React: usare le Fetch API

Le Fetch API sono un'interfaccia Javascript che consente di effettuare richieste HTTP e gestire le risposte in modo asincrono. Questa API è ampiamente utilizzata in React per recuperare dati da API REST e aggiornare lo stato del componente in modo dinamico. In questo articolo, vedremo come utilizzare le Fetch API in React per ottenere e gestire i dati.

Possiamo utilizzare la funzione `fetch()` per effettuare una richiesta HTTP e recuperare i dati. Ad esempio, se volessimo recuperare i dati da una API REST, potremmo utilizzare il seguente codice:

```
 fetch('https://api.example.com/data')
.then(response => response.json())
.then(data => console.log(data));
```

In questo codice, la funzione `fetch()` viene utilizzata per effettuare una richiesta HTTP all'indirizzo `https://api.example.com/data`. Una volta ottenuta la risposta, il metodo `json()` viene utilizzata per convertire i dati in formato JSON. Infine, i dati vengono registrati a titolo di esempio sulla console utilizzando il metodo `console.log()`.

Inoltre, per ottenere il massimo dalle Fetch API, possiamo utilizzare la sua funzionalità di configurazione. Ad esempio, se volessimo includere informazioni di autenticazione nella richiesta HTTP, potremmo utilizzare il seguente codice:

```
 fetch('https://api.example.com/data', {
        method: 'GET',
        headers: {
        'Authorization': 'Bearer ' + token
        }
})
.then(response => response.json())
.then(data => console.log(data));
```

In questo codice, stiamo passando un oggetto di configurazione come secondo parametro della funzione fetch(). L'oggetto di configurazione include informazioni come il metodo HTTP utilizzato (in questo caso `GET`) e gli header della richiesta HTTP, inclusa l'autenticazione con il Bearer token.

Infine, possiamo gestire gli errori in modo robusto utilizzando il metodo catch() per catturare eventuali errori nella richiesta HTTP o nella gestione dei dati. Ad esempio, potremmo utilizzare il seguente codice:

```
 fetch('https://api.example.com/data')
    .then(response => {
        if (!response.ok) {
            throw new Error(response.status);
        }
        return response.json();
})
.then(data => console.log(data))
.catch(error => console.error(error));
```

In questo codice, stiamo utilizzando la proprietà `ok` per verificare se la risposta HTTP è andata a buon fine. Se la risposta non è valida, viene sollevata un'eccezione con `throw`. In caso contrario, i dati vengono convertiti in formato JSON e registrati sulla console.

Possiamo astrarre la logica vista in precedenza in una funzione che sfrutti le Fetch API.

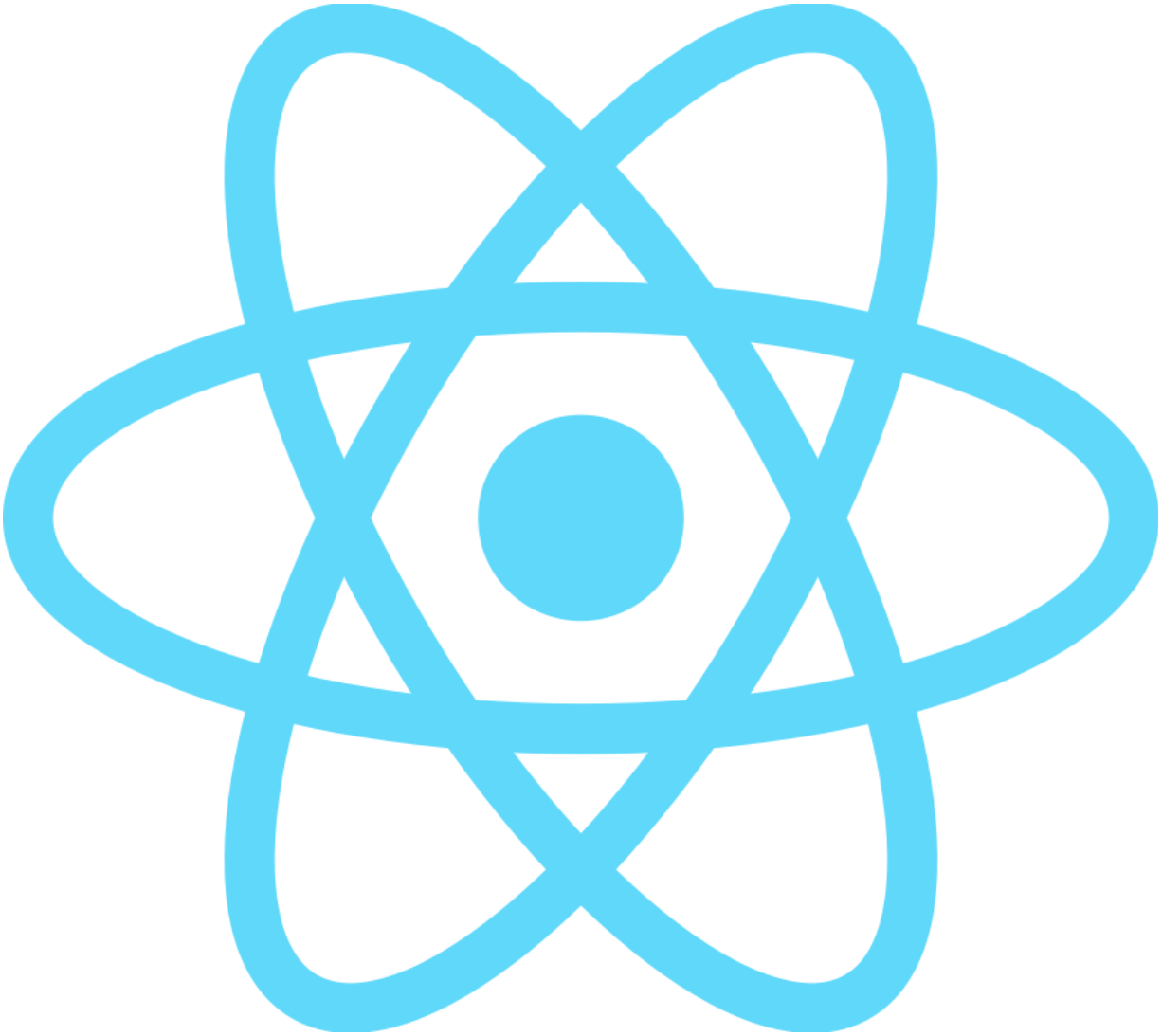
```
export async function getRequest(url = '') {
        try {
            const response = await fetch(url);
            if(!response.ok) {
                throw new Error(response.status);
            }
            return await response.json();
        } catch(err) {
            return [];
        }
}
```

Quindi possiamo usare questa funzione nei nostri componenti ad esempio per modificare il valore della proprietà dello state.

```
import { useState, useEffect } from 'react';
import { getRequest } from './lib/api';

export default function Test() {
    const [posts, setPosts] = useState([]);

    useEffect(() => {
            getRequest('https://api.example.com/posts').then(data => {
                setPosts(data);
            });
    }, []);

    return (
        <></>
    );
}
```

In sintesi, le Fetch API sono uno strumento potente per effettuare richieste HTTP asincrone in React.

Applicazioni Correlate



Book Store

Un'applicazione in React con Node.js e Fastify come backend.
DockerDocker ComposeNode.jsJavaScriptFastifyReact