Front End Testing

•••

Or: How to Help Your Tomorrow Self Today

Testing Goals

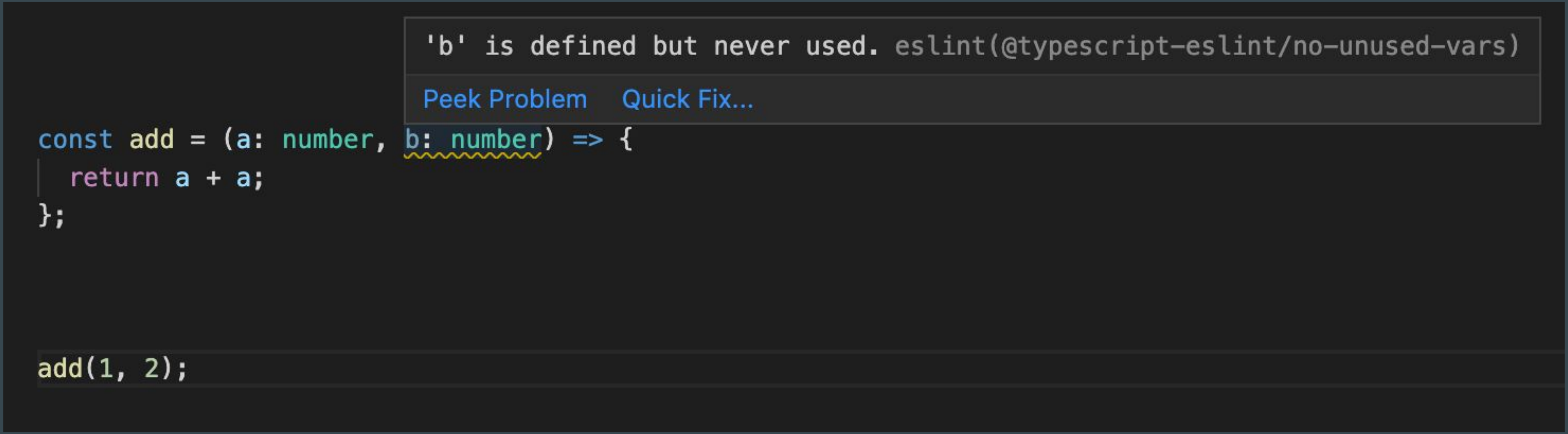
- Prevent simple mistakes
- Minimize manual checks
- Encode user flows
- Gain confidence
    - Code actually works
    - Easier refactoring

Types of Tests

- Static
- Unit
- Integration
- End to End

Static Testing

- Tools
  - Typescript
  - Eslint
  - Flow
- Pros
  - Very Fast
  - Relatively light-weight
  - Catch typos
  - Catch type errors
- Cons
  - No complex logic

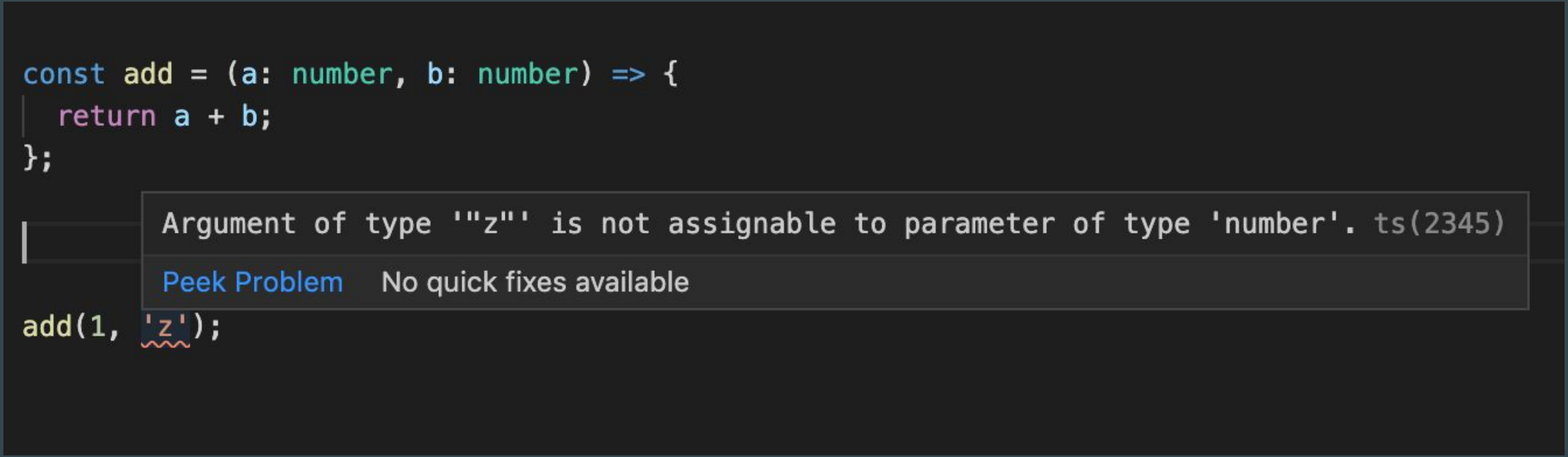


```
'b' is defined but never used. eslint(@typescript-eslint/no-unused-vars)
```

```ts
const add = (a: number, b: number) => {
  return a + a;
};



add(1, 2);
```



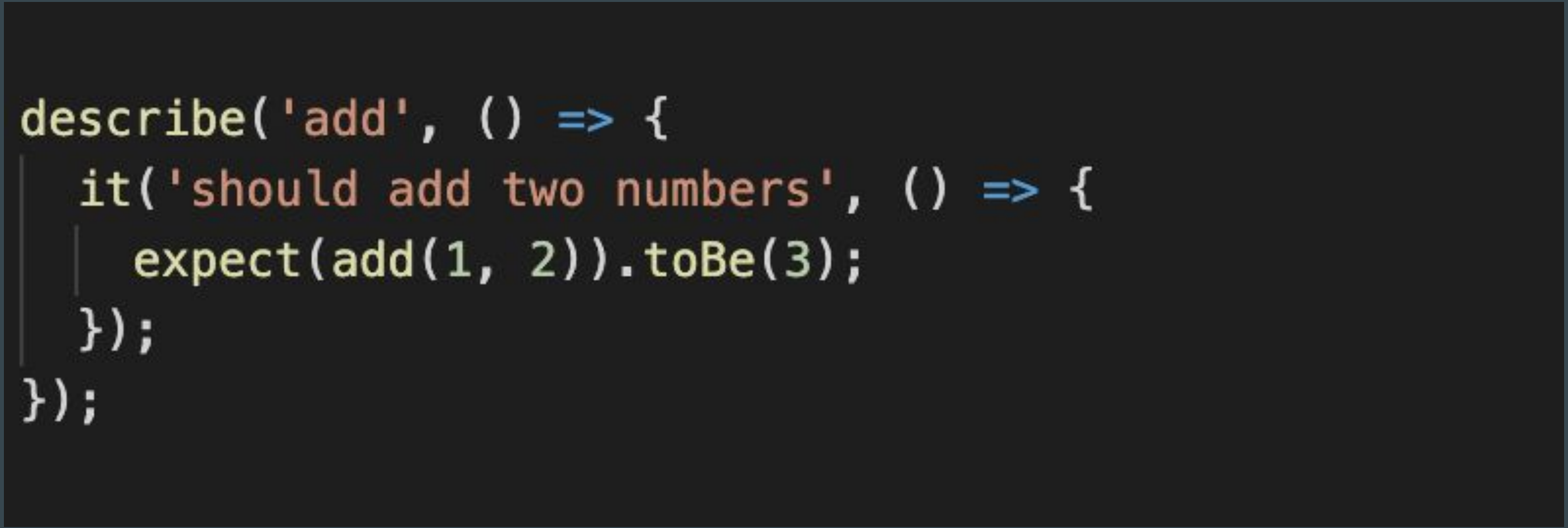
```
const add = (a: number, b: number) => {
  return a + b;
};

add(1, 'z');
```

```
Argument of type '"z"' is not assignable to parameter of type 'number'. ts(2345)
```

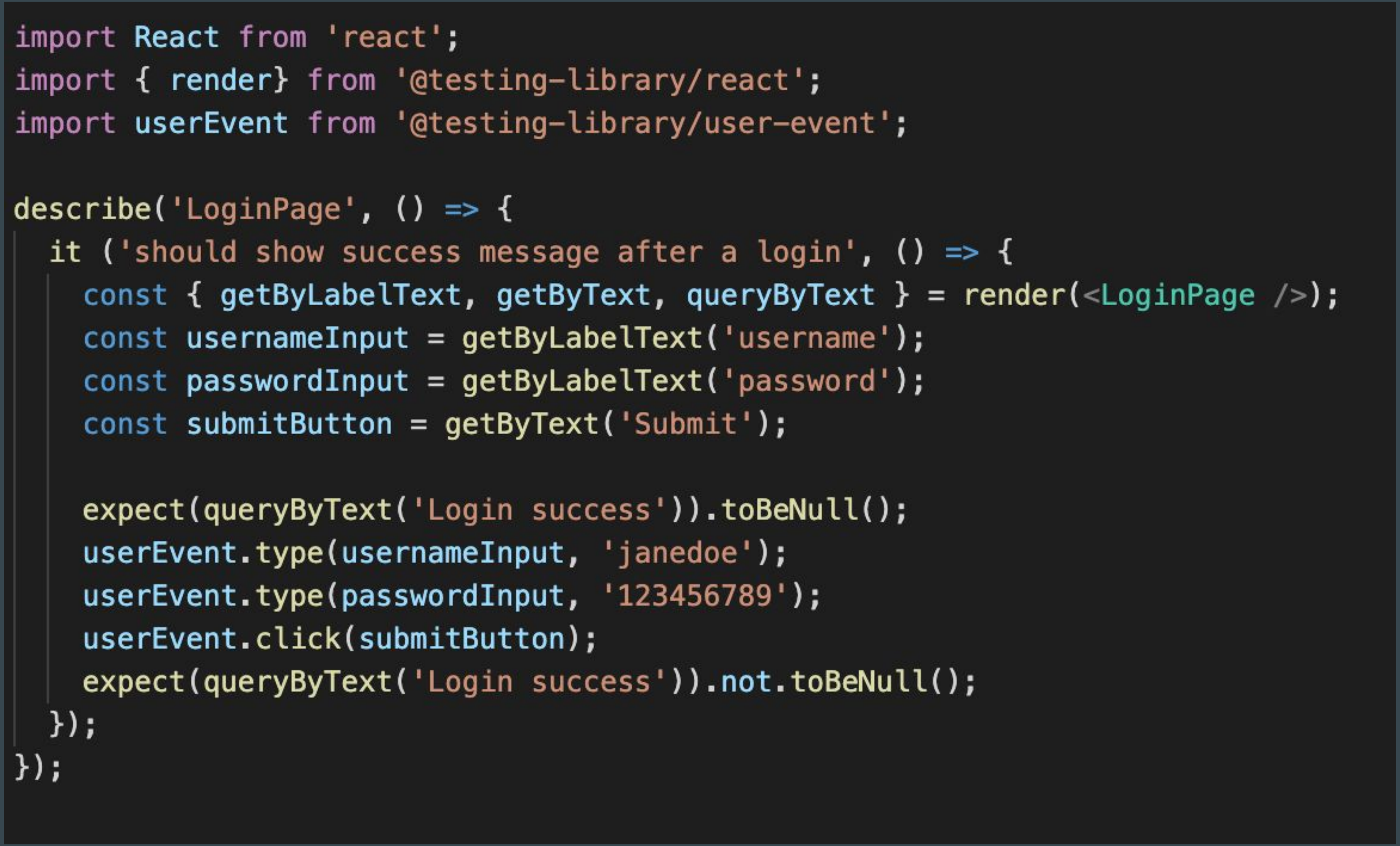
Unit Testing

- Tools
  - Jest
  - React Testing Library
  - Enzyme
- Pros
  - Relatively fast
  - Relatively light-weight
  - Test complex logic
  - Easy setup
- Cons
  - Isolated


```
describe('add', () => {
  it('should add two numbers', () => {
    expect(add(1, 2)).toBe(3);
  });
});
```

Integration Testing

- Tools
  - Jest
  - React Testing Library
  - Enzyme
- Pros
  - Relatively fast
  - Relatively light-weight
  - Test complex logic
  - Test multiple components
- Cons
  - Harder setup



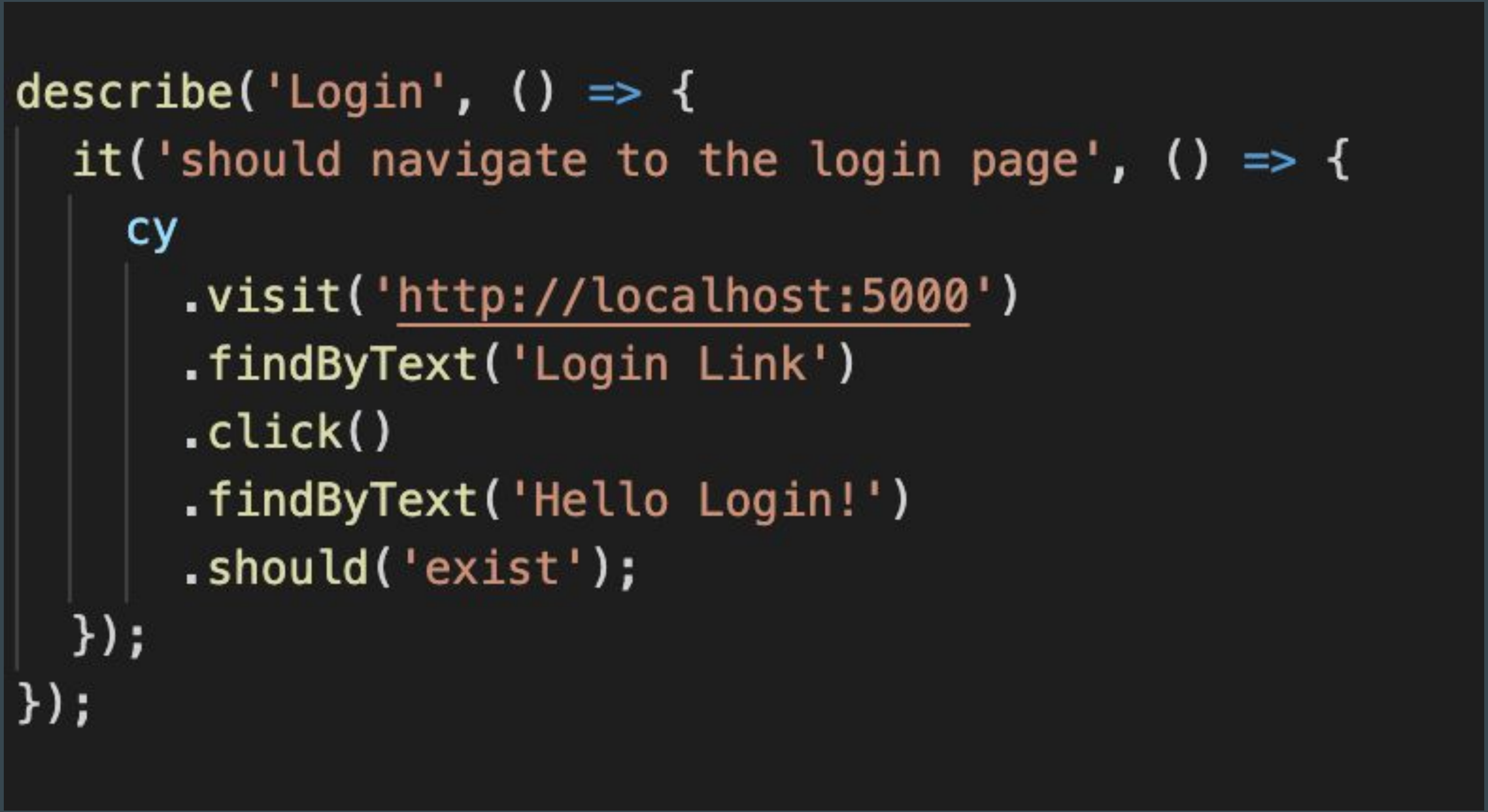
```
import React from 'react';
import { render} from '@testing-library/react';
import userEvent from '@testing-library/user-event';

describe('LoginPage', () => {
  it ('should show success message after a login', () => {
    const { getByLabelText, getByText, queryByText } = render(<LoginPage />);
    const usernameInput = getByLabelText('username');
    const passwordInput = getByLabelText('password');
    const submitButton = getByText('Submit');

    expect(queryByText('Login success')).toBeNull();
    userEvent.type(usernameInput, 'janedoe');
    userEvent.type(passwordInput, '123456789');
    userEvent.click(submitButton);
    expect(queryByText('Login success')).not.toBeNull();
  });
});
```

E2E Testing

- Tools
  - Cypress
  - Selenium
- Pros
  - Test full user flows
- Cons
  - Slow
  - Heavy-weight



```
describe('Login', () => {
  it('should navigate to the login page', () => {
    cy
      .visit('http://localhost:5000')
      .findByText('Login Link')
      .click()
      .findByText('Hello Login!')
      .should('exist');
  });
});
```

Semantics

- Unit
    - Single component or function
- Integration
    - Multiple components or functions
- End to End/Functional
    - Entire user flows

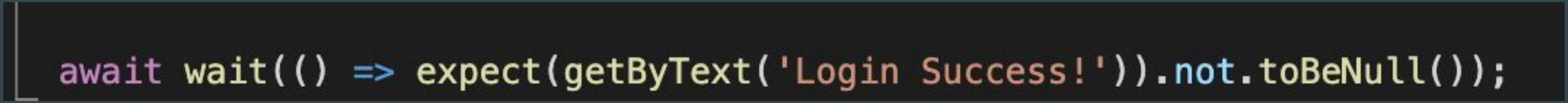
* It also depends who you ask

Common Pitfalls

- Async rendering
- File imports
- Using context
- Using mocks

Async rendering

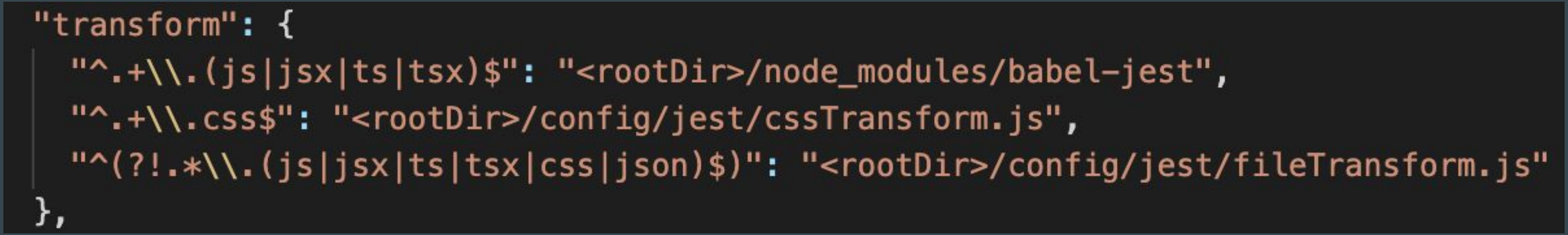
- Problem
  - Clicking a button makes a fetch call, and then component updates
- Solution
  - Use wait from @testing-library/react



```
await wait(() => expect(getByText('Login Success!')).not.toBeNull());
```

File Imports

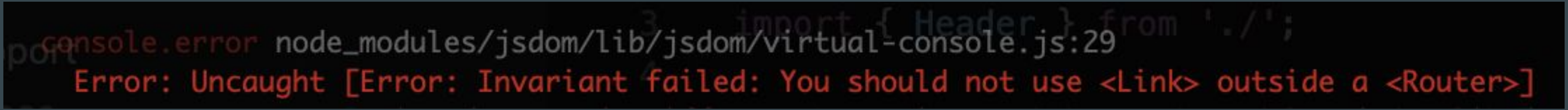
- Problem
  - Your component imports a non-js file like an svg
- Solution
  - Use a transform in jest.config.js



```
"transform": {
  "^.+\\.(js|jsx|ts|tsx)$": "<rootDir>/node_modules/babel-jest",
  "^.+\\.css$": "<rootDir>/config/jest/cssTransform.js",
  "^(?!.*\\.(js|jsx|ts|tsx|css|json)$)": "<rootDir>/config/jest/fileTransform.js"
},
```

Using Context

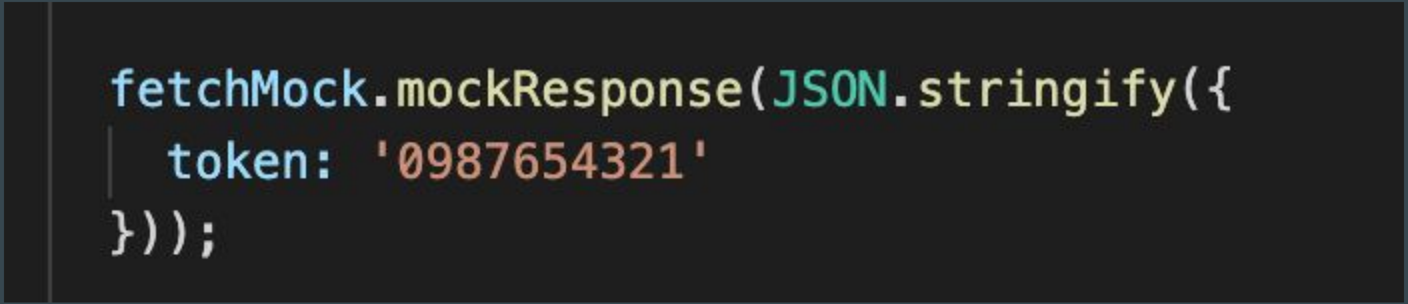
- Problem
  - Your component uses Context dependant data
- Solution
  - Use a provider when rendering your component



```
console.error node_modules/jsdom/lib/jsdom/virtual-console.js:29
  Error: Uncaught [Error: Invariant failed: You should not use <Link> outside a <Router>]
```

Using Mocks

- Problem
  - Your component makes fetch calls
- Solution
  - Use a library like jest-fetch-mock



```
fetchMock.mockResponse(JSON.stringify({
  token: '0987654321'
}));
```

Let’s Code!