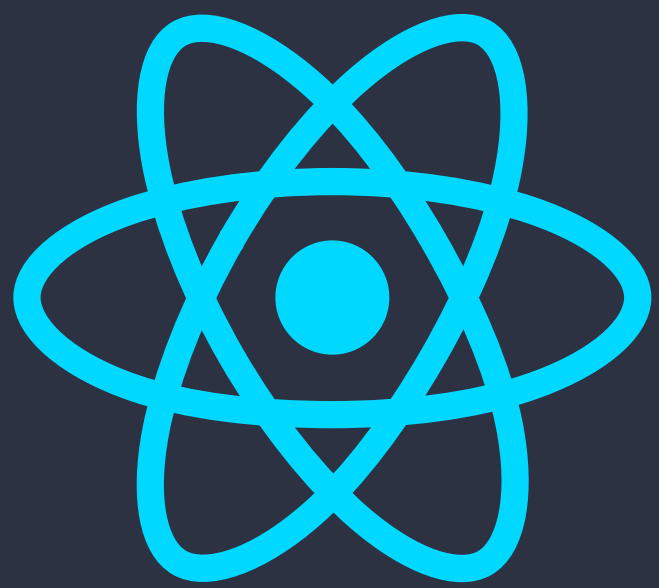
AhsanDev404

01



Introducing React 19: Elevating Your Development Experience

Welcome to React 19!

- React 19 brings significant improvements aimed at simplifying development and enhancing performance. This release
- focuses on reducing manual optimizations, allowing developers to write cleaner code.

Key Changes

- **React Compiler:** Automates performance optimizations, eliminating the need for manual memoization. **Server Components:** Enables
- rendering components on the server for improved performance and SEO. **Simplified Ref Handling:**
- Allows passing refs as props directly to function components, removing the necessity for forwardRef. **New Hooks:** Adds useFormStatus, useActionState,
- and useOptimistic to enhance form state management. **New use() Hook**: Simplifies
- asynchronous operations and context management. **Directives**: Introduces "use client" and "use server"
- directives to specify component rendering context. **Actions:** Simplify data mutations and state updates
- with automatic handling of pending states, errors, and form submissions.

The New React Compiler

- Transforms React code into optimized JavaScript, automating performance enhancements.
- Eliminates the need for manual memoization with hooks like useMemo and useCallback.

ExpensiveComponent.jsx

```jsx
// React 18
import React, { useMemo } from "react";

const ExpensiveComponent = ({ count }) => {
  const expensiveCalculation = useMemo(() => {
    let sum = 0;
    for (let i = 0; i < 1000000000; i++) {
      sum += i;
    }
    return sum;
  }, [count]);

  return <div>{expensiveCalculation}</div>;
};
```

AhsanDev404

05

The New React Compiler

Simplified version in React 19.

ExpensiveComponent.jsx

```jsx
// React 19
const ExpensiveComponent = ({ count }) => {
  let sum = 0;
  for (let i = 0; i < 1000000000; i++) {
    sum += i;
  }
  return <div>{sum}</div>;
};
```

Server Components

- Stabilized Feature: Server Components, initially experimental in React 18, are now stable and ready for production use in React 19.
- Improved Integration: Enhanced compatibility with frameworks like Next.js, facilitating smoother development workflows.
- Optimized Performance: Better server-side rendering capabilities, leading to faster load times and improved SEO.

ServerComponent.jsx

```
// Server Component
'use server';

export default async function fetchData() {
  const response = await fetch('https://api.example.com/data');
  return response.json();
}
```

🔗 Simplified Ref Handling

- In React 19, managing refs has become more intuitive:
- Direct Ref Passing: You can now pass refs directly as props to function components without using forwardRef

Form.jsx

```jsx
function ChildComponent({ innerRef }) {
  return <input ref={innerRef} />;
}

// Usage in Parent Component
function ParentComponent() {
  const inputRef = useRef(null);

  return <ChildComponent innerRef={inputRef} />;
}
```

🪝 New Hooks

- useFormStatus: Tracks form submission status, providing pending and data states.

Form.jsx

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

function SubmitButton() {
  const { pending } = useFormStatus();
  return <button disabled={pending}>Submit</button>;
}
```

- useActionState: Manages state associated with form actions, simplifying state handling.

Form.jsx

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

const [state, formAction] = useActionState(submitAction, {
  users: [],
  error: null,
});
```

🪝 New Hooks

- useOptimistic: Manages optimistic updates, providing immediate feedback during async operations.

Form.jsx

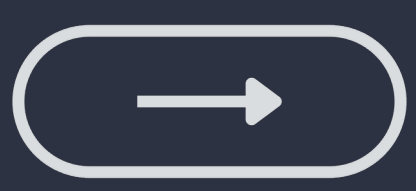
```jsx
import { useOptimistic } from "react";

function ChangeName({ currentName }) {
  const [optimisticName, setOptimisticName] = useOptimistic(currentName);

  const submitAction = async (formData) => {
    const newName = formData.get("name");
    setOptimisticName(newName);
    await updateName(newName);
  };

  return (
    <form action={submitAction}>
      <p>Your name is: {optimisticName}</p>
      <input type="text" name="name" />
    </form>
  );
}
```

🪝 Simplifying Asynchronous Operations with the use() Hook

React 19 introduces the use() hook, designed to streamline asynchronous operations and context management within components.

Key Features:

- Promise Handling: Allows reading the value of a promise directly within the component, suspending rendering until the promise resolves. Conditional
- Usage: Unlike traditional hooks, use() can be called inside conditionals and loops, offering greater flexibility.



In this example, use(commentsPromise) reads the resolved value of commentsPromise. If the promise is still pending, the component suspends, and the fallback UI specified in Suspense is displayed until the promise resolves.

Component.jsx

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

function Comments({ commentsPromise }) {
  const comments = use(commentsPromise); // Suspends until the promise resolves
  return comments.map(comment => <p key={comment.id}>{comment.text}</p>);
}

function Page({ commentsPromise }) {
  return (
    <Suspense fallback={<div>Loading comments...</div>}>
      <Comments commentsPromise={commentsPromise} />
    </Suspense>
  );
}
```

Introducing Directives

- React 19 introduces directives to specify component rendering context:
- Directive: 'use client' Marks components to be rendered on the client side. 'use server' Directive: Specifies functions to be executed on the server.

Component.jsx

```
'use client';

function ClientComponent() {
  return <div>Client Side Rendering</div>;
}
```

Component.jsx

```
'use server';

export async function serverFunction() {
  // Server-side logic
}
```

⚡ Simplified Data Mutations with Actions

React 19 introduces Actions to streamline data mutations and state updates, automatically managing pending states and errors.

Key Features:

- Automatic Pending State Management: Actions handle the pending state during asynchronous operations, providing immediate feedback to users.
- Error Handling: Built-in mechanisms capture and manage errors during data mutations, enhancing reliability.
- Optimistic Updates: Integrates with the new useOptimistic hook to provide instant UI updates while awaiting server confirmation.

AhsanDev404

14

⚡Actions: Example

Form.jsx

```jsx
import { useState, useTransition } from 'react';

function UpdateName() {
  const [name, setName] = useState('');
  const [error, setError] = useState(null);
  const [isPending, startTransition] = useTransition();

  const handleSubmit = () => {
    startTransition(async () => {
      const error = await updateName(name);
      if (error) {
        setError(error);
        return;
      }
      // Handle success
    });
  };

  return (
    <div>
      <input value={name} onChange={(e) => setName(e.target.value)} />
      <button onClick={handleSubmit} disabled={isPending}>
        Update
      </button>
      {error && <p>{error}</p>}
    </div>
  );
}
```

🚀 Upgrade to React 19 Today!

- **Installation:**

```
Terminal

npm install react@19 react-dom@19
```

- Review the official React 19 documentation for detailed guidance.
- Test new features in a development environment before deploying to production.

🎉 Embrace the Future with React 19

- React 19 simplifies development and enhances performance, empowering developers to build efficient applications.
- Explore the new features to elevate your React projects.
- Share your experiences with React 19 in the comments below!

Follow For More!

in/ahsandev404