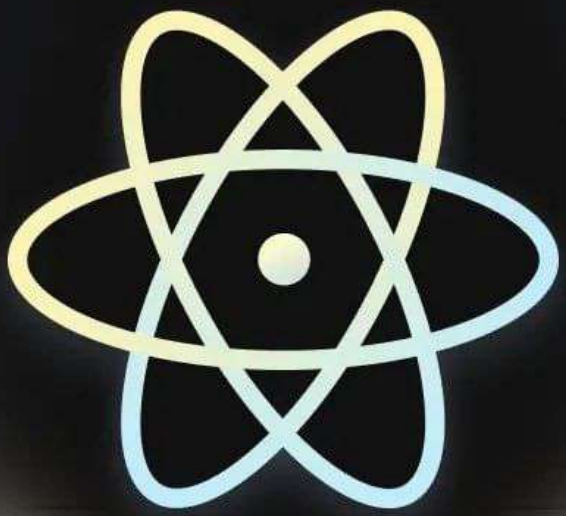
Save This !

Different Ways To Fetch Data In React

You Need To Know About



Swipe To Know

Fetch API

The Fetch API Is A Built-In Web API In Modern Browsers That Allows Making HTTP Requests. You Can Use It To Fetch Data From A Remote Server.

Example :

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => {
    // Process fetched data
  })
  .catch(error => {
    // Handle errors
  });
```

Axios

Axios Is A Popular Third-Party Library For Making HTTP Requests. It Provides A Simple API And Supports Features Like Request And Response Interception, Request Cancellation, And More.

Example :

```
import axios from 'axios';

axios.get('https://api.example.com/data')
  .then(response => {
    // Process fetched data
  })
  .catch(error => {
    // Handle errors
  });
```

Async/Await

Using Async/Await With Fetch Or Axios : You Can Use Async/Await Syntax To Make Asynchronous Code Look Synchronous, Which Can Make Your Code More Readable.

Example :

```
async function fetchData() {
  try {
    const response = await
fetch('https://api.example.com/data');
    const data = await response.json();
    // Process fetched data
  } catch (error) {
    // Handle errors
  }
}
```

React-Query Or Swr

Libraries Such As React-Query Or Swr Provide Hooks And Utilities For Managing And Caching Data Fetching In React Applications. They Offer Features Like Automatic Caching, Background Data Fetching, And Request Deduplication.

Example :

```
import { useQuery } from 'react-query';

function MyComponent() {
  const { data, error, isLoading } = useQuery('myData', () =>
    fetch('https://api.example.com/data').then(res => res.json())
  );

  if (isLoading) return <div>Loading...</div>;
  if (error) return <div>Error fetching data: {error.message}</div>;

  // Render UI with fetched data
}
```

GraphQL

If Your Backend Supports GraphQL, You Can Use Libraries Like Apollo Client Or Relay To Fetch Data Using GraphQL Queries.

Example :

```
import { useQuery, gql } from '@apollo/client';

const GET_DATA = gql`
  query {
    getData {
      // Query fields
    }
  }
`;

function MyComponent() {
  const { loading, error, data } = useQuery(GET_DATA);

  if (loading) return <div>Loading...</div>;
  if (error) return <div>Error fetching data: {error.message}</div>;

  // Render UI with fetched data
}
```

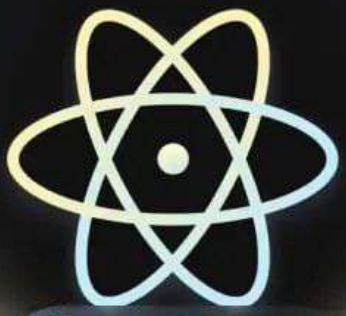
Thanks For Watching

I Hope You Found This Useful

Save This !

Different Ways To Fetch Data In React

You Need To Know About



Save It !

Follow : @_infoinfinity