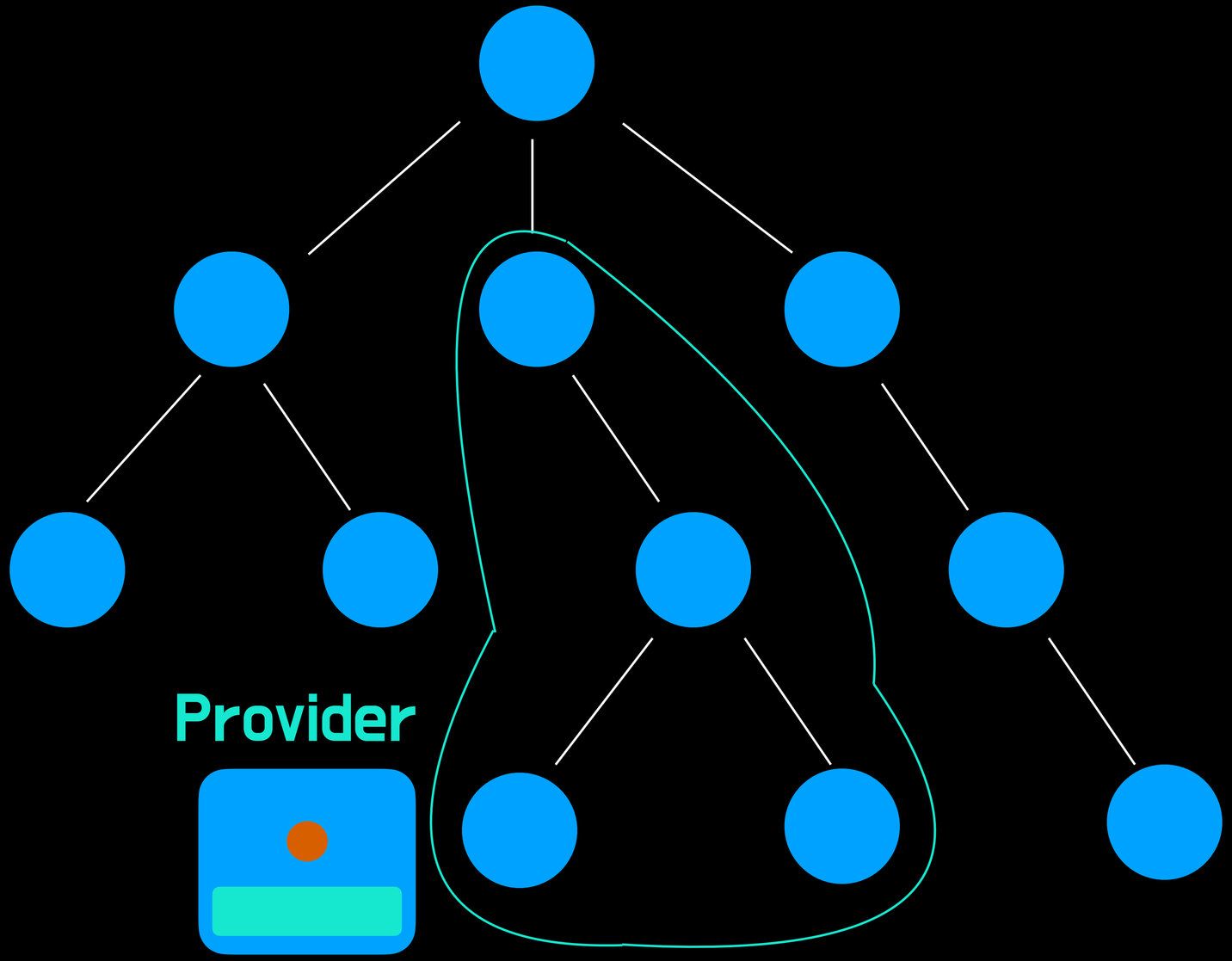


Context



Subscribe(Module)

Context + Subscribe



Context + Subscribe

```tsx
type State = { count: number; text?: string };

const StoreContext = createContext<Store<State>>(
  createStore<State>({ count: 0, text: "hello" })
);

const StoreProvider = ({
  initialState,
  children,
}: {
  initialState: State;
  children: ReactNode;
}) => {
  const storeRef = useRef<Store<State>>();
  if (!storeRef.current) {
    storeRef.current = createStore(initialState);
  }

  return (
    <StoreContext.Provider value={storeRef.current}>
      {children}
    </StoreContext.Provider>
  );
};
```

Context + Subscribe

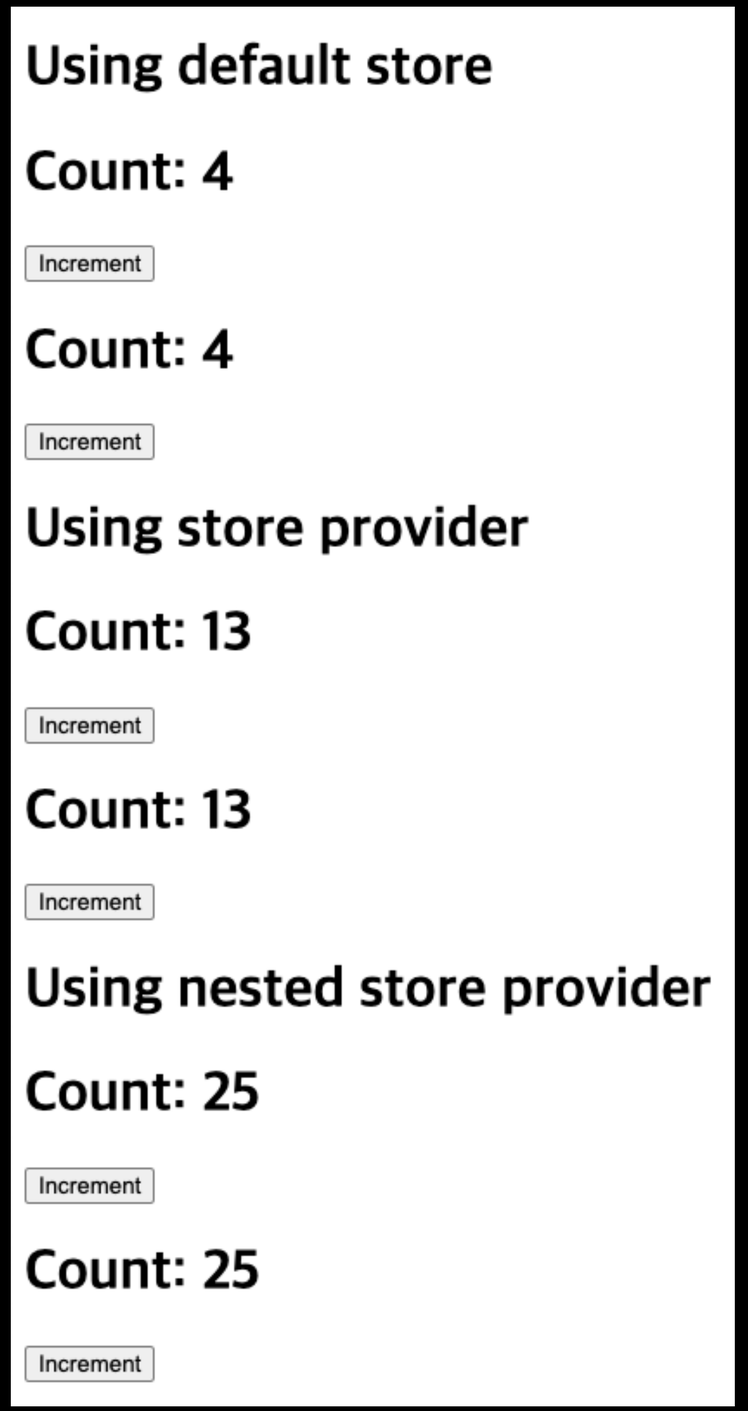
```ts
const useSelector = <S extends unknown>(selector: (state: State) => S) => {
  const store = useContext(StoreContext);

  return useSyncExternalStore(
    store.subscribe,
    useCallback(() => selector(store.getState()), [selector, store])
  );
};
```

```ts
const useSetState = () => {
  const store = useContext(StoreContext);
  return store.setState;
}
```

```ts
const selectCount = (state: State) => state.count;

const Component = () => {
  const count = useSelector(selectCount);
  const setState = useSetState();
  const inc = () => {
    setState((prev) => ({ ...prev, count: prev.count + 1 }));
  };

  return (
    <div>
      <h1>Count: {count}</h1>
      <button onClick={inc}>Increment</button>
    </div>
  );
};
```

Context + Subscribe

```tsx
function App() {
  return (
    <>
      <h1>Using default store</h1>
      <Component />
      <Component />
      <StoreProvider initialState={{ count: 10 }}>
        <h1>Using store provider</h1>
        <Component />
        <Component />
        <StoreProvider initialState={{ count: 20 }}>
          <h1>Using nested store provider</h1>
          <Component />
          <Component />
        </StoreProvider>
      </StoreProvider>
    </>
  );
}
```



Zustand

```jsx
import { create } from 'zustand'

const useStore = create((set) => ({
  bears: 0,
  increasePopulation: () => set((state) => ({ bears: state.bears + 1 })),
  removeAllBears: () => set({ bears: 0 }),
  updateBears: (newBears) => set({ bears: newBears }),
}))

function BearCounter() {
  const bears = useStore((state) => state.bears)
  return <h1>{bears} around here...</h1>
}

function Controls() {
  const increasePopulation = useStore((state) => state.increasePopulation)
  return <button onClick={increasePopulation}>one up</button>
}
```

Review

Thank you

QnA & Discussion

Retrospect

NICKNAME.MD

in 01_06_StateManagement/Retrospect