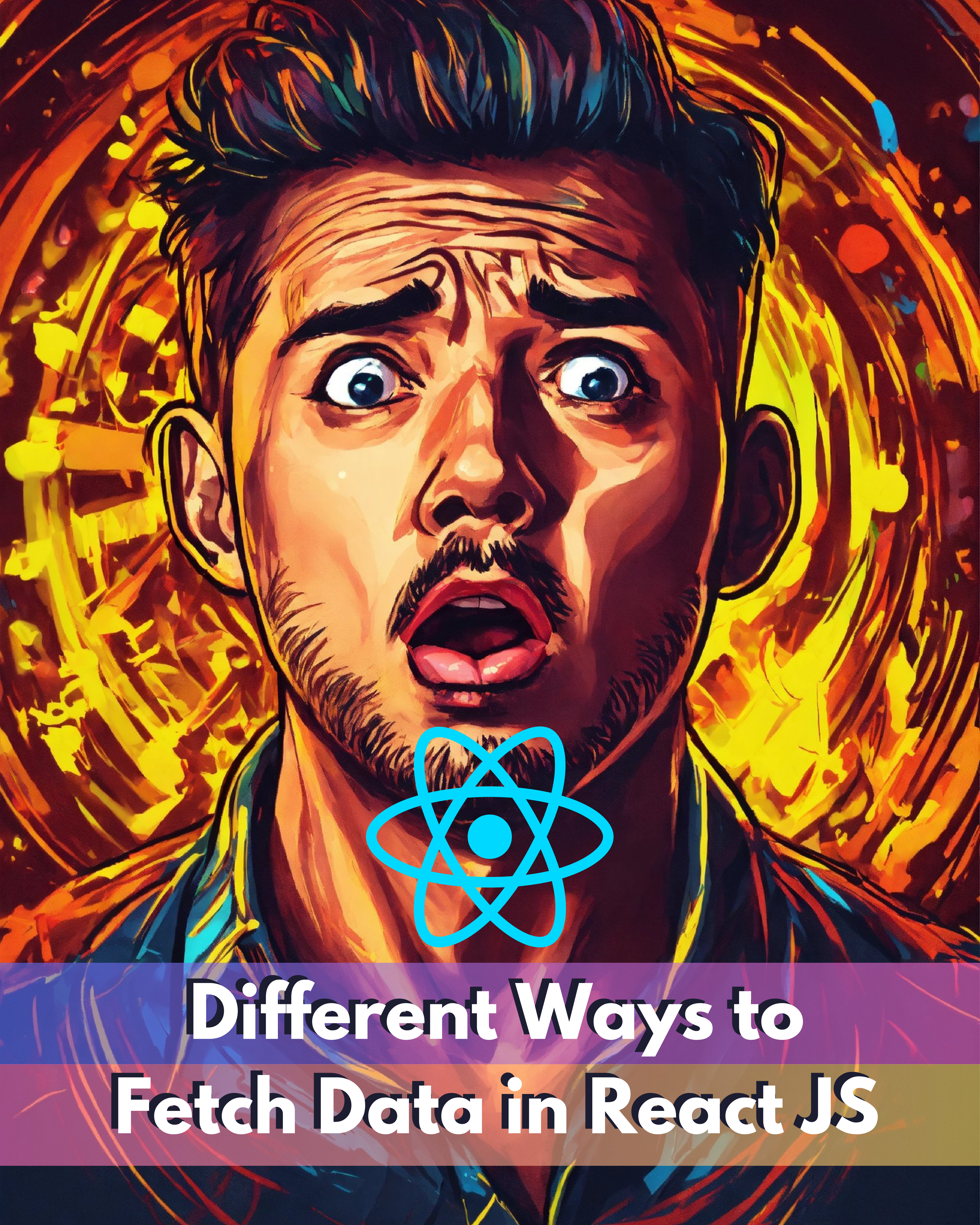


Different Ways to Fetch Data in React JS

Fetch method

The fetch() method in JS is used to request to the server and load the information in the webpages. The request can be of any APIs that return the data of the format JSON or XML. This method returns a promise.

```
function App() {

  useEffect(() => {
    fetch('https://site.com/')
    .then(response => response.json())
    .then(json => console.log(json))
    }, []);

  return (
    <div> Different ways to fetch Data </div>
  );
}
```

Async-Await

It is the preferred way of fetching the data from an API as it enables us to remove our .then() callbacks and return asynchronously resolved data. In the async block, we can use Await function to wait for the promise.

```
function App() {
  useEffect(() => {
    (async () => {
      try {
        const result = await axios.get('https://site.com/')
        console.log(result.data)
      } catch (error) {
        console.error(error)
      }
    })()
  })

  return <div>Different ways to fetch Data</div>
}
```

Axios library

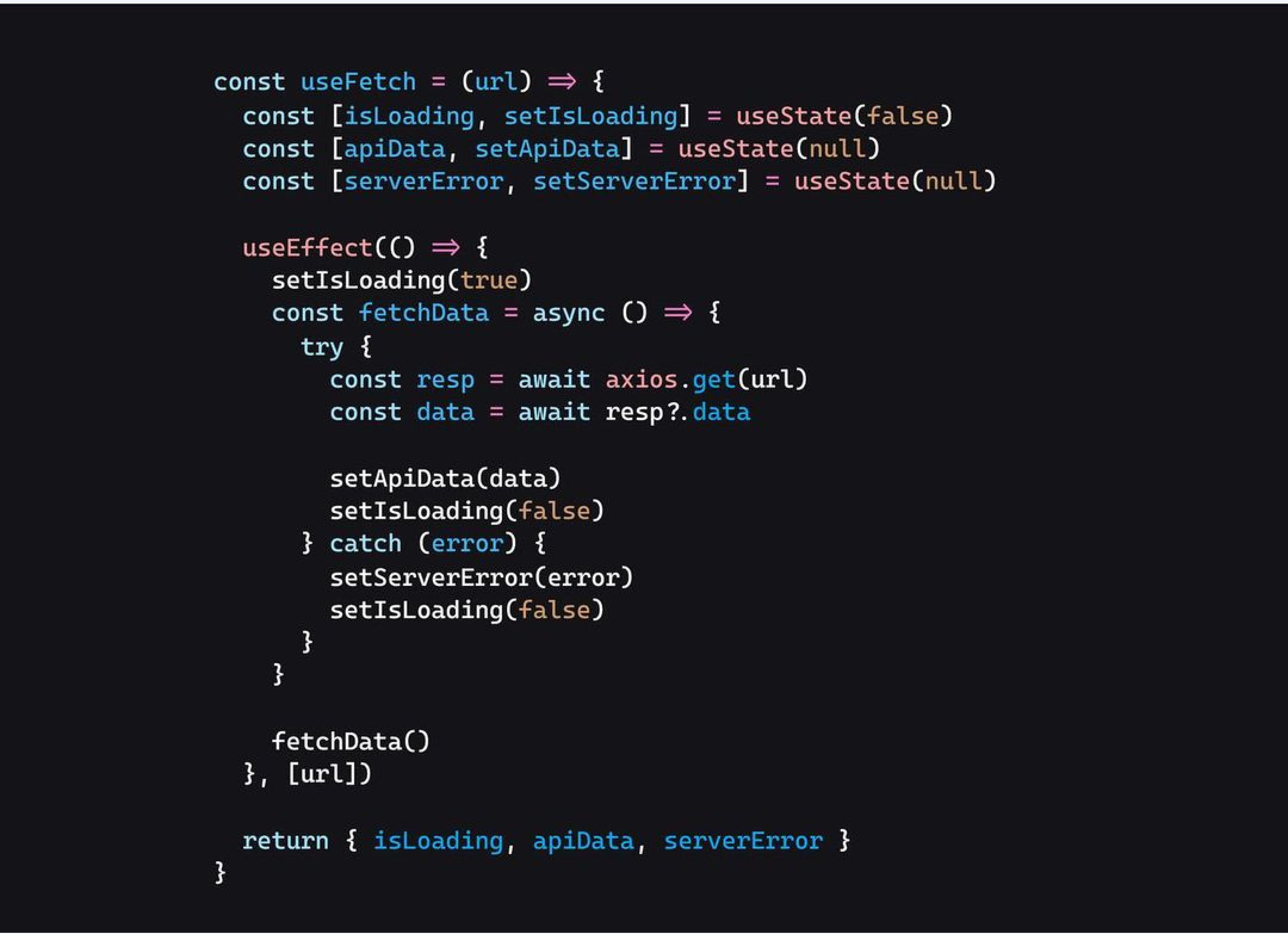
With Axios, we can easily send asynchronous HTTP requests to REST APIs & perform create, read, update and delete operations. Axios can be imported in plain JavaScript or with any library accordingly.

```
function App() {

  useEffect(() => {
    axios.get("https://site.com")
      .then((response) => console.log(response.data));
    }, []);

  return (
    <div>
        Different ways to fetch Data
    </div>
  );
}
```

DevTools99.com

Custom Hook

It is basically a React component whose name will start with “use” like *useFetch*. It can use one or more React hooks inside them.



```
const useFetch = (url) => {
  const [isLoading, setIsLoading] = useState(false)
  const [apiData, setApiData] = useState(null)
  const [serverError, setServerError] = useState(null)

  useEffect(() => {
    setIsLoading(true)
    const fetchData = async () => {
      try {
        const resp = await axios.get(url)
        const data = await resp?.data

        setApiData(data)
        setIsLoading(false)
      } catch (error) {
        setServerError(error)
        setIsLoading(false)
      }
    }

    fetchData()
  }, [url])

  return { isLoading, apiData, serverError }
}
```

DevTools99.com

Usage in the component

Import the *useFetch* hook and pass the URL of the API endpoint from where you want to fetch data.

Now just like any React hook we can directly use our custom hook to fetch the data.

```
import useFetch from "./useFetch";

const App = () => {
  const { isLoading, serverError, apiData } = useFetch('https://site.com')

  return (
    <div>
      <h2>API data</h2>
      {isLoading && <span>Loading.....</span>}
      {!isLoading && serverError ? (
        <span>Error in fetching data ... </span>
      ) : (
        <span>{JSON.stringify(apiData)}</span>
      )}
    </div>
  )
}
```

DevTools99.com

React Query library

With this we can achieve a lot more than just fetching data. It provides support for caching and refetching, which impacts the overall user experience by preventing irregularities and ensuring our app feels faster.

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

function App() {
  const { isLoading, error, data } =
  useQuery('posts', () => axios('https://site.com'))

  console.log(data)

  return <div>Different ways to fetch data</div>
}
```

DevTools99

Like, Share & Subscribe :

youtube.com/@DevTools99

Follow us on:

facebook.com/DevTools99

twitter.com/DevTools99

instagram.com/devtools99

pinterest.com/devtools99