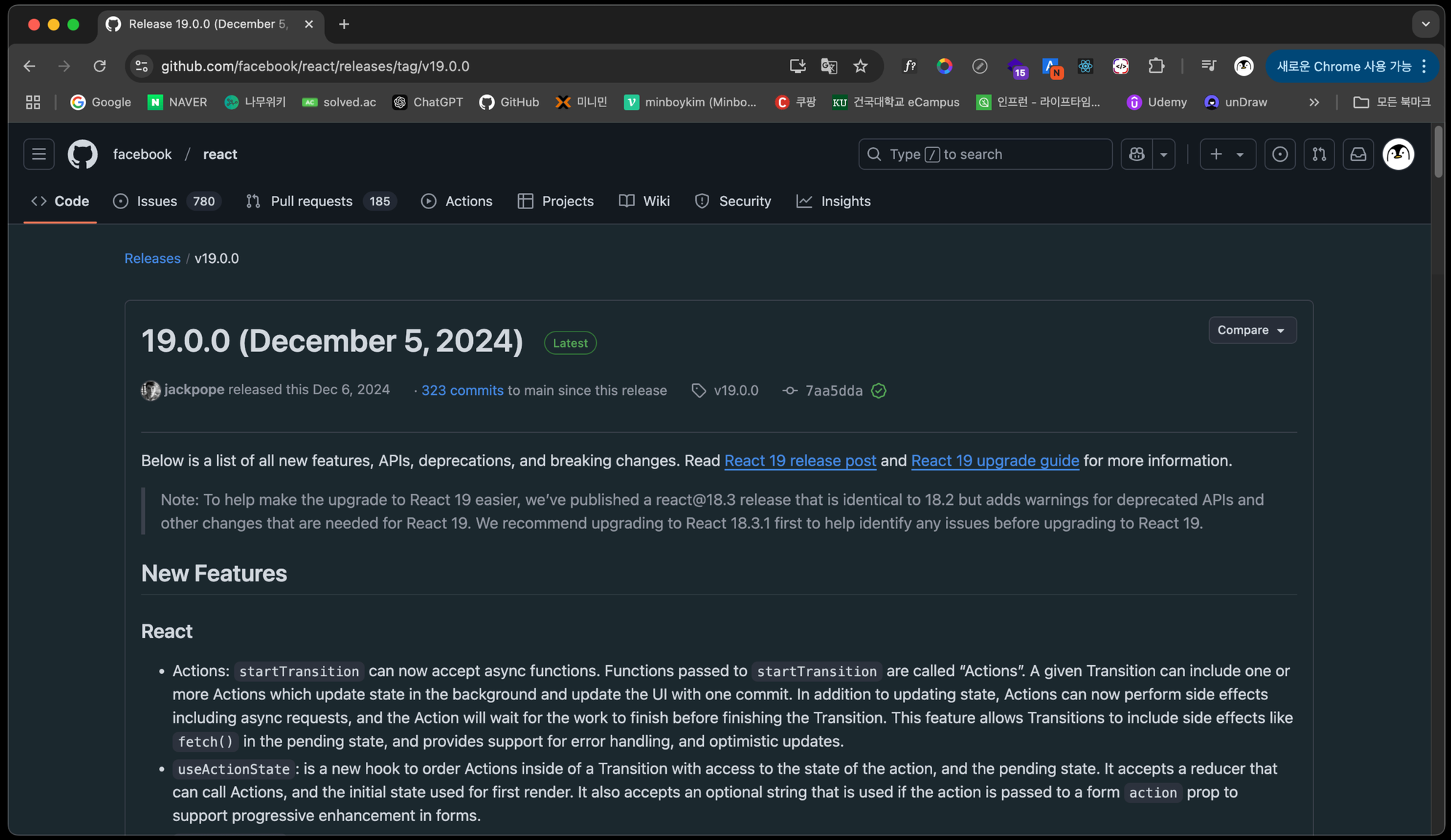
React 19

React & Next.js Official Docs Study

DongMin Kim

Contents

React 19


Release 19.0.0 (December 5,
github.com/facebook/react/releases/tag/v19.0.0
새로운 Chrome 사용 가능
Google
NAVER
나무위키
solved.ac
ChatGPT
GitHub
미니민
minboykim (Minbo...
쿠팡
건국대학교 eCampus
인프런 - 라이프타임...
Udemy
unDraw
모든 북마크
facebook / react
Type / to search
Code
Issues 780
Pull requests 185
Actions
Projects
Wiki
Security
Insights
Releases / v19.0.0
19.0.0 (December 5, 2024)
Latest
Compare
jackpope released this Dec 6, 2024
· 323 commits to main since this release
v19.0.0
7aa5dda
Below is a list of all new features, APIs, deprecations, and breaking changes. Read React 19 release post and React 19 upgrade guide for more information.
Note: To help make the upgrade to React 19 easier, we've published a react@18.3 release that is identical to 18.2 but adds warnings for deprecated APIs and other changes that are needed for React 19. We recommend upgrading to React 18.3.1 first to help identify any issues before upgrading to React 19.
New Features
React
Actions: startTransition can now accept async functions. Functions passed to startTransition are called "Actions". A given Transition can include one or more Actions which update state in the background and update the UI with one commit. In addition to updating state, Actions can now perform side effects including async requests, and the Action will wait for the work to finish before finishing the Transition. This feature allows Transitions to include side effects like fetch() in the pending state, and provides support for error handling, and optimistic updates.
useActionState : is a new hook to order Actions inside of a Transition with access to the state of the action, and the pending state. It accepts a reducer that can call Actions, and the initial state used for first render. It also accepts an optional string that is used if the action is passed to a form action prop to support progressive enhancement in forms.

React 19

```
~
❯ npm create vite@latest reat19
```

package.json

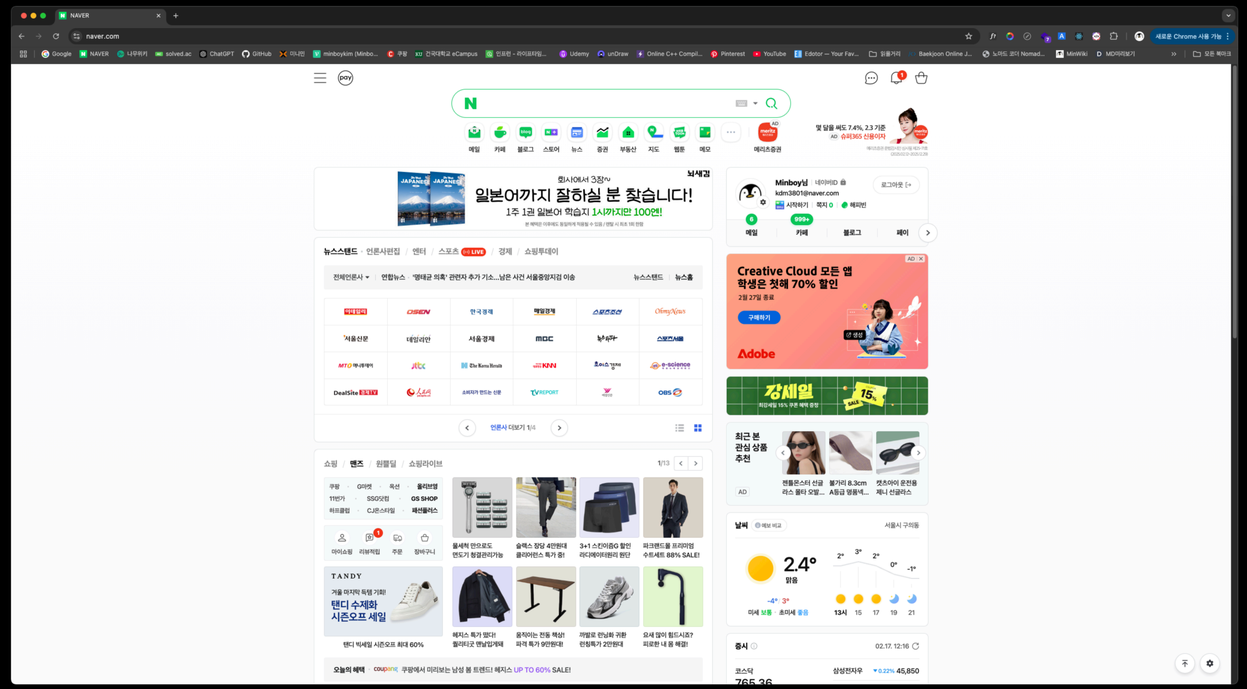
```
{
  "name": "react19",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  Debug
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^19.0.0",
    "react-dom": "^19.0.0"
  },
  "devDependencies": {
    "@eslint/js": "^9.19.0",
    "@types/react": "^19.0.8",
    "@types/react-dom": "^19.0.3",
    "@vitejs/plugin-react": "^4.3.4",
    "eslint": "^9.19.0",
    "eslint-plugin-react-hooks": "^5.0.0",
    "eslint-plugin-react-refresh": "^0.4.18",
    "globals": "^15.14.0",
    "typescript": "~5.7.2",
    "typescript-eslint": "^8.22.0",
    "vite": "^6.1.0"
  }
}
```

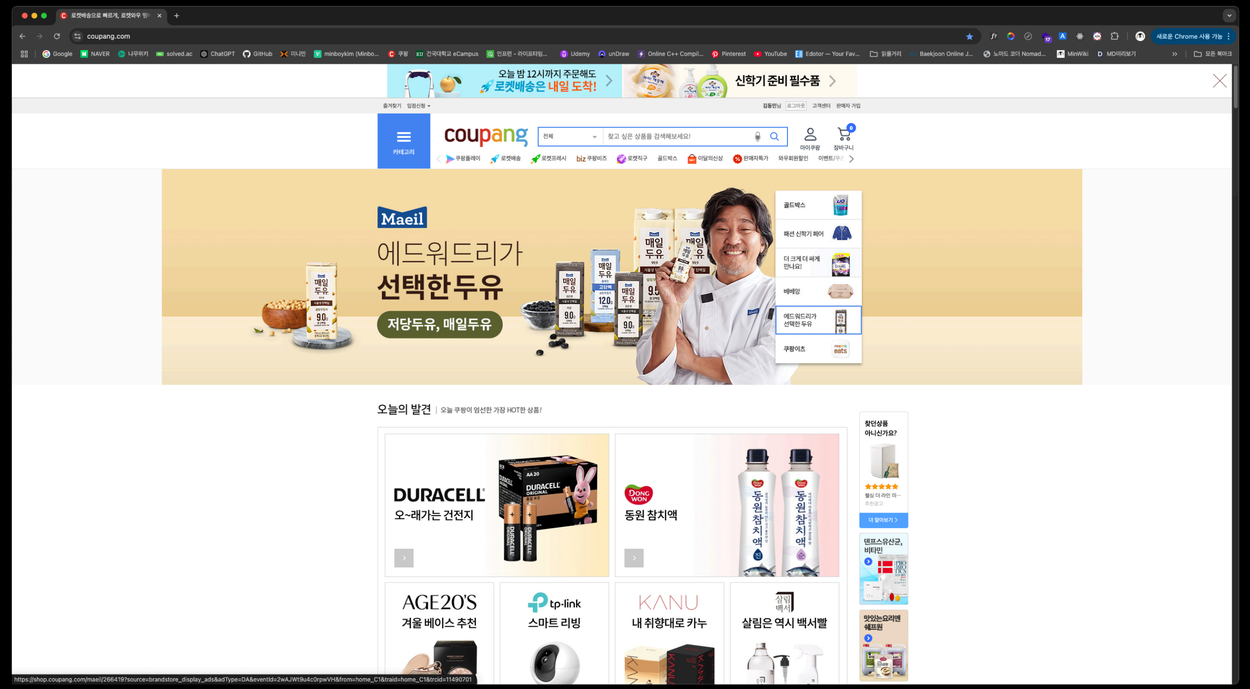
Before

```tsx
function UpdateName() {
  const [name, setName] = useState("");
  const [error, setError] = useState<string | null>(null);
  const [isPending, setIsPending] = useState(false);

  const handleSubmit = async () => {
    setError(null);
    setIsPending(true);
    const error = await updateName(name);
    setIsPending(false);
    if (error) {
      setError(error);
      return;
    }
    alert("Name updated successfully");
  };

  return (
    <div>
      <input value={name} onChange={(event) => setName(event.target.value)} />
      <button onClick={handleSubmit} disabled={isPending}>
        {isPending ? "Saving..." : "Save"}
      </button>
      {error && <p style={{ color: "red" }}>{error}</p>}
    </div>
  );
}

function App() {
  return <UpdateName />;
}
```

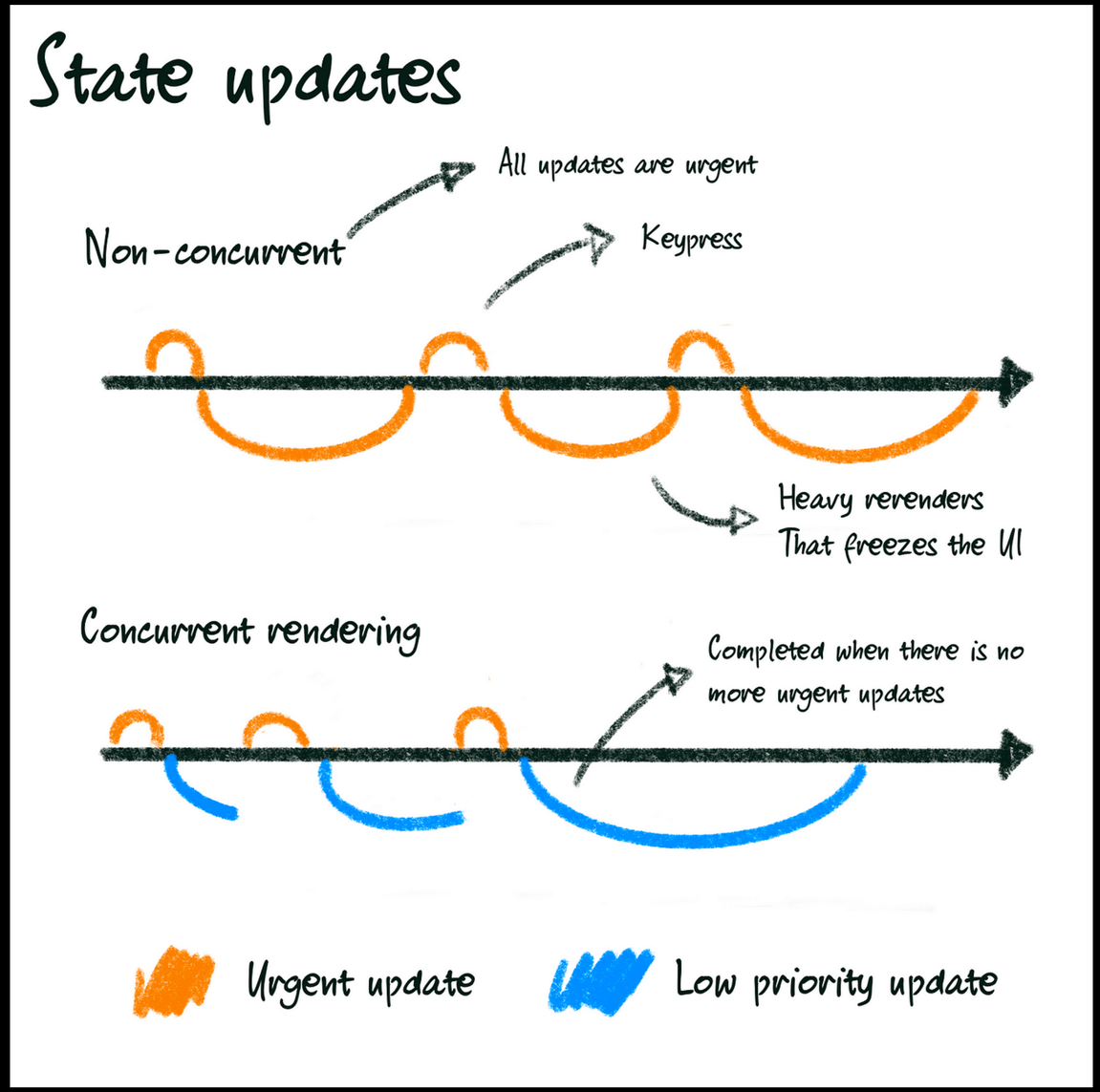
```tsx
async function updateName(name: string): Promise<string | null> {
  if (name.length === 0) {
    return "Name cannot be empty";
  }

  try {
    await new Promise((resolve, reject) => setTimeout(reject, 1000));
  } catch {
    return "Something went wrong";
  }

  return null;
}
```

useTransition





useTransition



useTransition

```tsx
function UpdateName() {
  const [name, setName] = useState("");
  const [error, setError] = useState<string | null>(null);
  const [isPending, startTransition] = useTransition();

  const handleSubmit = async () => {
    setError(null);
    startTransition(async () => {
      const error = await updateName(name);
      if (error) {
        setError(error);
        return;
      }
      alert("Name updated successfully");
    });
  };

  return (
    <div>
      <input value={name} onChange={(event) => setName(event.target.value)} />
      <button onClick={handleSubmit} disabled={isPending}>
        {isPending ? "Saving..." : "Save"}
      </button>
      {error && <p style={{ color: "red" }}>{error}</p>}
    </div>
  );
}
```

useActionState

```
const [state, formAction, isPending] = useActionState(fn, initialState, permalink?);
```

```tsx
type SubmitActionState = string | null;

function UpdateName() {
  const [error, submitAction, isPending] = useActionState(
    async (_: SubmitActionState, formData: FormData) => {
      const error = await updateName(formData.get("name") as string);
      if (error) {
        return error;
      }
      alert("Name updated successfully");
      return null;
    },
    null
  );

  return (
    <form action={submitAction}>
      <input name="name" />
      <button type="submit" disabled={isPending}>
        {isPending ? "Saving..." : "Save"}
      </button>
      {error && <p style={{ color: "red" }}>{error}</p>}
    </form>
  );
}
```

useFormStatus

```
const { pending, data, method, action } = useFormStatus();
```

```
function DesignButton() {
  const { pending } = useFormStatus();
  return <button disabled={pending}>{pending ? "Saving..." : "Save"}</button>;
}

type SubmitActionState = string | null;

function UpdateName() {
  const [error, submitAction] = useActionState(
    async (_: SubmitActionState, formData: FormData) => {
      const error = await updateName(formData.get("name") as string);
      if (error) {
        return error;
      }
      alert("Name updated successfully");
      return null;
    },
    null
  );

  return (
    <form action={submitAction}>
      <input name="name" />
      <DesignButton />
      {error && <p style={{ color: "red" }}>{error}</p>}
    </form>
  );
}
```

useOptimistic

```
const [optimisticState, addOptimistic] = useOptimistic(state, updateFn);
```

use

use

`use` is a React API that lets you read the value of a resource like a Promise or context.

```
const value = use(resource);
```

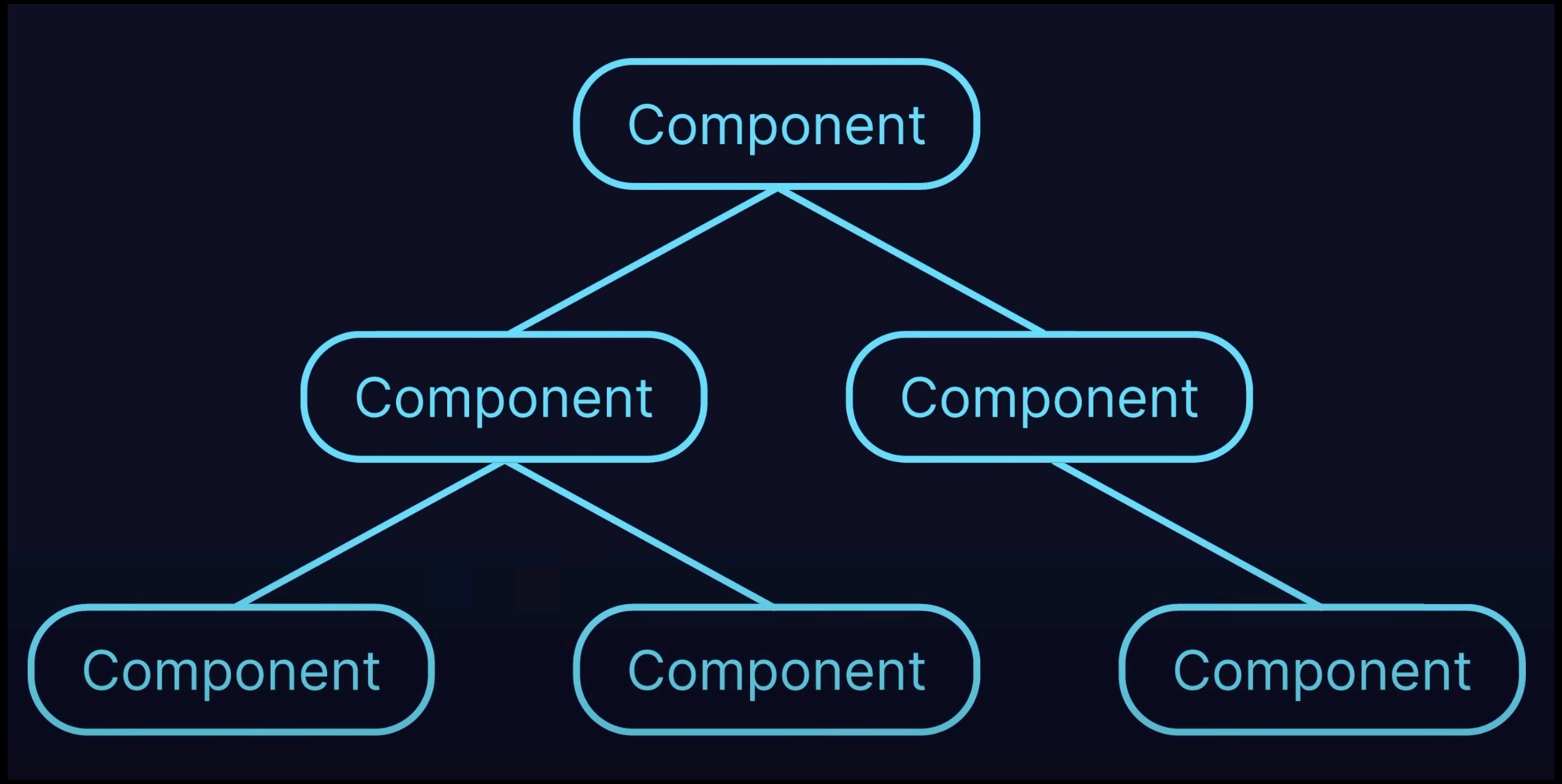
use

```jsx
function MyPage() {
  return (
    <ThemeContext.Provider value="dark">
      <Form />
    </ThemeContext.Provider>
  );
}

function Form() {
  // ... renders buttons inside ...
}
```

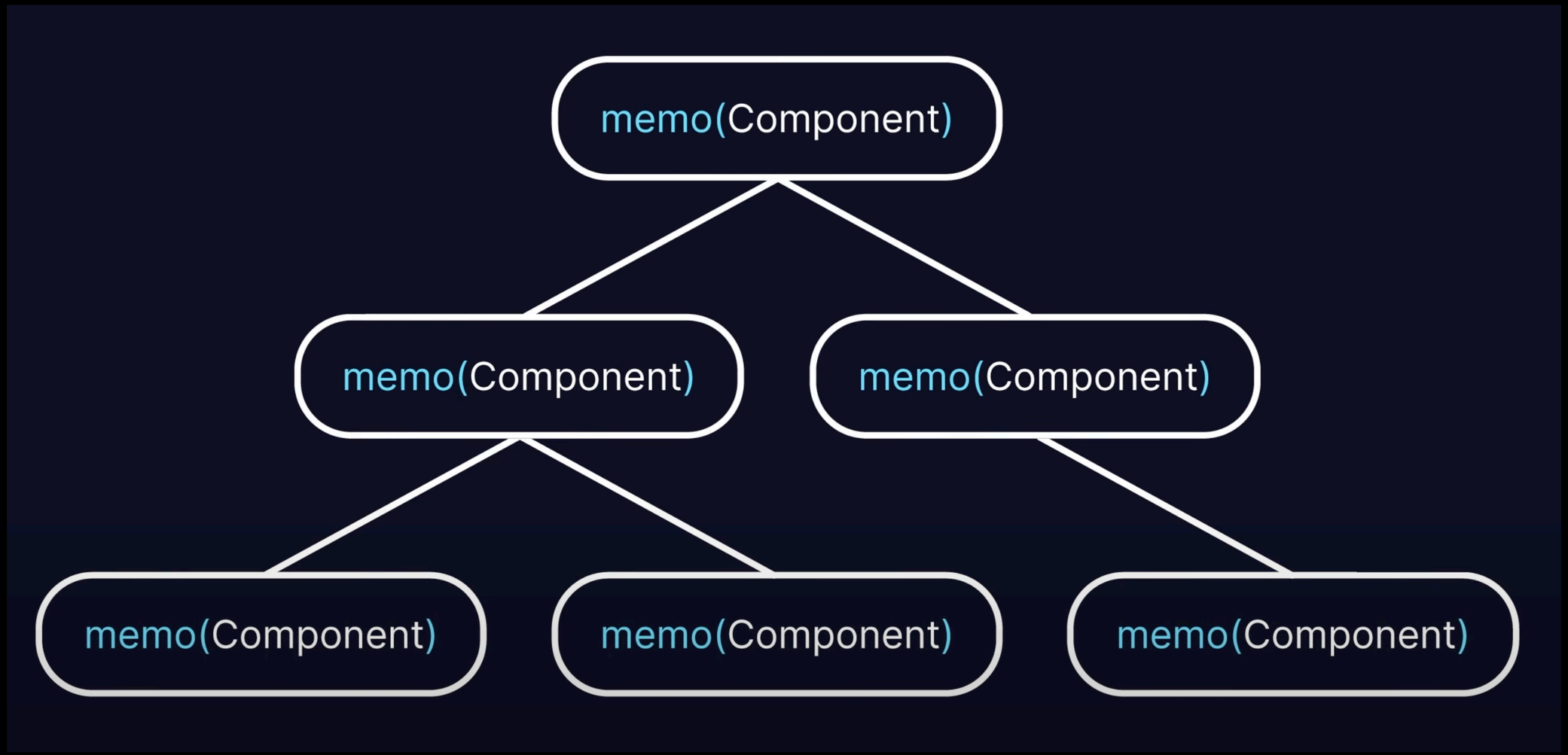
```jsx
import { use } from 'react';

function Button() {
  const theme = use(ThemeContext);
  // ...
```

```jsx
function HorizontalRule({ show }) {
  if (show) {
    const theme = use(ThemeContext);
    return <hr className={theme} />;
  }
  return false;
}
```

React Compiler



React Compiler



React Compiler

useCallback

`useCallback` is a React Hook that lets you cache a function definition between re-renders.

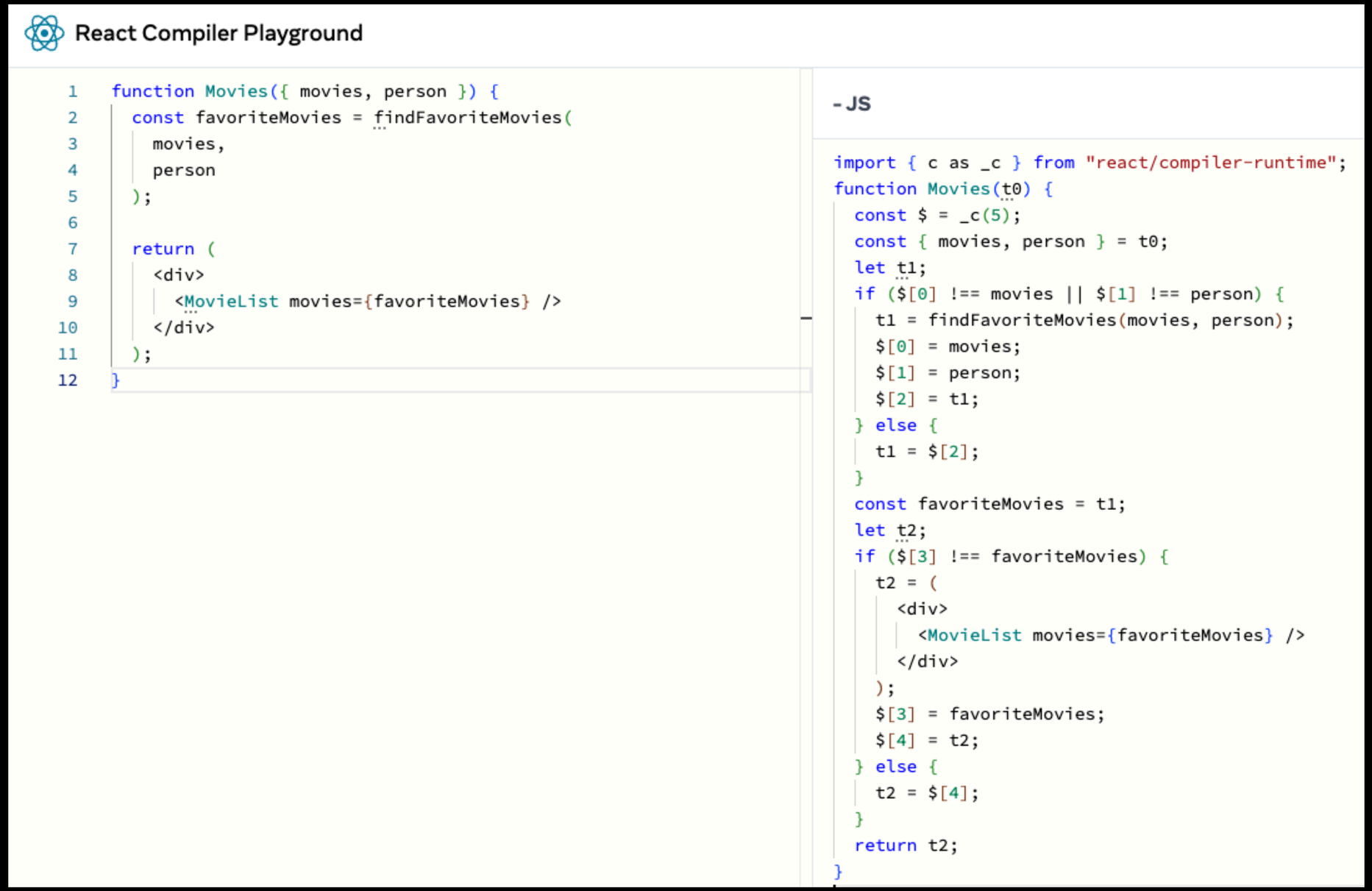
```
const cachedFn = useCallback(fn, dependencies)
```

useMemo

`useMemo` is a React Hook that lets you cache the result of a calculation between re-renders.

```
const cachedValue = useMemo(calculateValue, dependencies)
```

React Compiler



Thank you

Reference

https://react.dev/

https://youtu.be/Z-a4B7EIDXw?si=dWKNHsITQq6v6ZkJ

QnA & Discussion

Retrospect

NICKNAME.MD

in 02_17_React19/Retrospect