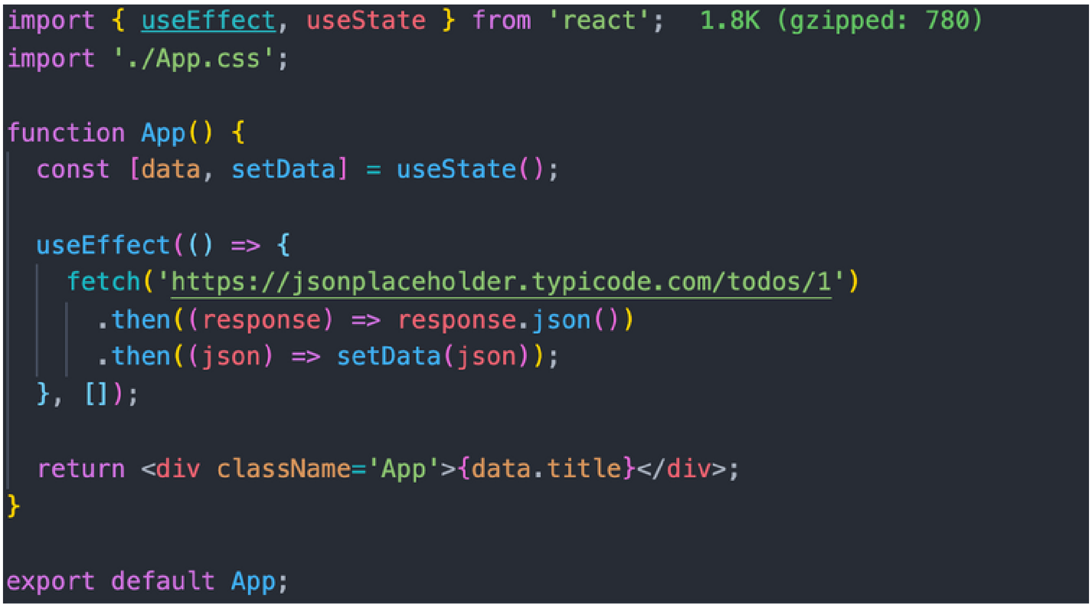
React Async API Calls

Getting data from a server in React and displaying the data can be daunting at first, so we will cover it.

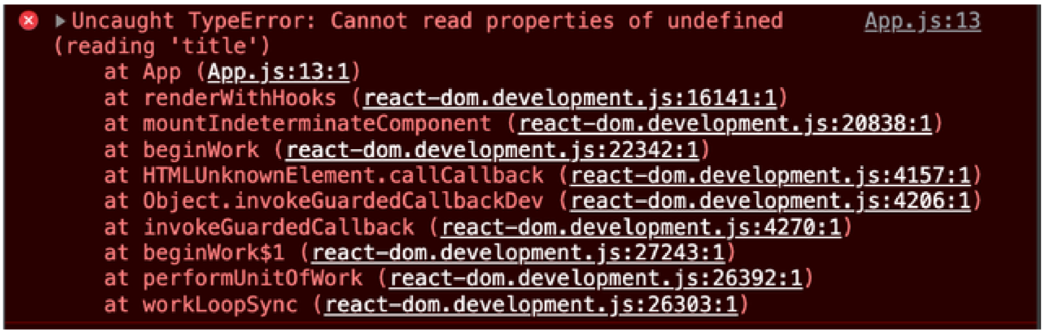
This is an **API** call that will fetch some random data and try to display it on the screen.



```
import { useEffect, useState } from 'react';  1.8K (gzipped: 780)
import './App.css';

function App() {
  const [data, setData] = useState();

  useEffect(() => {
    fetch('https://jsonplaceholder.typicode.com/todos/1')
      .then((response) => response.json())
      .then((json) => setData(json));
  }, []);

  return <div className='App'>{data.title}</div>;
}

export default App;
```

The Problem

If you look at the console, after you run the code, you'll get a nasty error.



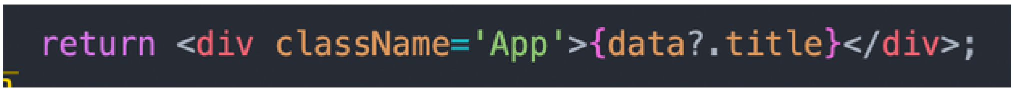
```
▸ Uncaught TypeError: Cannot read properties of undefined      App.js:13
(reading 'title')
    at App (App.js:13:1)
    at renderWithHooks (react-dom.development.js:16141:1)
    at mountIndeterminateComponent (react-dom.development.js:20838:1)
    at beginWork (react-dom.development.js:22342:1)
    at HTMLUnknownElement.callCallback (react-dom.development.js:4157:1)
    at Object.invokeGuardedCallbackDev (react-dom.development.js:4206:1)
    at invokeGuardedCallback (react-dom.development.js:4270:1)
    at beginWork$1 (react-dom.development.js:27243:1)
    at performUnitOfWork (react-dom.development.js:26392:1)
    at workLoopSync (react-dom.development.js:26303:1)
```

The reason why this is happening is because React does **NOT** care if we already have our data or not. Remember that the action of getting data from a server is, itself, **asynchronous**.

Solution #1

Use Optional Chaining

Optional chaining ?. is a safe way to access nested object properties, even if an intermediate property **doesn’t exist.**



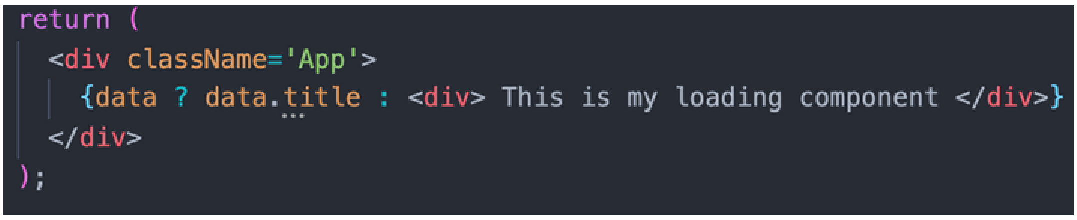
```
return <div className='App'>{data?.title}</div>;
```

This will solve the issue however it's certainly not the most effective way of solving this issue.

Solution #2

The most effective way to solve this issue is to have a **check** for asynchronous data.

Whether that's an **if statement** or a **ternary operator**, this is ideal.



```
return (
  <div className='App'>
    {data ? data.title : <div> This is my loading component </div>}
  </div>
);
```

Ideally, you would have a whole component or interface dedicated to this.