React: JSX e metodi degli array

In React, JSX ci impone di padroneggiare i metodi degli array JavaScript.

In JSX (JavaScript XML), poiché si opera spesso all'interno di un'espressione, non possiamo usare un normale costrutto for/foreach/for..of per trasformare una sequenza di dati in elementi DOM o componenti specializzati. Invece, dobbiamo usare `map()` o `filter()` per questo scopo.

`map()`, il metodo degli array più utilizzato in React, di solito viene usato nel codice con il seguente schema:

1. Si inizia con un array, solitamente preso dallo state del componente.
2. Ora si può usare `map()`, ma è molto importante a questo punto aver impostato un array vuoto come valore iniziale predefinito nello state. In caso contrario, si ottiene un errore.
3. Se si usa la funzione di callback con un solo parametro, allora il parametro rappresenta l'elemento dell'array corrente (ad esempio un oggetto JSON recuperato da un'API REST. A seconda del numero di proprietà di un tale oggetto, si può utilizzare il destructuring degli oggetti per avere tutti i valori come variabili disponibili nello scope del callback corrente).
4. Infine, la funzione di callback restituirà un'espressione dove si possono usare le variabili JSX o componenti personalizzati.

Vediamo un esempio:

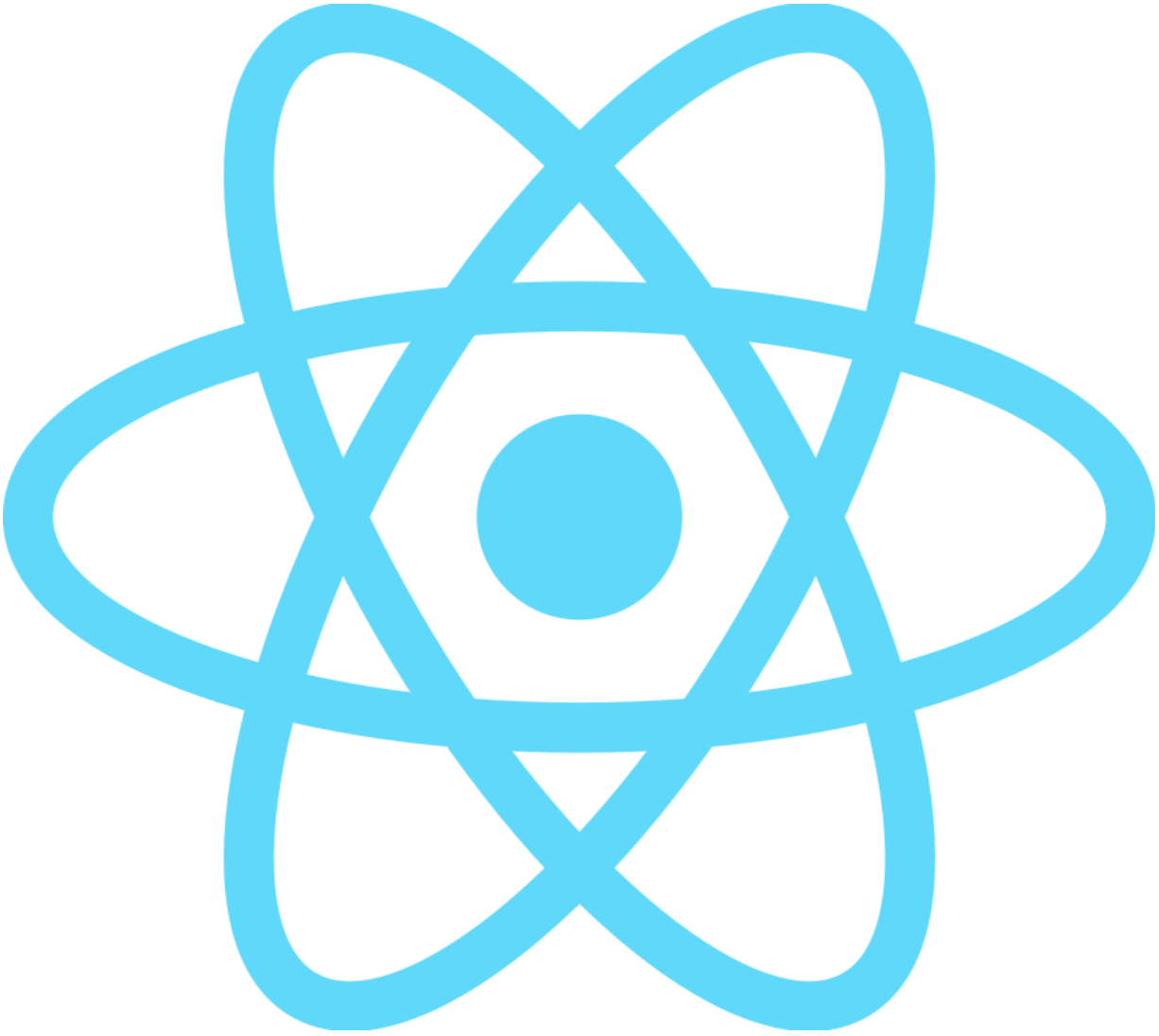
```
import { useState, useEffect } from 'react';
import axios from 'axios';

export default function Test() {
    // Array vuoto
    const [items, setItems] = useState([]);

    useEffect(() => {
        axios.get('/api/items').then(resp => {
            setItems(resp.data);
            // items[] è ora impostato
        });
    }, []);
    // Restituisce un'espressione.
    return (
        <ul className="items">
            {items.map(({ name, price, id }) => (
               <li key={id}>
                    { name } / { price }
                </li>
            ))}
        </ul>
    );
}
```

A questo punto abbiamo il componente `<Test />`. In questo caso, soddisfiamo il requisito di React di avere una chiave univoca per ogni elemento in una sequenza utilizzando la proprietà `id` di ciascun elemento. Se non ci sono identificatori univoci, possiamo usare l'indice dell'array passato come secondo argomento della funzione di callback:

```
return (
        <ul className="items">
            {items.map(({ name, price }, index) => (
                 <li key={index}>
                   { name } / { price }
                 </li>
            ))}
        </ul>
);
```

Si potrebbe essere tentati di scrivere una funzione che generi una stringa pseudo-casuale per soddisfare il requisito di React ma questa soluzione è lungi dall'essere ideale.

Applicazioni Correlate



- ### Book Store

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