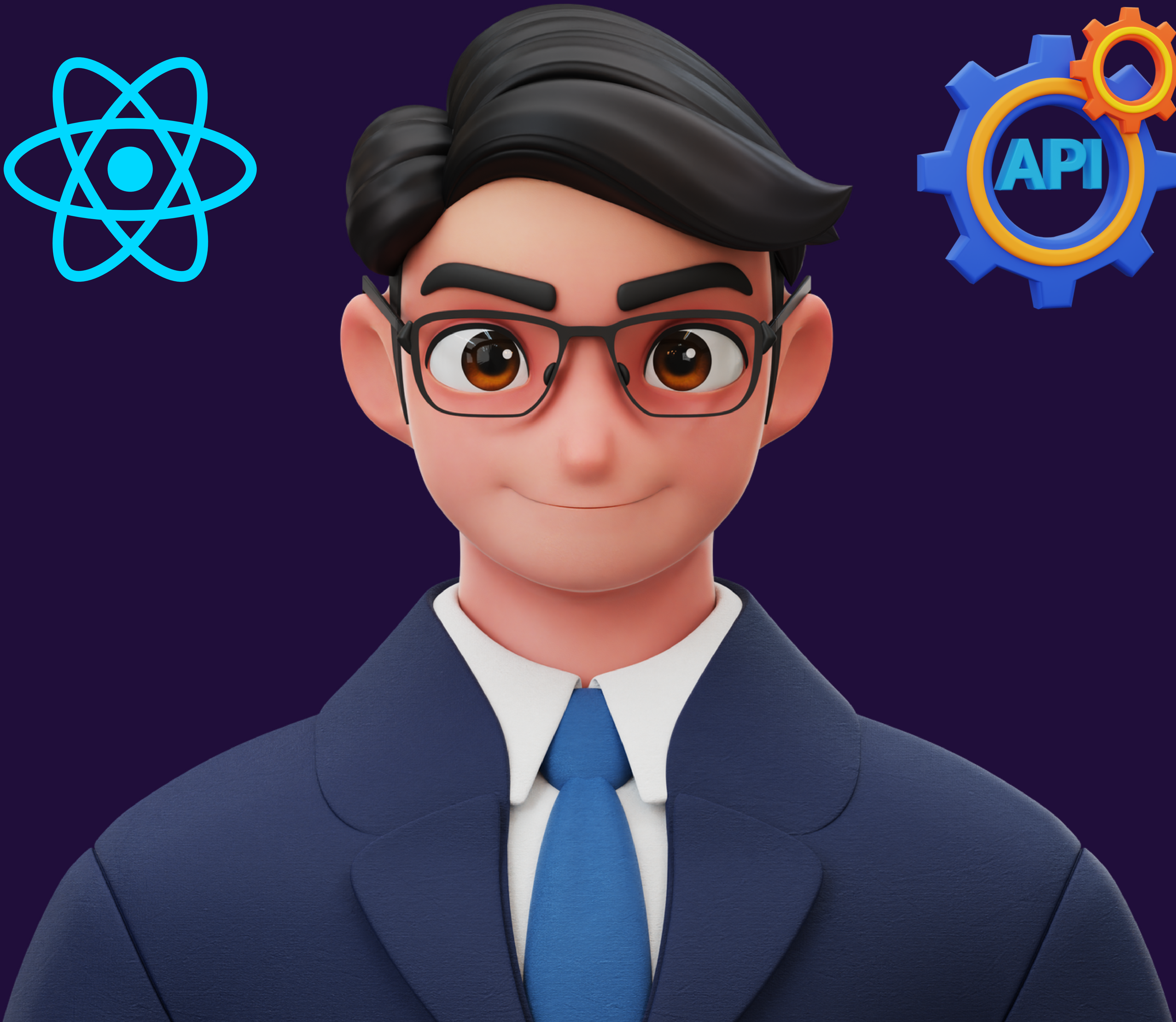
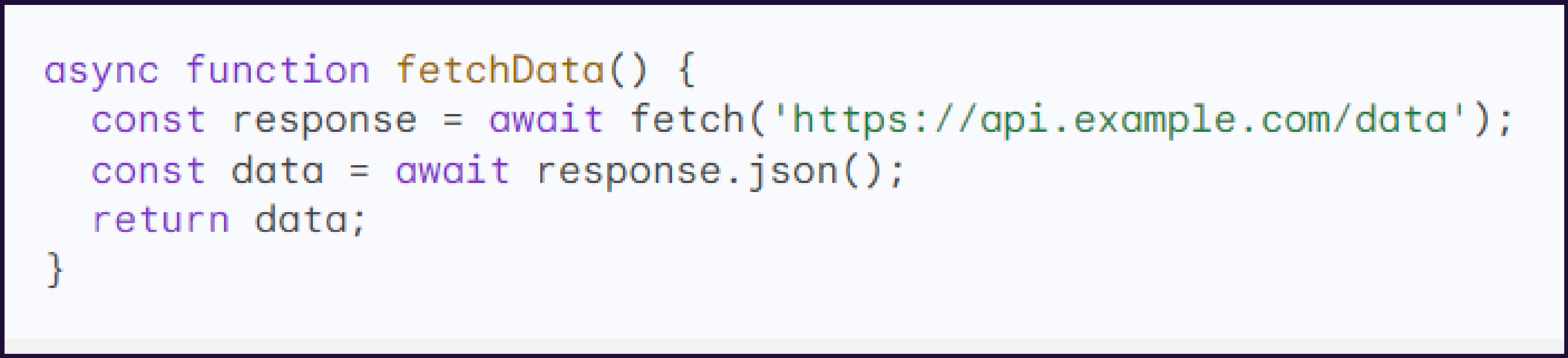
Mastering Data Fetching in React



1. Fetch API

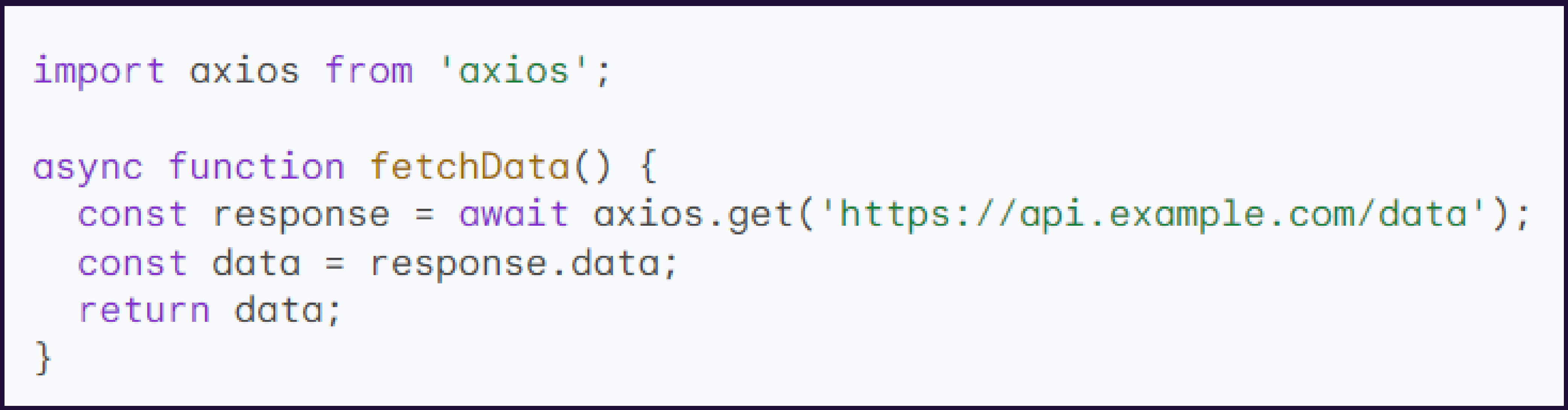
JavaScript's built-in method for making network requests



```
async function fetchData() {
  const response = await fetch('https://api.example.com/data');
  const data = await response.json();
  return data;
}
```

2. Axios

Popular, promise-based HTTP client library.

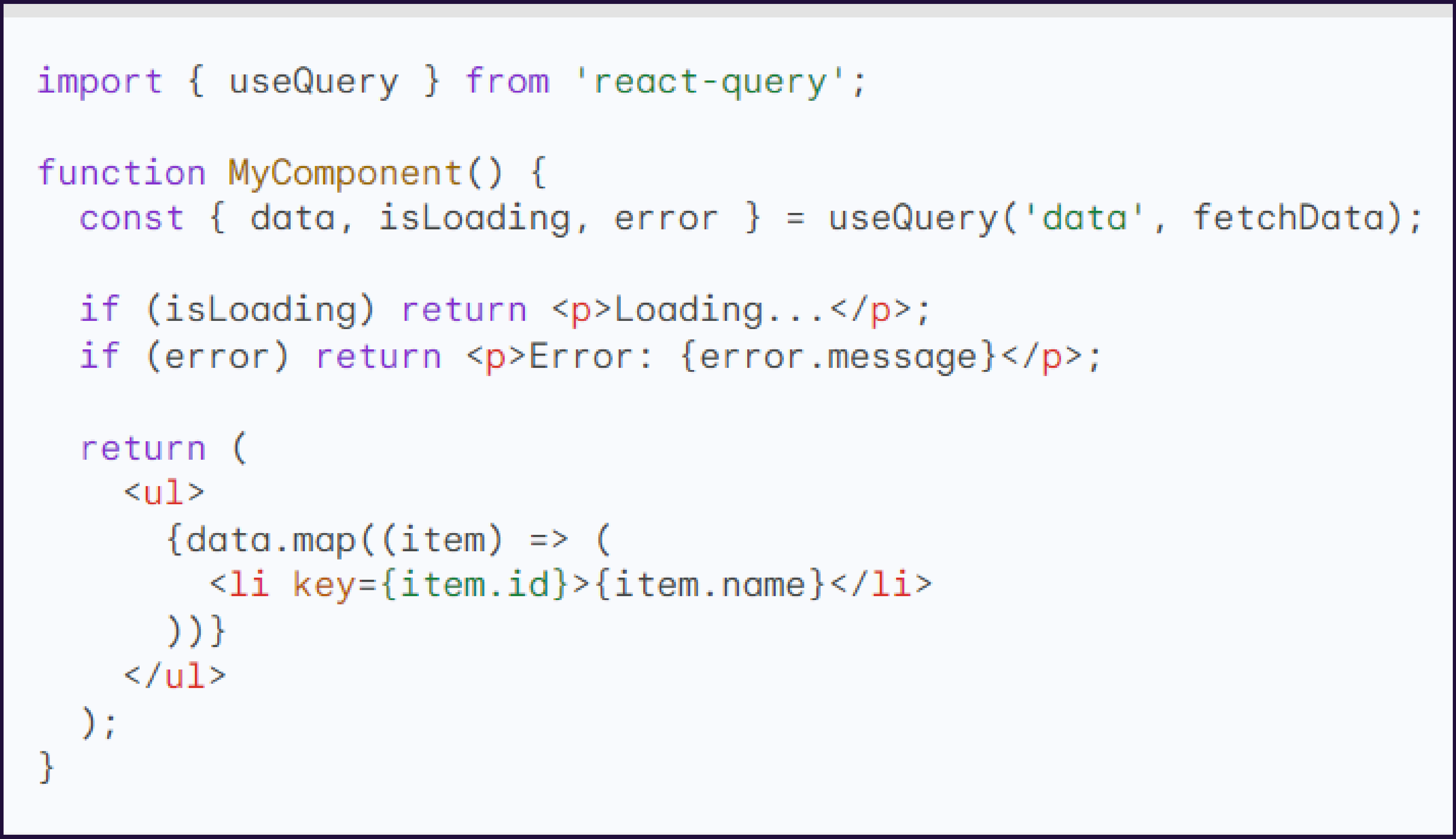


```
import axios from 'axios';

async function fetchData() {
  const response = await axios.get('https://api.example.com/data');
  const data = response.data;
  return data;
}
```

3. React Query

Powerful library for fetching, caching, and managing server state in React.



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

function MyComponent() {
  const { data, isLoading, error } = useQuery('data', fetchData);

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

  return (
    <ul>
      {data.map((item) => (
        <li key={item.id}>{item.name}</li>
      ))}
    </ul>
  );
}
```

4. Custom Hooks

Encapsulate data fetching logic and make it reusable across components.

```
function useFetchData() {
  const [data, setData] = useState(null);
  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      setIsLoading(true);
      try {
        const response = await fetch('https://api.example.com/data');
        const fetchedData = await response.json();
        setData(fetchedData);
      } catch (error) {
        setError(error);
      } finally {
        setIsLoading(false);
      }
    };

    fetchData();
  }, []);

  return { data, isLoading, error };
}
```

Follow for more