What's new in React 19

REACT 19

React 19 was officially released on April 25, 2024 and it brings a variety of new features and improvements designed to enhance the developer experience and application performance.

React 19 introduces several new features and improvements aimed at enhancing performance, developer experience, and user interaction. Here's a brief overview of the key updates along with code snippets to illustrate their usage:

/in/mulukenm

The New ‘use’ Hook

The use hook can fetch and utilize resources like Promises or context directly within components, even inside loops and conditional statements.

It’s designed to simplify the process of fetching and consuming asynchronous data within your components. It allows you to handle resources directly in your render logic, making it easier to deal with asynchronous operations such as data fetching, waiting for data to load, and handling errors. Here is a simple example and explanation of how to use the use hook.

Check the following example to understand more about it

/in/mulukenm

How to use the ‘use’ Hook

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

// Asynchronous data fetching function
const fetchData = () => new Promise(resolve =>
    setTimeout(() => resolve('Hello, React 19!'), 1000));

const MyComponent = () => {
  // Using the `use` hook to fetch data
  const data = use(fetchData());

  return (
    <div>
      <p>Data: {data}</p>
    </div>
  );
};

export default MyComponent;
```

/in/mulukenm

The New 'useOptimistic' Hook

Handles optimistic updates, showing expected results while async operations are in progress.

It's designed to handle optimistic updates, which are updates that assume a successful result before actually confirming the result from an asynchronous operation like a server request. This can improve user experience by making the UI feel more responsive. Here's a breakdown of how it works and an example to illustrate its use.

- Optimistic State Management: It allows you to set a temporary state assuming a successful operation. If the operation fails, you can revert the state.
- User Feedback: Provides immediate feedback to users while waiting for the actual operation to complete.
- Revert on Failure: If the operation fails, you can easily revert the state to its previous value.

How to use the the ‘useOptimistic’ Hook

```jsx
import React, { useState } from 'react';
import { useOptimistic } from 'react';

// Mock API call to simulate server update
const updateServerData = (newData) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      // Simulate a successful update 80% of the time
      Math.random() > 0.2 ? resolve(newData) : reject('Update failed');
    }, 1000);
  });
};

const MyOptimisticComponent = () => {
  const [data, setData] = useState('Initial Data');
  const [optimisticData, setOptimisticData, revert] = useOptimistic(data);

  const handleUpdate = async () => {
    try {
      // Set optimistic data assuming the update will succeed
      setOptimisticData('Optimistic Update');
      // Attempt to update server data
      const updatedData = await updateServerData('Optimistic Update');
      // Confirm the update by setting the actual data
      setData(updatedData);
    } catch (error) {
      // Revert to the original data on failure
      revert();
      console.error(error);
    }
  };

  return (
    <div>
      <p>Current Data: {optimisticData}</p>
      <button onClick={handleUpdate}>Update Data</button>
    </div>
  );
};

export default MyOptimisticComponent;
```

/in/mulukenm

Document Metadata

Managing document metadata like titles and meta tags is now simpler without needing additional packages like react-helmet.

```jsx
const HomePage = () => {
  return (
    <>
      <title>Home Page</title>
      <meta name="description" content="Welcome to the Home Page" />
      {/* Page content */}
    </>
  );
}
export default HomePage;
```

/in/mulukenm

React Server Components

React 19 supports Server Components, enabling components to render on the server, improving load times and SEO.

```
"use server";

export async function getData() {
  const res = await fetch('https://jsonplaceholder.typicode.com/posts');
  return res.json();
}
```

That's not all, but it's a wrap for now.

/in/mulukenm