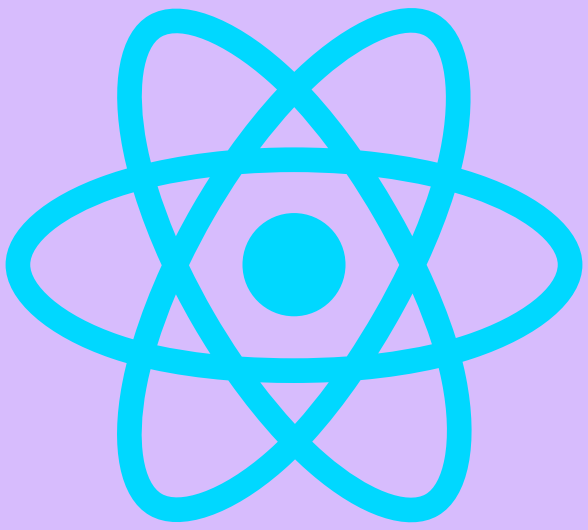
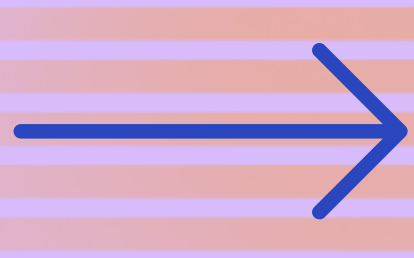
React 19: New Hooks And New Patterns

A New Era of Simplicity



Vladyslav Demirov
@vladyslav-demirov



What's New?

React 19 updates the hooks and introduces new patterns:

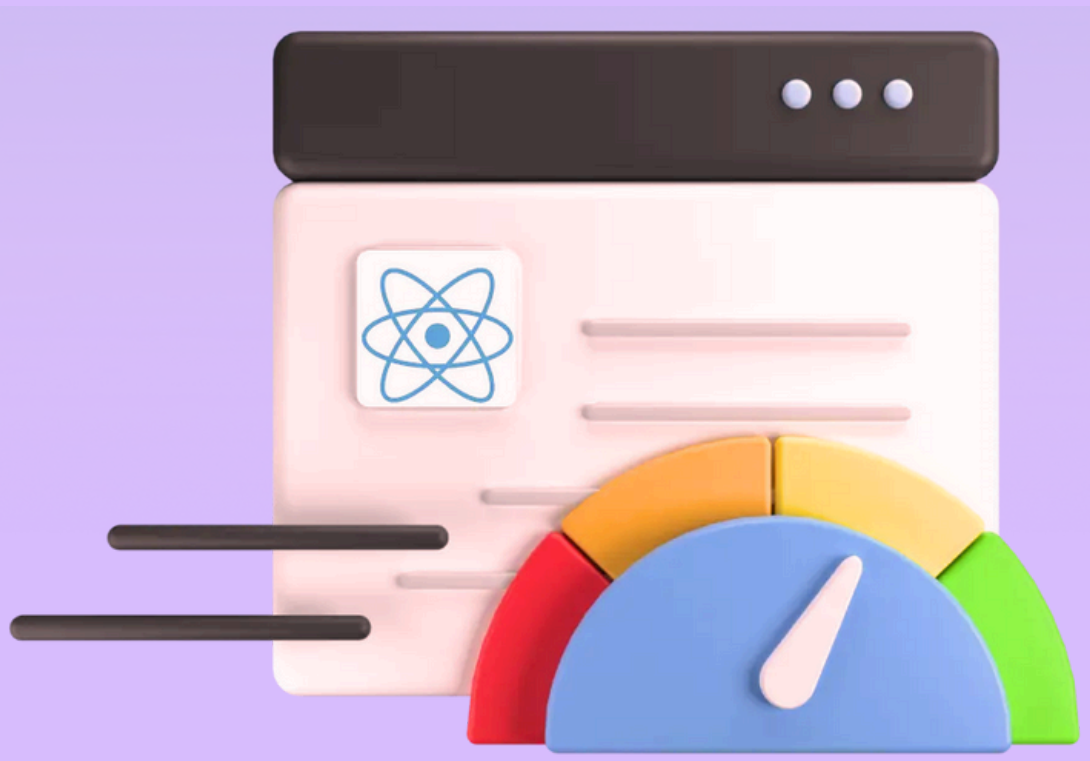
- **Memoization Simplified**: No need to explicitly call useMemo().

- **Simpler Ref Management**: Directly pass ref as a prop.

- **Introducing use**: A cleaner way to manage async data and context.

- **Enhanced Forms**: Hooks like useFormStatus and useFormState simplify handling form submissions.

Let's **explore** each **feature** with examples.



Vladyslav Demirov

No More Explicit useMemo()

Before React 19: You manually wrap calculations with useMemo() to optimize performance.
After React 19: React's Compiler handles memoization automatically.

Before React 19:

```tsx
import React, { useMemo, useState } from 'react';

function ExampleComponent() {
  const [value, setValue] = useState('');
  const isEmpty = useMemo(() => value.trim() === '', [value]);

  return <p>{isEmpty ? 'Empty' : 'Not Empty'}</p>;
}
```

After React 19:

```tsx
import React, { useState } from 'react';

function ExampleComponent() {
  const [value, setValue] = useState('');
  const isEmpty = () => value.trim() === '';

  return <p>{isEmpty() ? 'Empty' : 'Not Empty'}</p>;
}
```

Vladyslav Demirov

Simplified Ref Management

React 19 simplifies forwardRef by allowing direct ref as a prop.

Before React 19:

```tsx
const MyButton = React.forwardRef((props, ref) => (
  <button ref={ref}>{props.children}</button>
));
```

After React 19:

```tsx
const MyButton = ({ ref, children }) => (
  <button ref={ref}>{children}</button>
);
```

Key Benefit: More intuitive and concise code.

Vladyslav Demirov

Meet the use Hook

What is it? A new hook to manage promises, async logic, and context effortlessly.

tsx

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

const fetchUsers = async () => {
  const response = await fetch('https://api.example.com/users');
  return response.json();
};

const UserList = () => {
  const users = use(fetchUsers());

  return (
    <ul>
      {users.map(user => (
        <li key={user.id}>{user.name}</li>
      ))}
    </ul>
  );
};
```

Key Benefit: Replace useEffect and useState with a single use call.

Vladyslav Demirov

Simplified Context with use

What is it? Replace useContext() with use(context).

tsx

```tsx
import { createContext, use, useState } from 'react';

const ThemeContext = createContext();

const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState('light');
  const toggleTheme = () => setTheme(prev => (prev === 'light' ? 'dark' : 'light'));

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};

const ThemedCard = () => {
  const { theme, toggleTheme } = use(ThemeContext);

  return (
    <div className={theme === 'light' ? 'light-card' : 'dark-card'}>
      <p>Current Theme: {theme}</p>
      <button onClick={toggleTheme}>Toggle Theme</button>
    </div>
  );
};
```

Key Benefit: Cleaner syntax with direct access to context values

Vladyslav Demirov

Enhanced Forms with useFormStatus

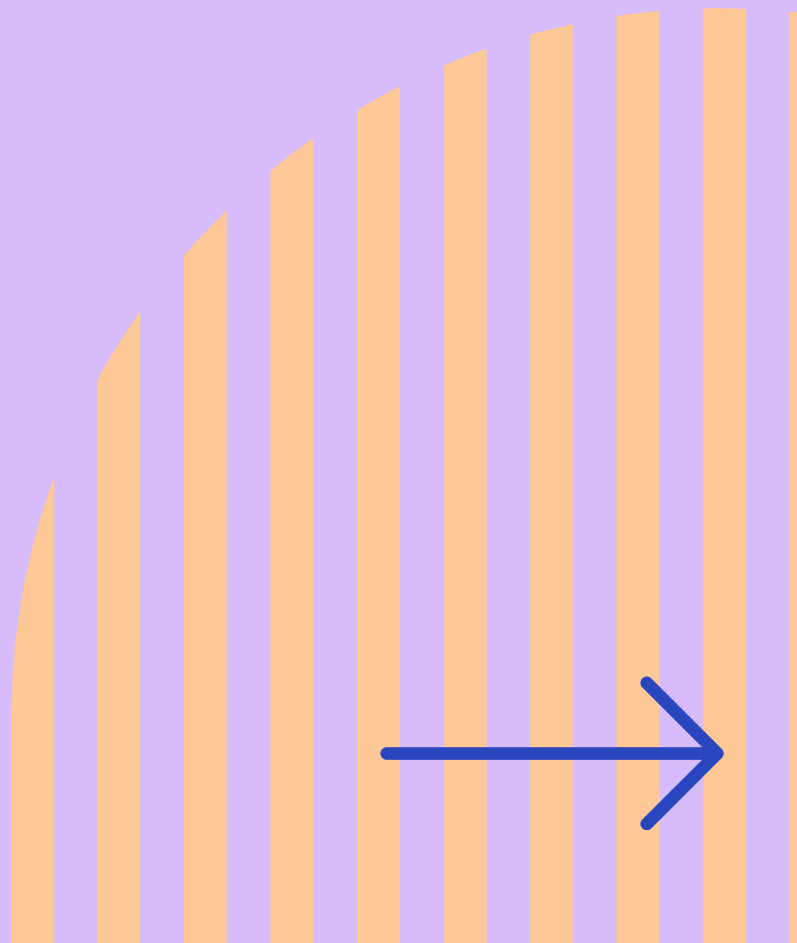
Track form submission states easily.

tsx

```tsx
import { useFormStatus } from 'react-dom';

function SubmitButton() {
  const { pending } = useFormStatus();
  return <button disabled={pending}>{pending ? 'Submitting...' : 'Submit'}</button>;
}

const Form = () => (
  <form action={async () => console.log('Form submitted')}>
    <SubmitButton />
  </form>
);
```

Key Benefit: Cleaner syntax with direct access to context values

Vladyslav Demirov



Optimistic UI with useOptimistic

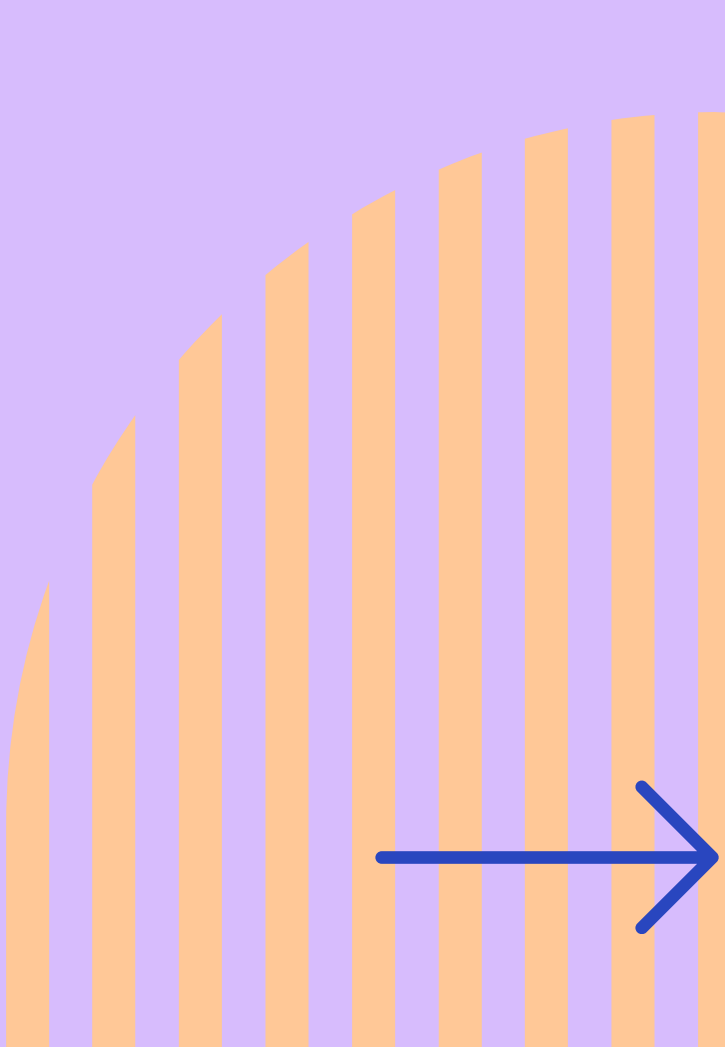
Update UI instantly while awaiting server responses.

tsx

```tsx
import { useOptimistic } from 'react';

function LikeButton() {
  const [likes, setLikes] = useOptimistic(0, (state, delta) => state + delta);

  const handleClick = () => {
    setLikes(1); // Immediate UI update
    fetch('/like').catch(() => setLikes(-1)); // Rollback on error
  };

  return <button onClick={handleClick}>Likes: {likes}</button>;
}
```

Key Benefit: Faster feedback and better user experience.

Vladyslav Demirov



Key Takeaways

- **Automatic Memoization**: Simplified with React's Compiler.

- **Refactoring Made Easy**: Direct ref support.

- **use Hook**: Streamline async logic and context.

- **Form Handling**: Enhanced with useFormStatus and useFormState.

- **Optimistic UI**: Achieved with useOptimistic.

Vladyslav Demirov

HAPPY CODING

Vladyslav Demirov
@vladyslav-demirov