Front End

Class 17

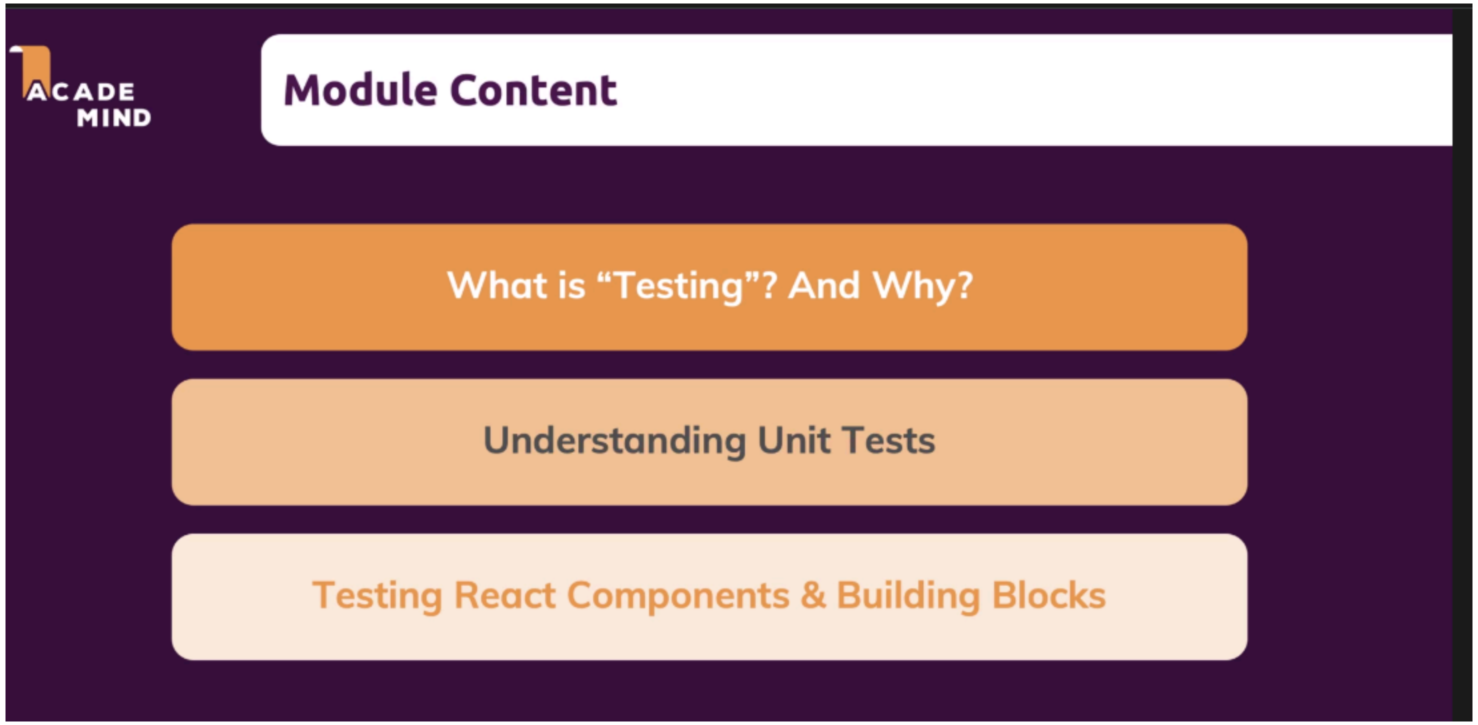
October 20, 2021

Agenda

- Kahoot
- Review
- Studio

Testing React Apps

Automated Testing

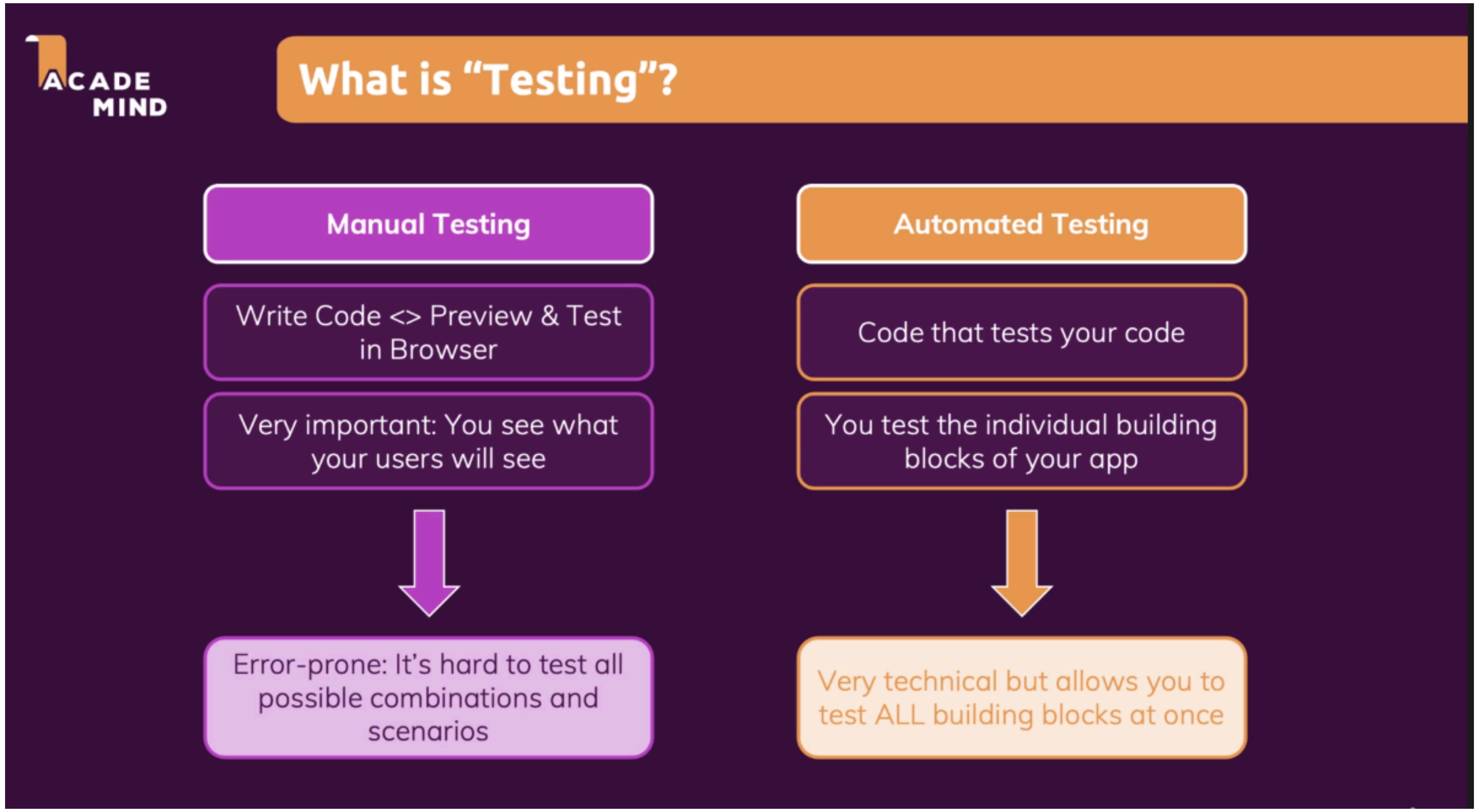


Module Content

What is “Testing”? And Why?

Understanding Unit Tests

Testing React Components & Building Blocks



What is "Testing"?

Manual Testing

Write Code <> Preview & Test in Browser

Very important: You see what your users will see

↓

Error-prone: It's hard to test all possible combinations and scenarios

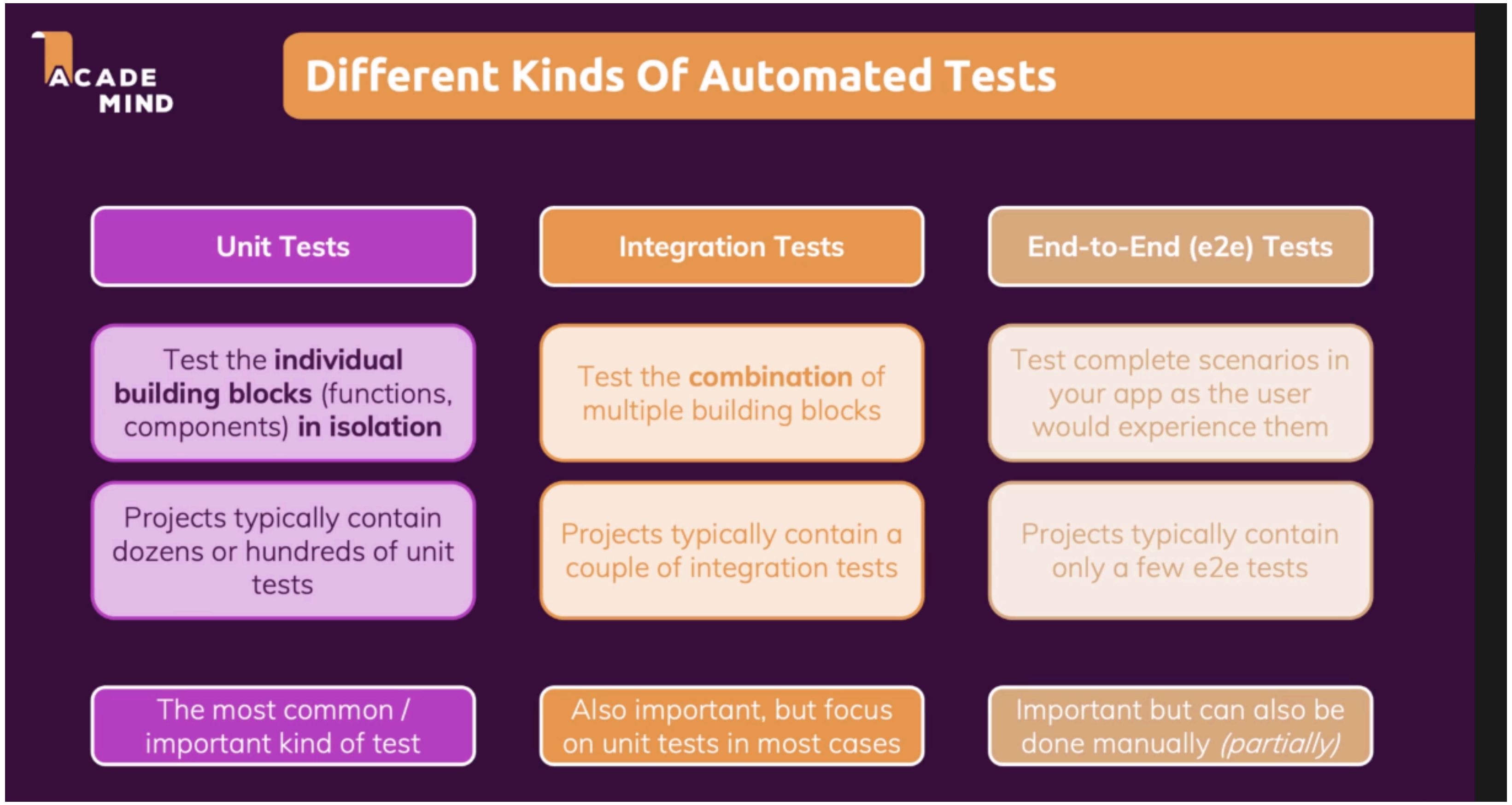
Automated Testing

Code that tests your code

You test the individual building blocks of your app

↓

Very technical but allows you to test ALL building blocks at once

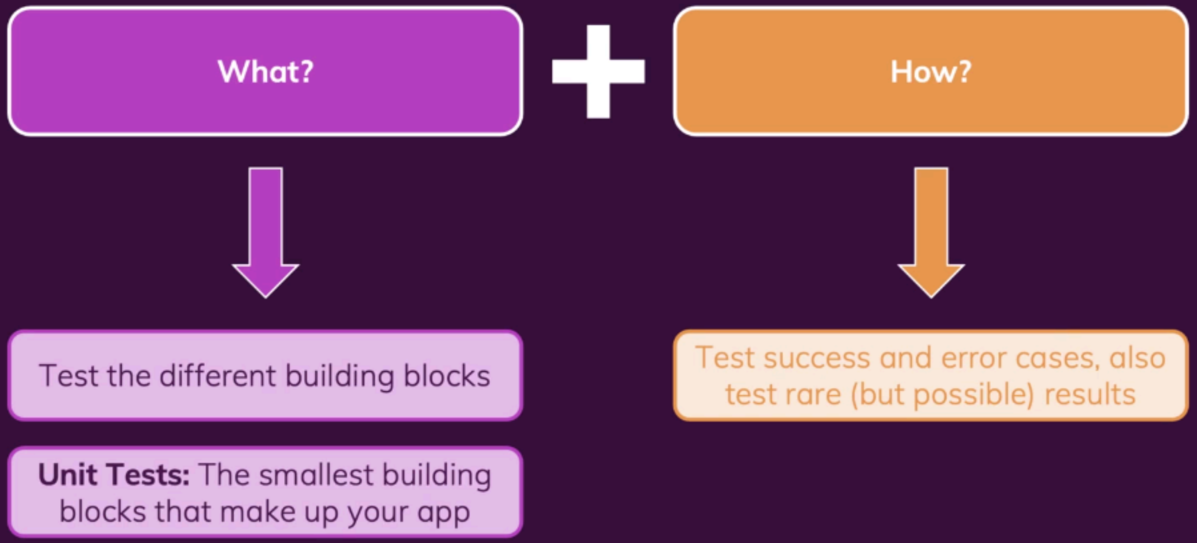


ACADE MIND

Different Kinds Of Automated Tests

Unit Tests	Integration Tests	End-to-End (e2e) Tests
Test the **individual building blocks** (functions, components) **in isolation**	Test the **combination** of multiple building blocks	Test complete scenarios in your app as the user would experience them
Projects typically contain dozens or hundreds of unit tests	Projects typically contain a couple of integration tests	Projects typically contain only a few e2e tests
The most common / important kind of test	Also important, but focus on unit tests in most cases	Important but can also be done manually *(partially)*

What To Test



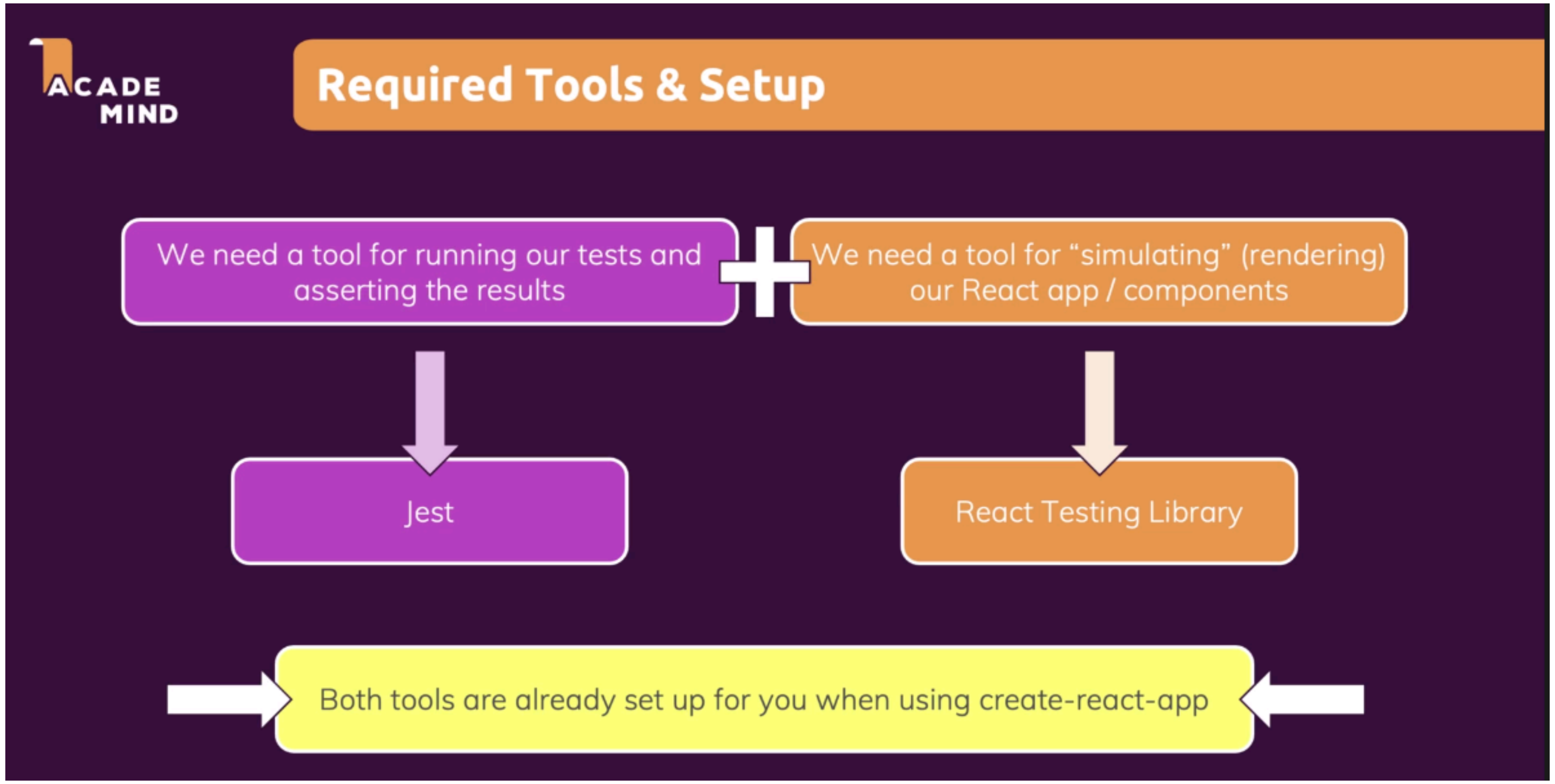
What?

Test the different building blocks

Unit Tests: The smallest building blocks that make up your app

How?

Test success and error cases, also test rare (but possible) results



ACADE MIND

Required Tools & Setup

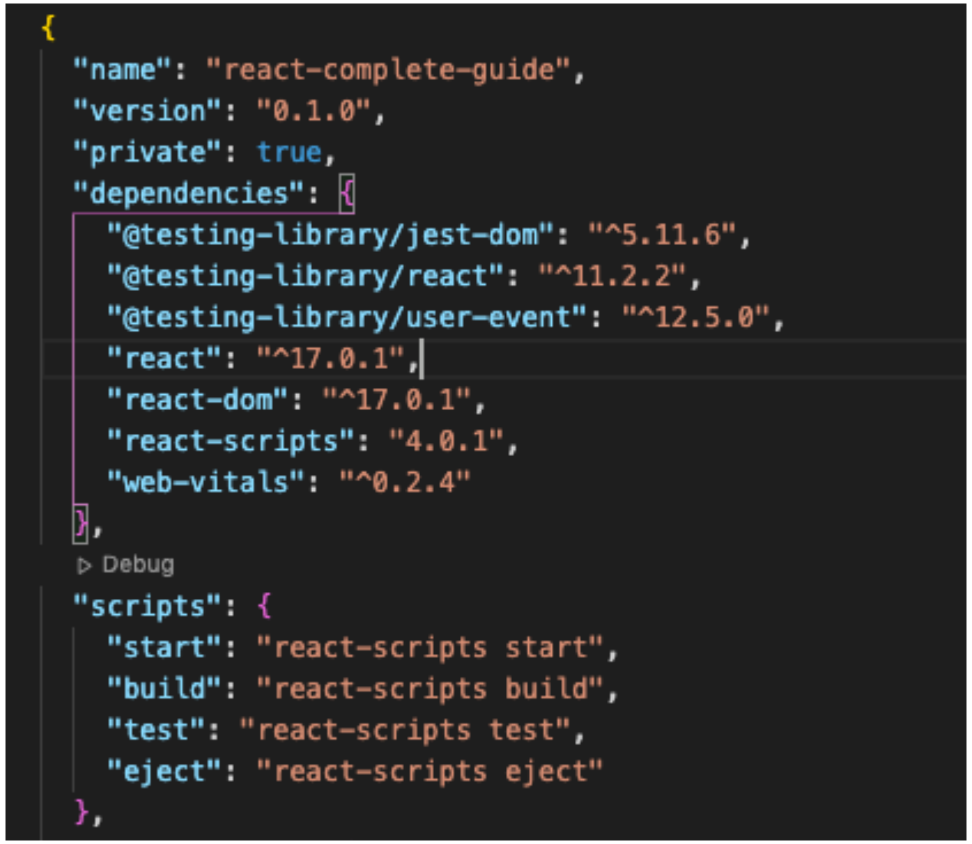
We need a tool for running our tests and asserting the results

We need a tool for "simulating" (rendering) our React app / components

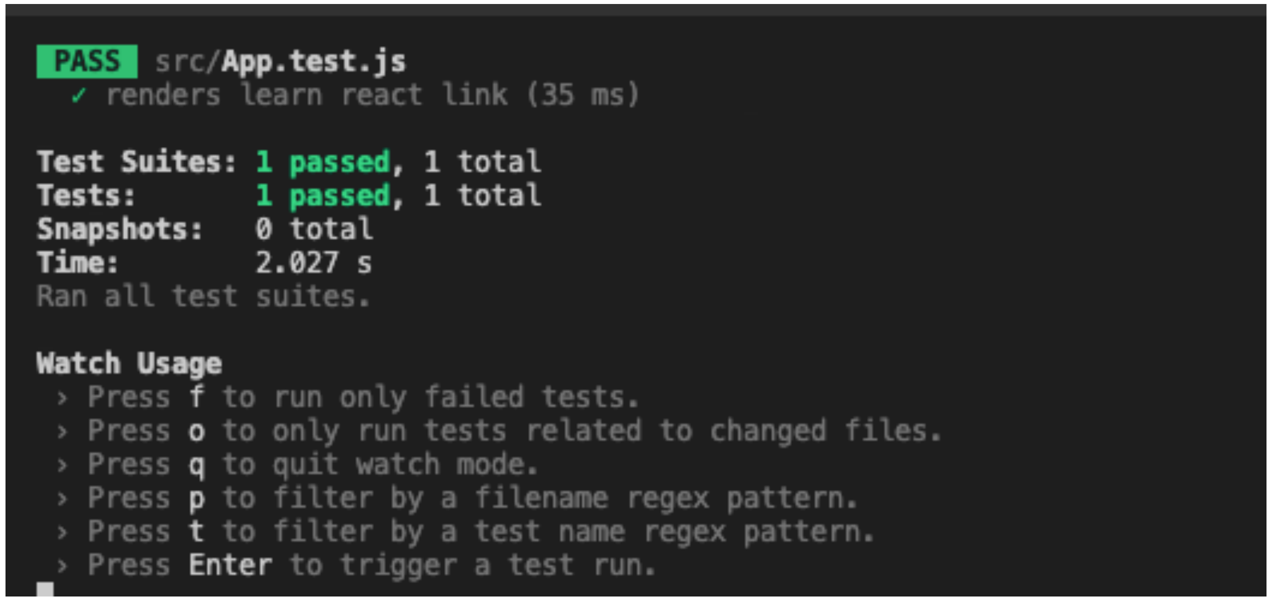
Jest

React Testing Library

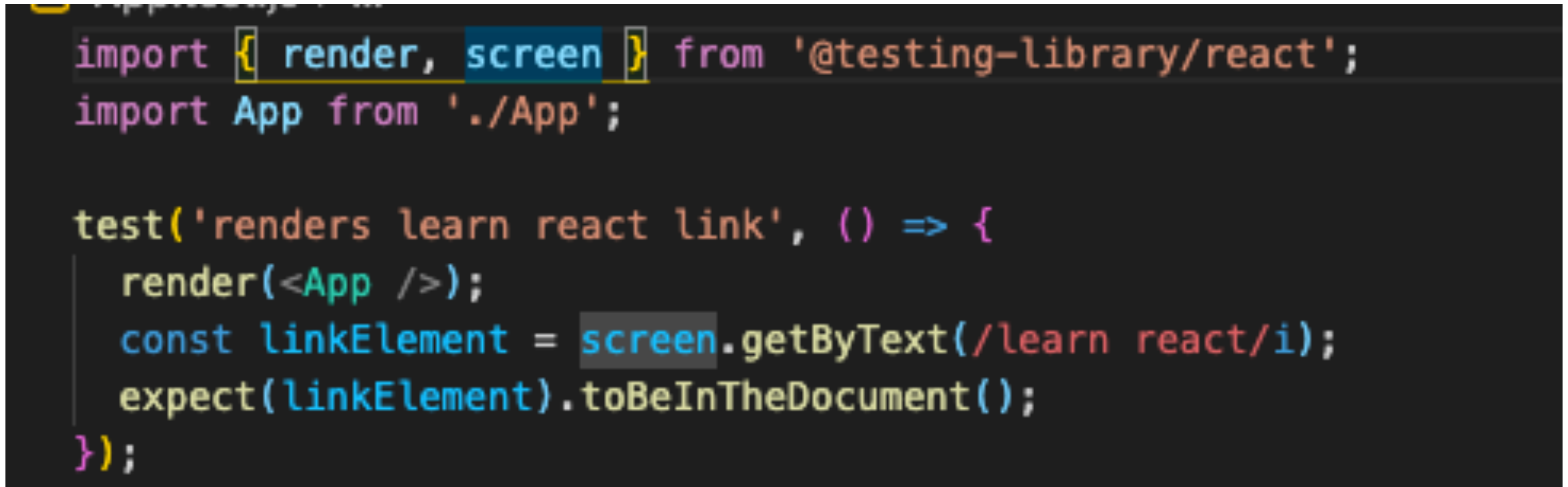
Both tools are already set up for you when using create-react-app package.json



```
{
  "name": "react-complete-guide",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.11.6",
    "@testing-library/react": "^11.2.2",
    "@testing-library/user-event": "^12.5.0",
    "react": "^17.0.1",
    "react-dom": "^17.0.1",
    "react-scripts": "4.0.1",
    "web-vitals": "^0.2.4"
  },
  ▷ Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
```

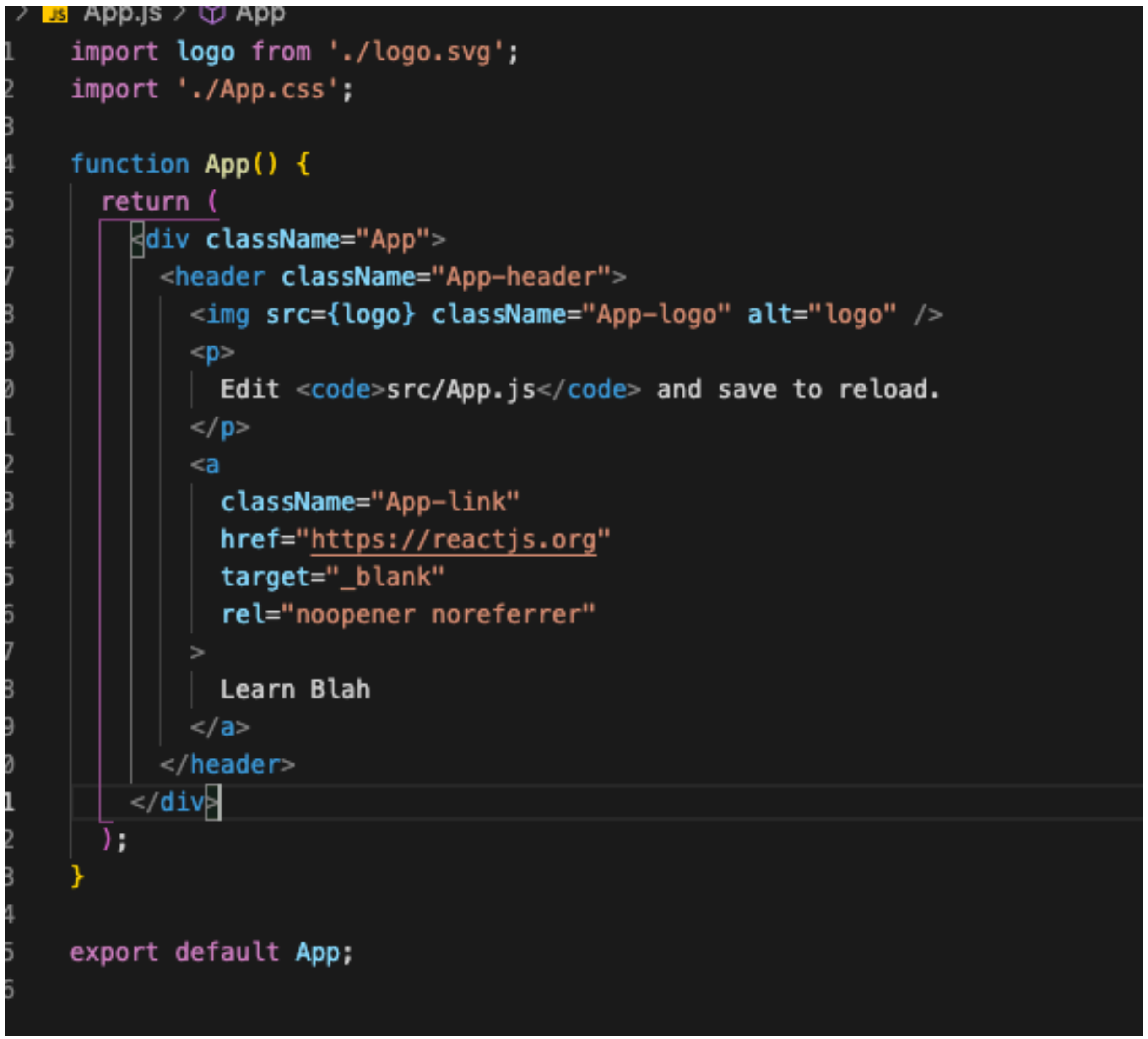


```
 PASS  src/App.test.js
  ✓ renders learn react link (35 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.027 s
Ran all test suites.

Watch Usage
 › Press f to run only failed tests.
 › Press o to only run tests related to changed files.
 › Press q to quit watch mode.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
```



```
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

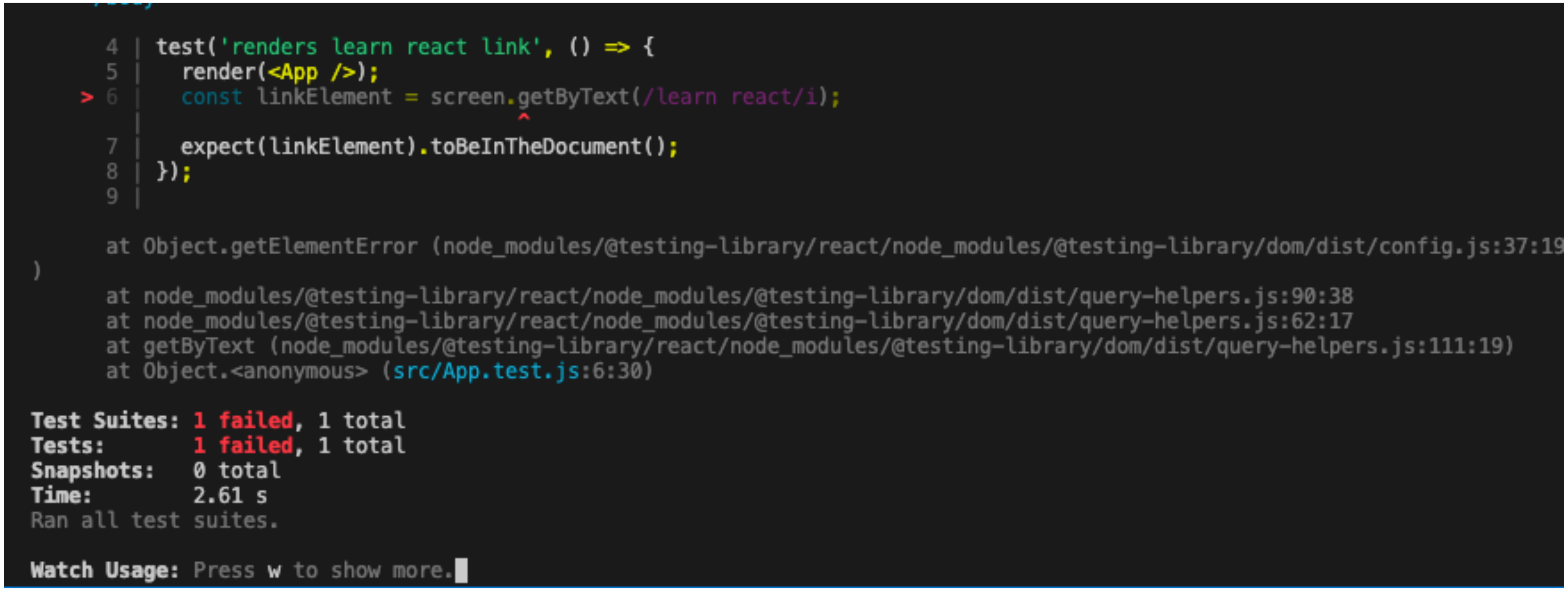
App Component



App.js > App

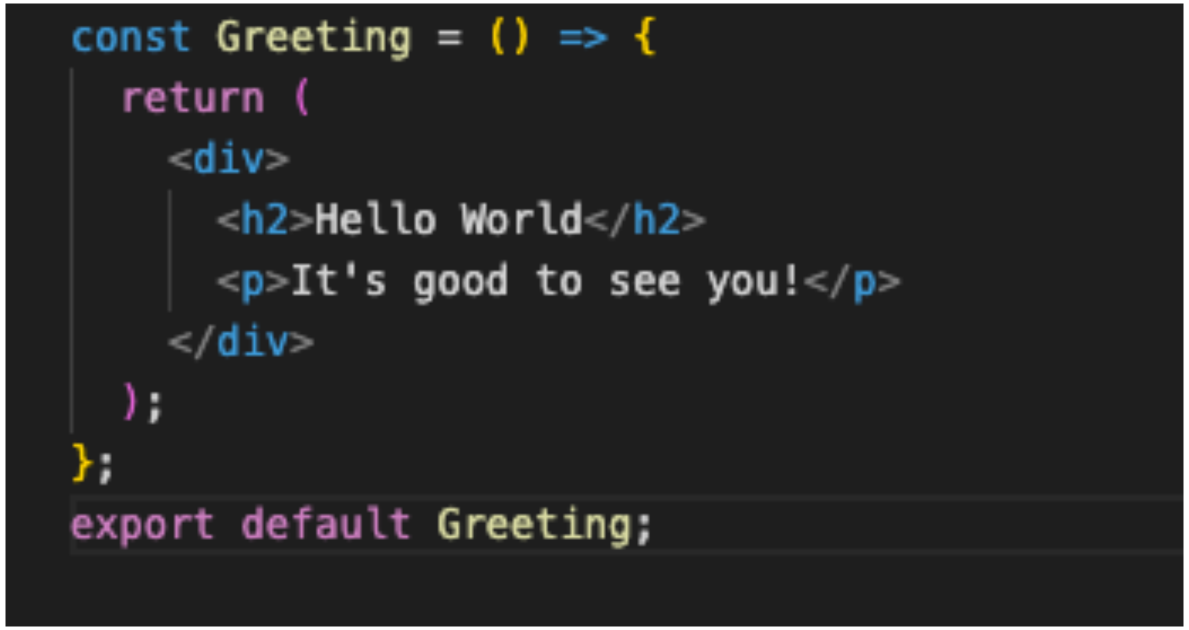
```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn Blah
        </a>
      </header>
    </div>
  );
}

export default App;
```

npm test

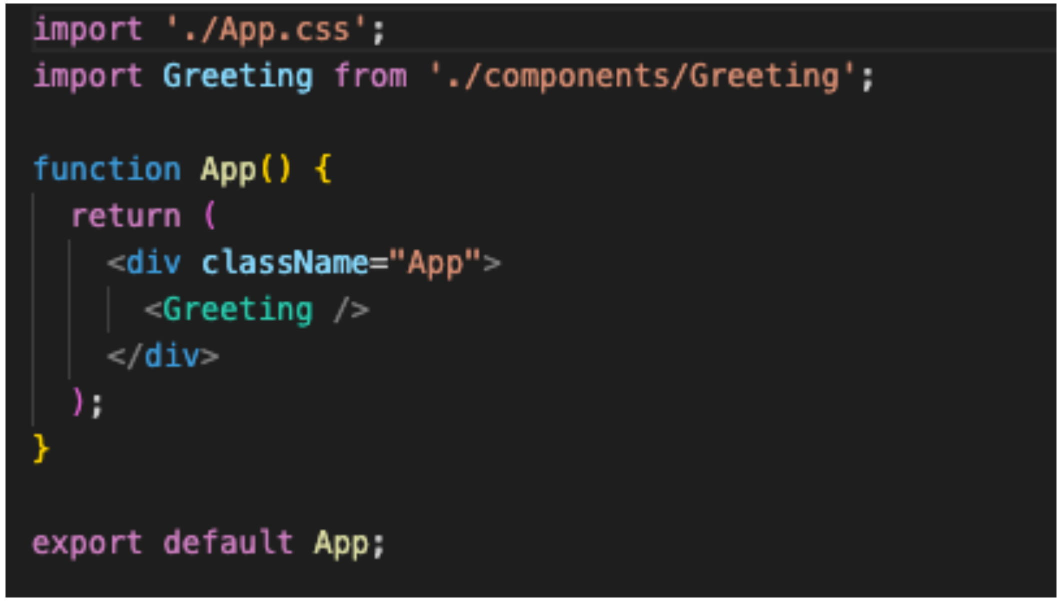


```
      4 | test('renders learn react link', () => {
      5 |   render(<App />);
    > 6 |   const linkElement = screen.getByText(/learn react/i);
        |                             ^
      7 |   expect(linkElement).toBeInTheDocument();
      8 | });
      9 |

      at Object.getElementError (node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/config.js:37:19
)
      at node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:90:38
      at node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:62:17
      at getByText (node_modules/@testing-library/react/node_modules/@testing-library/dom/dist/query-helpers.js:111:19)
      at Object.<anonymous> (src/App.test.js:6:30)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        2.61 s
Ran all test suites.

Watch Usage: Press w to show more.
```

New Component



```
const Greeting = () => {
  return (
    <div>
      <h2>Hello World</h2>
      <p>It's good to see you!</p>
    </div>
  );
};
export default Greeting;
```

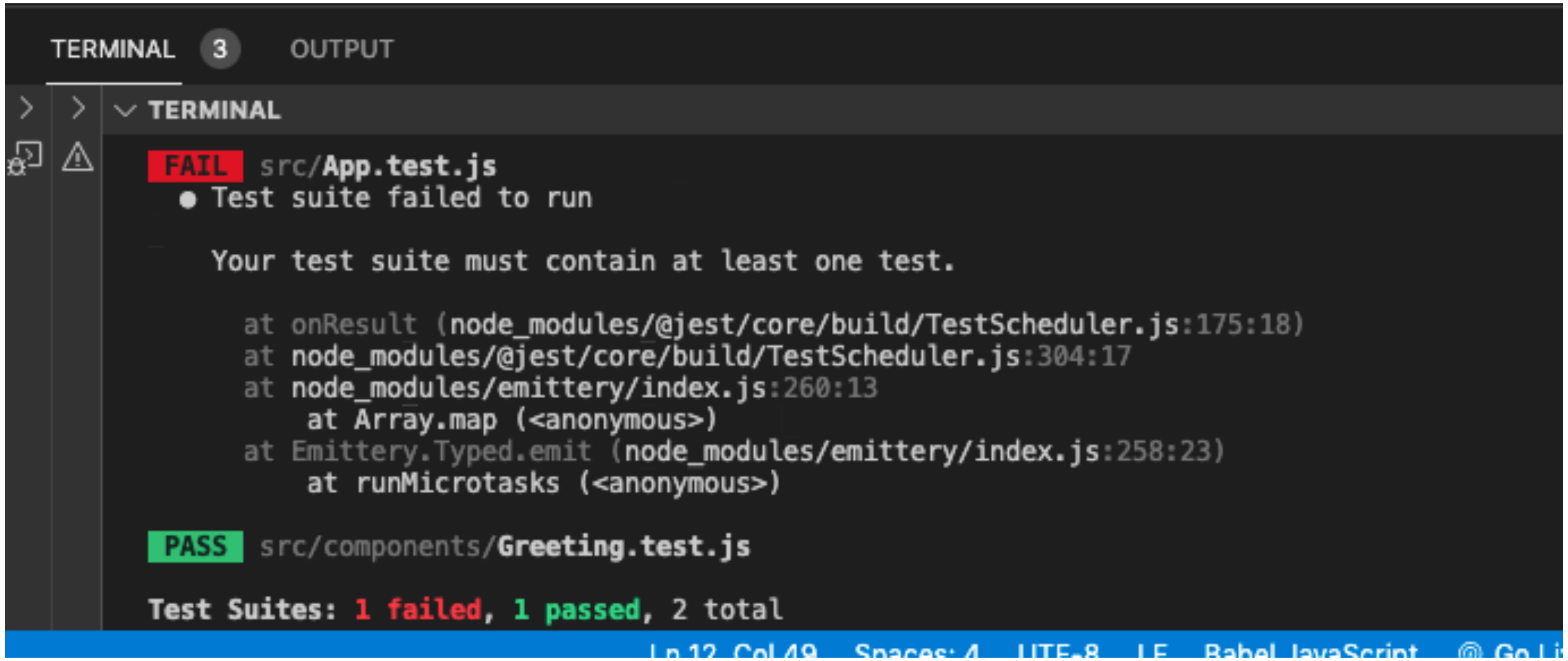


```
import './App.css';
import Greeting from './components/Greeting';

function App() {
  return (
    <div className="App">
      <Greeting />
    </div>
  );
}

export default App;
```

Writing Tests – The Three "A"s

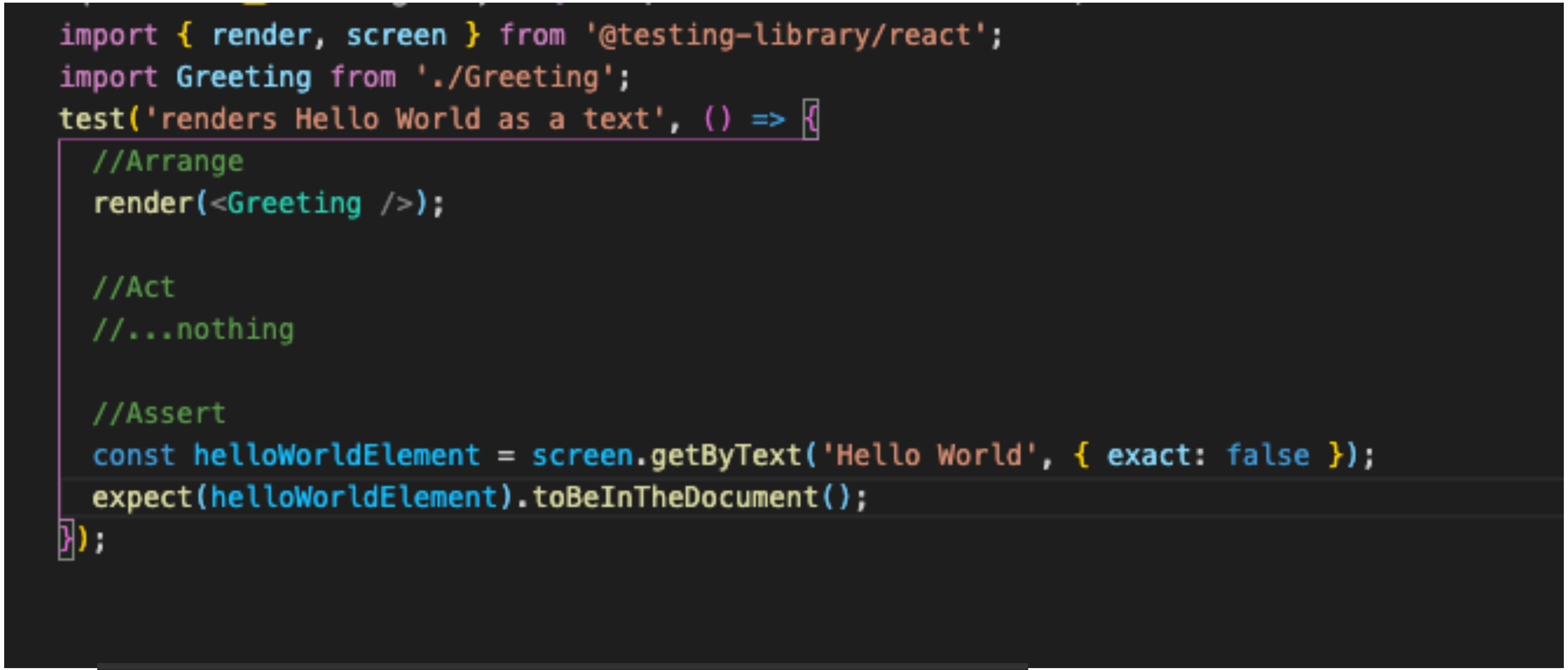
Arrange	Set up the test data, test conditions and test environment
Act	Run logic that should be tested (e.g. execute function)
Assert	Compare execution results with expected results

App.test has no tests

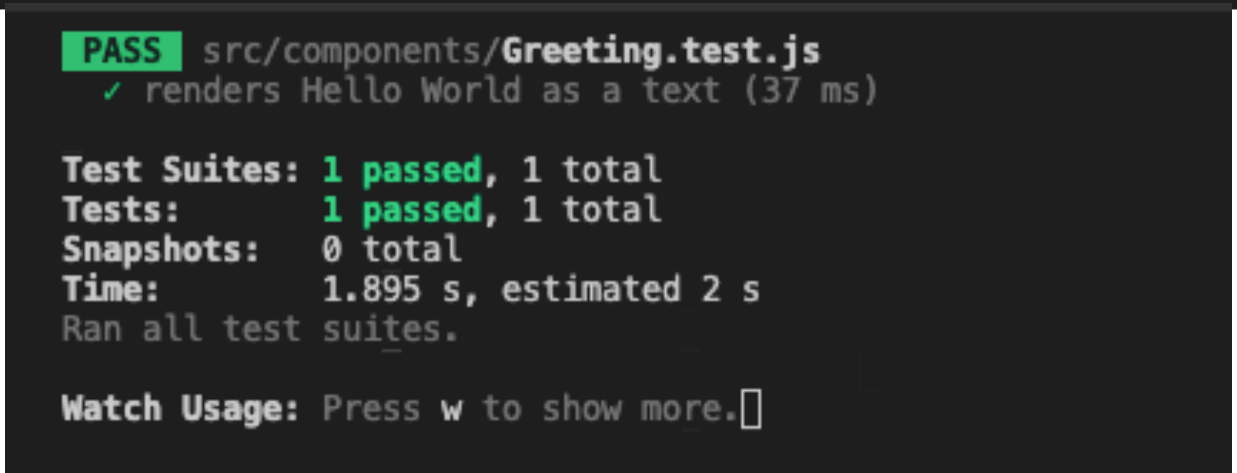


```
FAIL src/App.test.js
  ● Test suite failed to run

    Your test suite must contain at least one test.

      at onResult (node_modules/@jest/core/build/TestScheduler.js:175:18)
      at node_modules/@jest/core/build/TestScheduler.js:304:17
      at node_modules/emittery/index.js:260:13
          at Array.map (<anonymous>)
      at Emittery.Typed.emit (node_modules/emittery/index.js:258:23)
          at runMicrotasks (<anonymous>)

PASS src/components/Greeting.test.js

Test Suites: 1 failed, 1 passed, 2 total
```

Unit Test

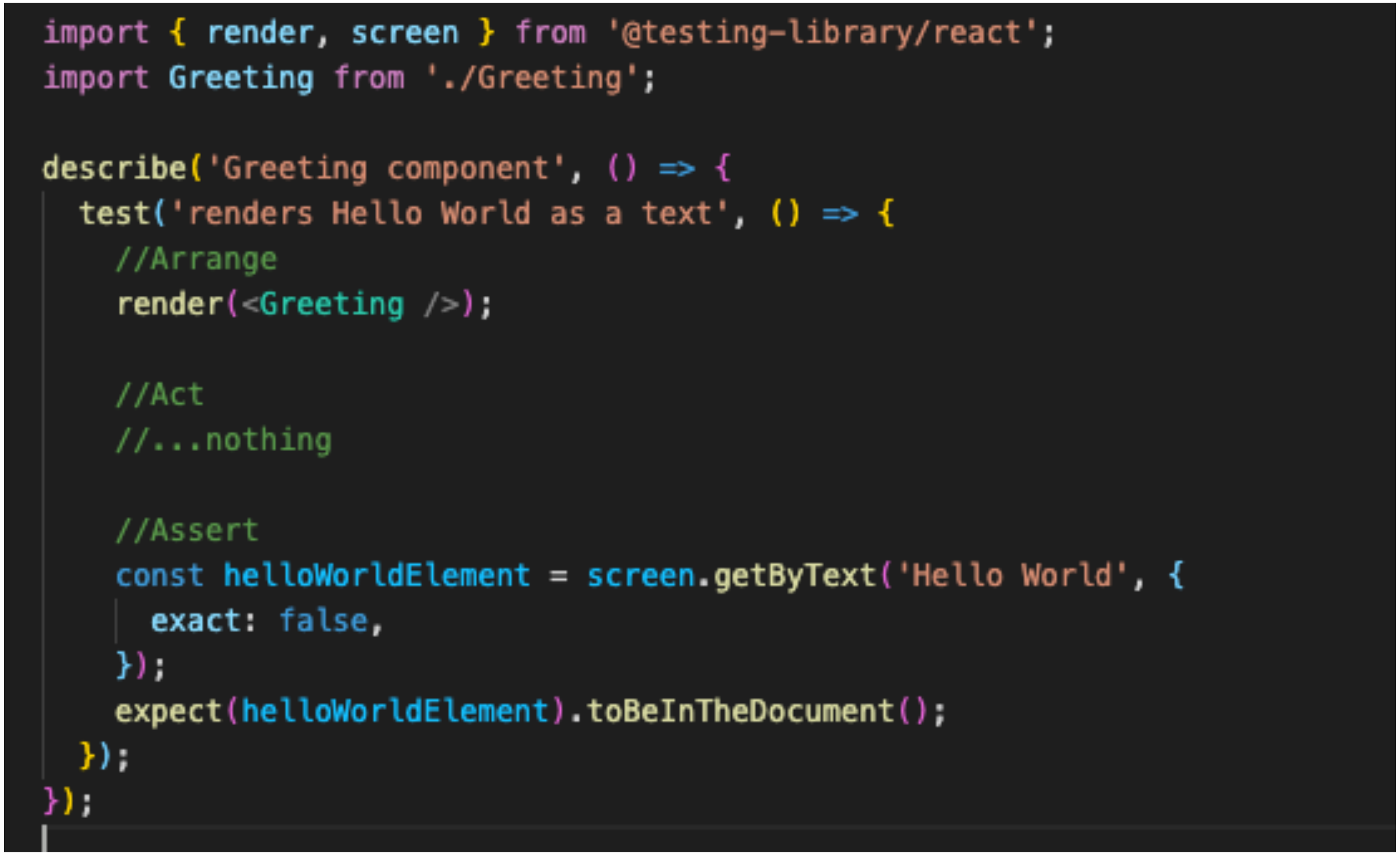


```
import { render, screen } from '@testing-library/react';
import Greeting from './Greeting';
test('renders Hello World as a text', () => {
  //Arrange
  render(<Greeting />);

  //Act
  //...nothing

  //Assert
  const helloWorldElement = screen.getByText('Hello World', { exact: false });
  expect(helloWorldElement).toBeInTheDocument();
});
```



```
 PASS  src/components/Greeting.test.js
  ✓ renders Hello World as a text (37 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.895 s, estimated 2 s
Ran all test suites.

Watch Usage: Press w to show more.
```

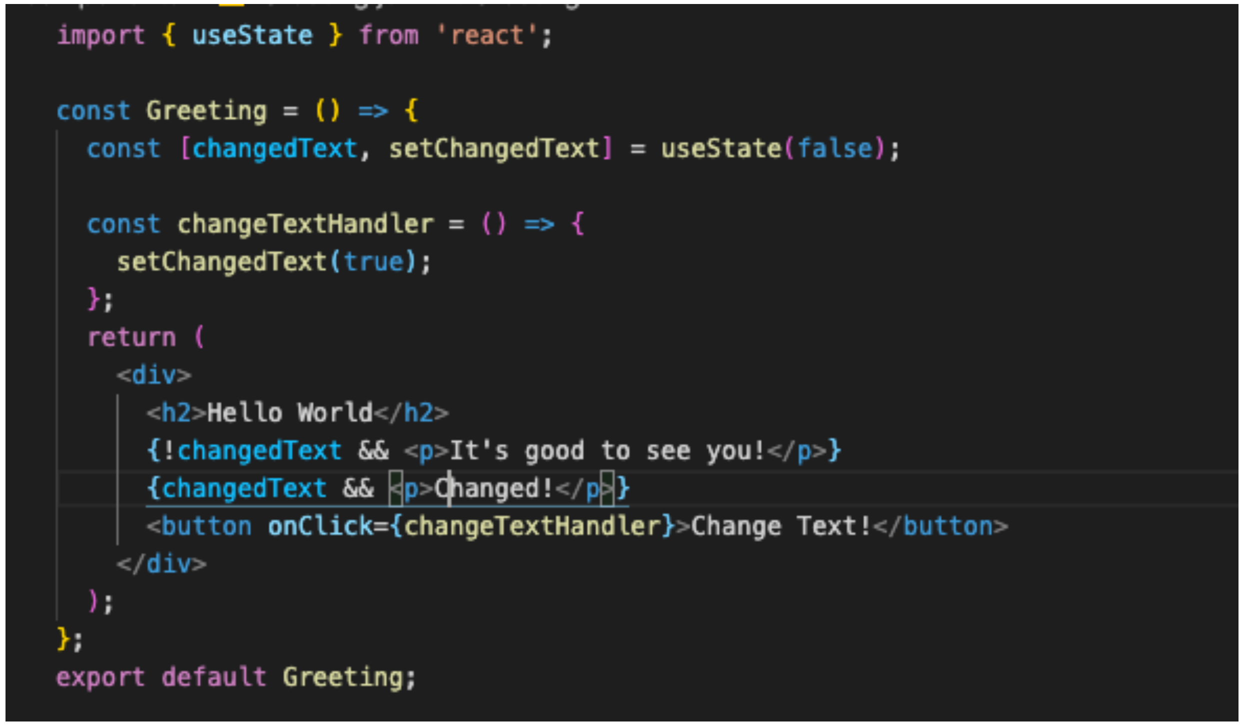
Test Suite



```
import { render, screen } from '@testing-library/react';
import Greeting from './Greeting';

describe('Greeting component', () => {
  test('renders Hello World as a text', () => {
    //Arrange
    render(<Greeting />);

    //Act
    //...nothing

    //Assert
    const helloWorldElement = screen.getByText('Hello World', {
      exact: false,
    });
    expect(helloWorldElement).toBeInTheDocument();
  });
});
```

Testing User Interaction and State

Test all possible scenarios in your test!

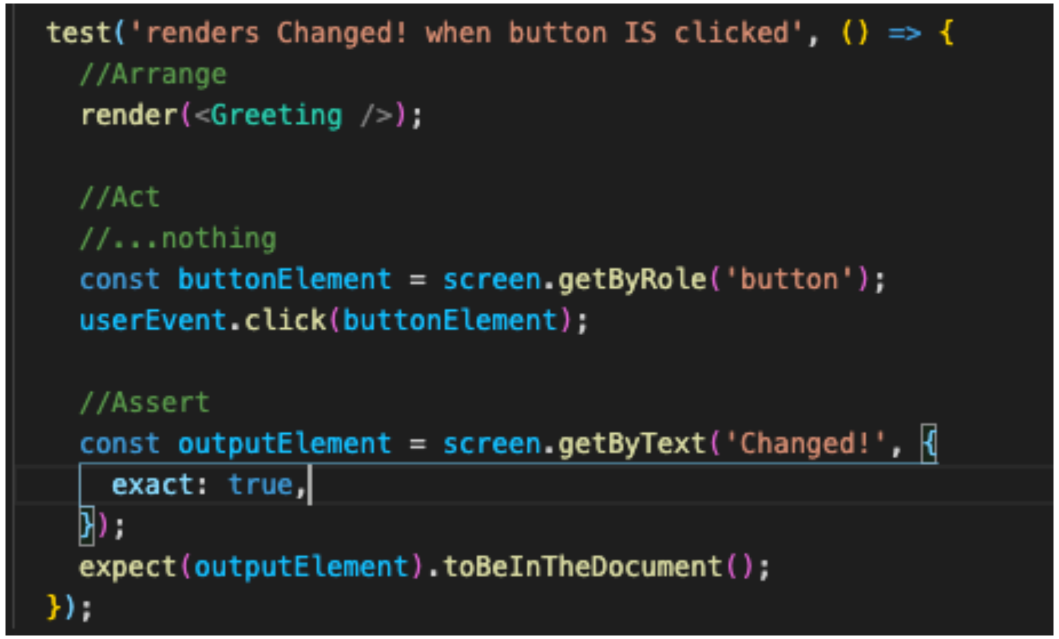


```
import { useState } from 'react';

const Greeting = () => {
  const [changedText, setChangedText] = useState(false);

  const changeTextHandler = () => {
    setChangedText(true);
  };
  return (
    <div>
      <h2>Hello World</h2>
      {!changedText && <p>It's good to see you!</p>}
      {changedText && <p>Changed!</p>}
      <button onClick={changeTextHandler}>Change Text!</button>
    </div>
  );
};
export default Greeting;
```

Testing User Interaction and State

Test all possible scenarios in your test!

userEvent is an object that lets us trigger events by simulating them.

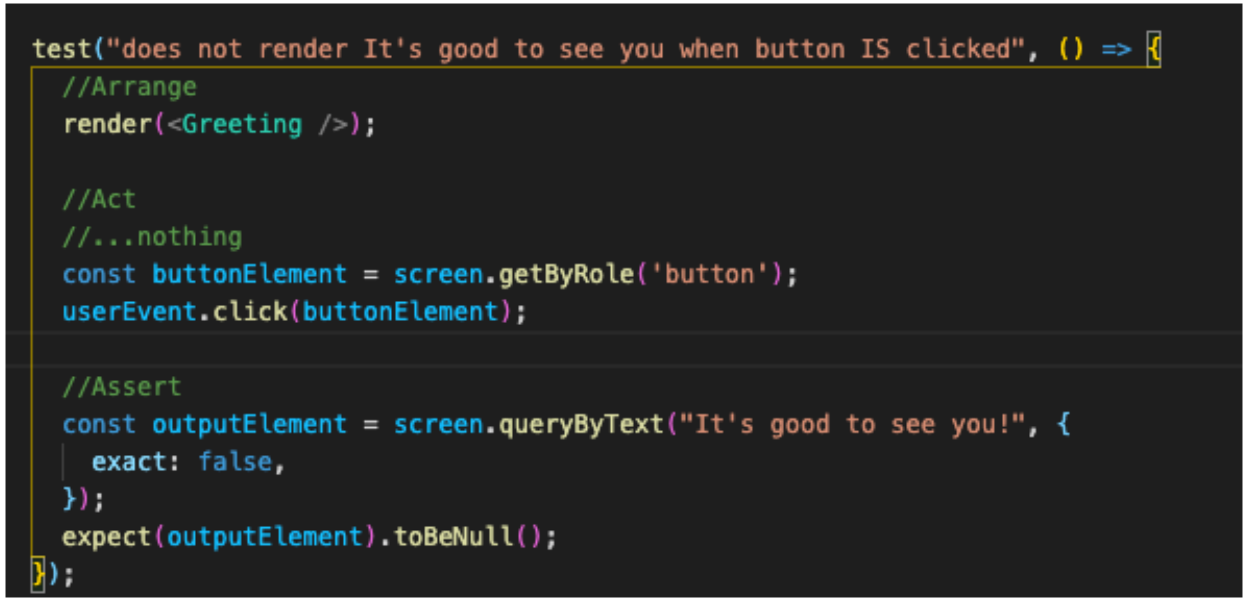


```
test('renders Changed! when button IS clicked', () => {
  //Arrange
  render(<Greeting />);

  //Act
  //...nothing
  const buttonElement = screen.getByRole('button');
  userEvent.click(buttonElement);

  //Assert
  const outputElement = screen.getByText('Changed!', {
    exact: true,
  });
  expect(outputElement).toBeInTheDocument();
});
```

Testing User Interaction and State

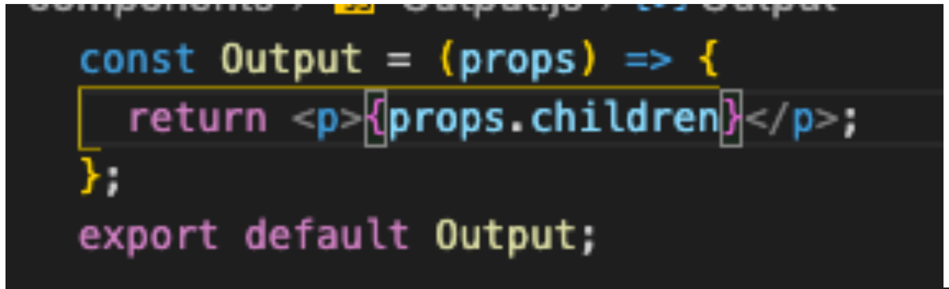
Test all possible scenarios in your test!

Also need to test that the altered text does not show when button is clicked.

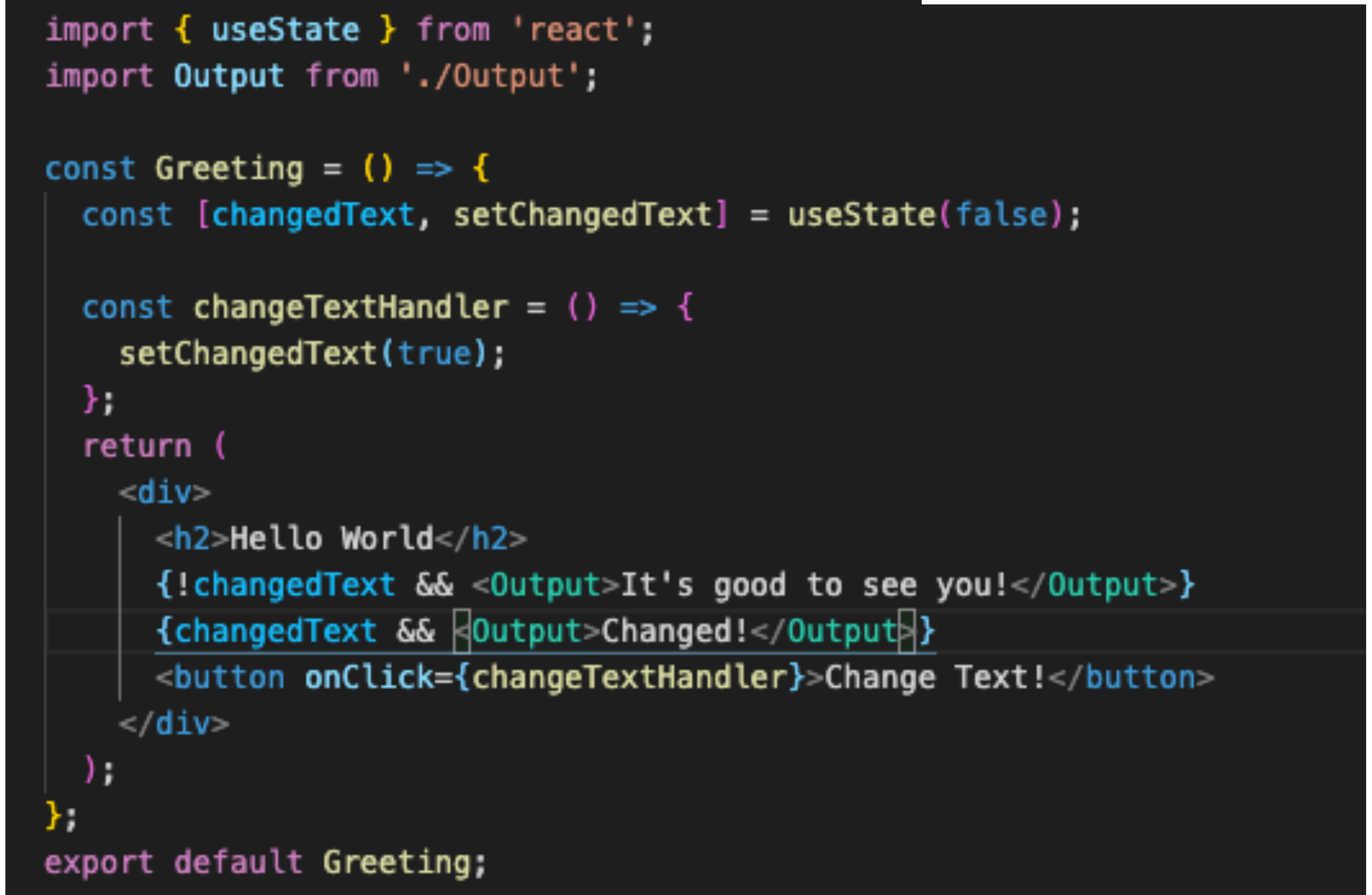


```
test("does not render It's good to see you when button IS clicked", () => {
  //Arrange
  render(<Greeting />);

  //Act
  //...nothing
  const buttonElement = screen.getByRole('button');
  userEvent.click(buttonElement);

  //Assert
  const outputElement = screen.queryByText("It's good to see you!", {
    exact: false,
  });
  expect(outputElement).toBeNull();
});
```

Testing Connected Components

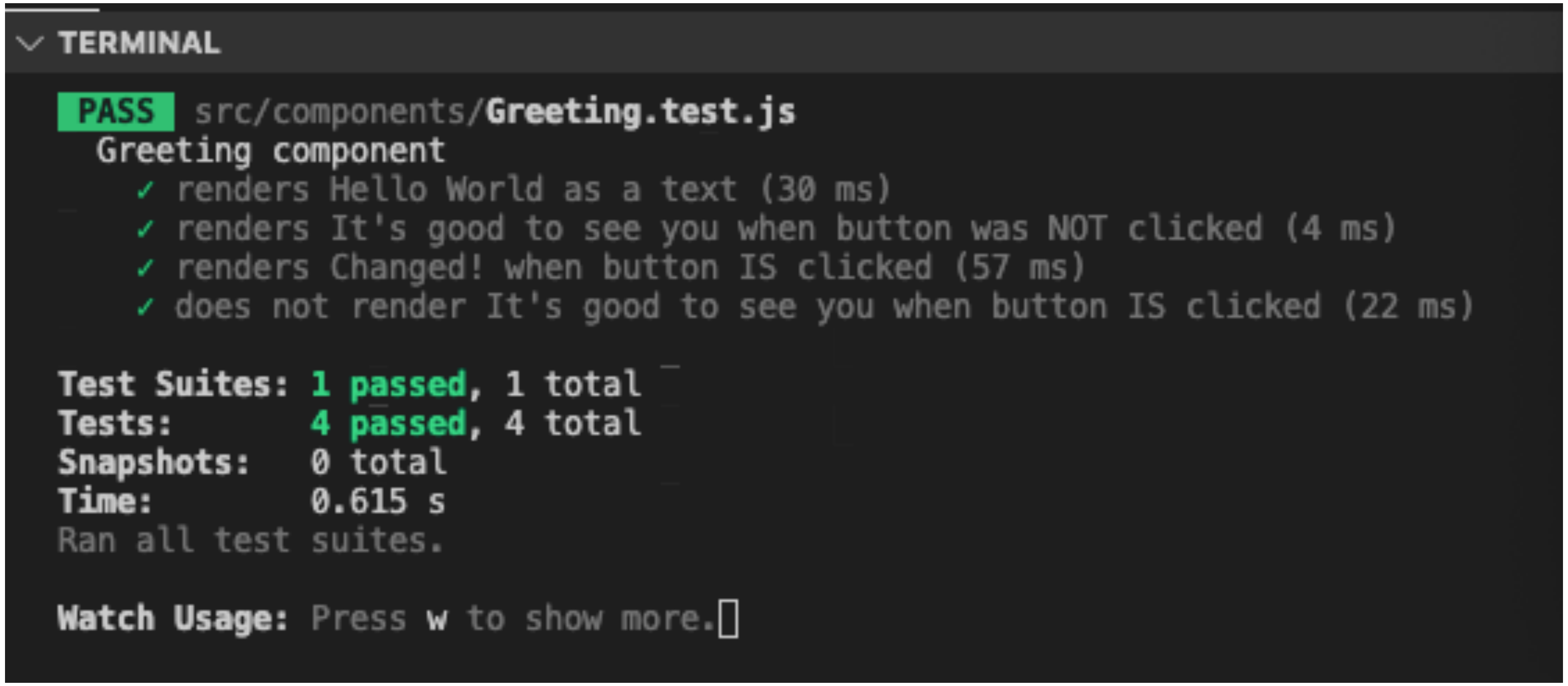


```
const Output = (props) => {
  return <p>{props.children}</p>;
};
export default Output;
```



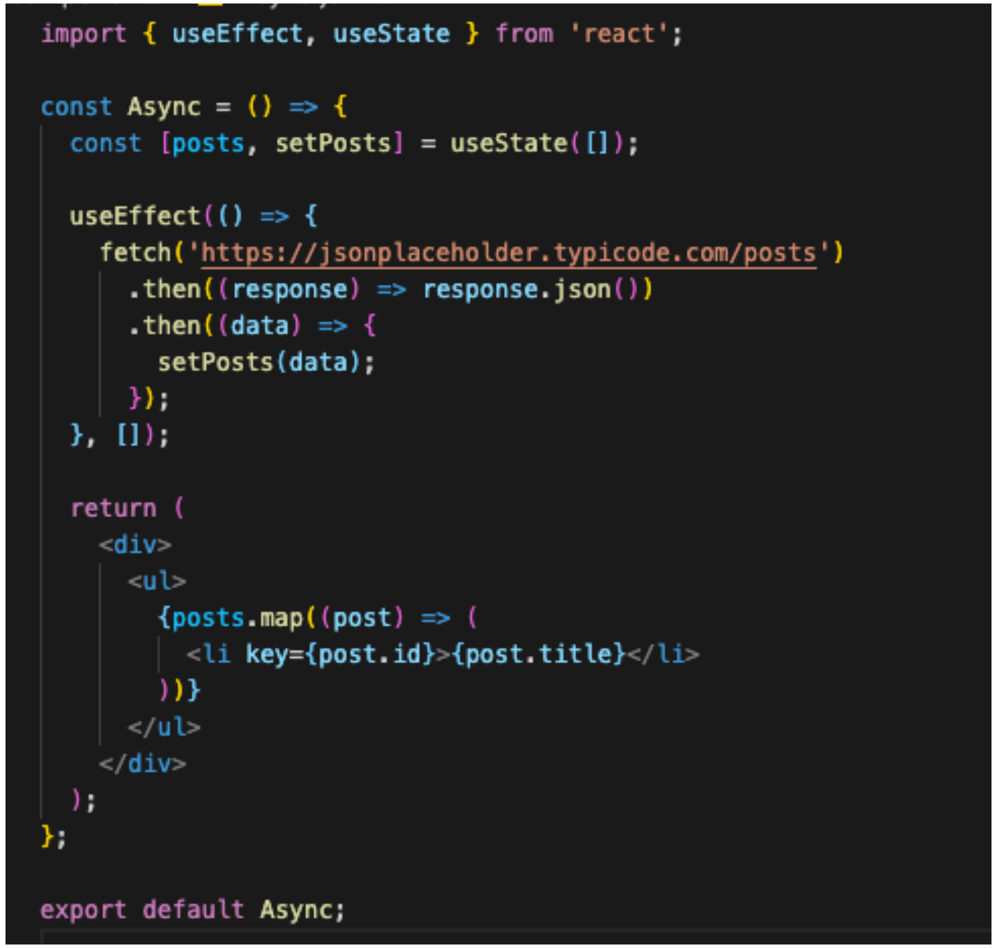
```
import { useState } from 'react';
import Output from './Output';

const Greeting = () => {
  const [changedText, setChangedText] = useState(false);

  const changeTextHandler = () => {
    setChangedText(true);
  };
  return (
    <div>
      <h2>Hello World</h2>
      {!changedText && <Output>It's good to see you!</Output>}
      {changedText && <Output>Changed!</Output>}
      <button onClick={changeTextHandler}>Change Text!</button>
    </div>
  );
};
export default Greeting;
```

Testing Connected Components


TERMINAL

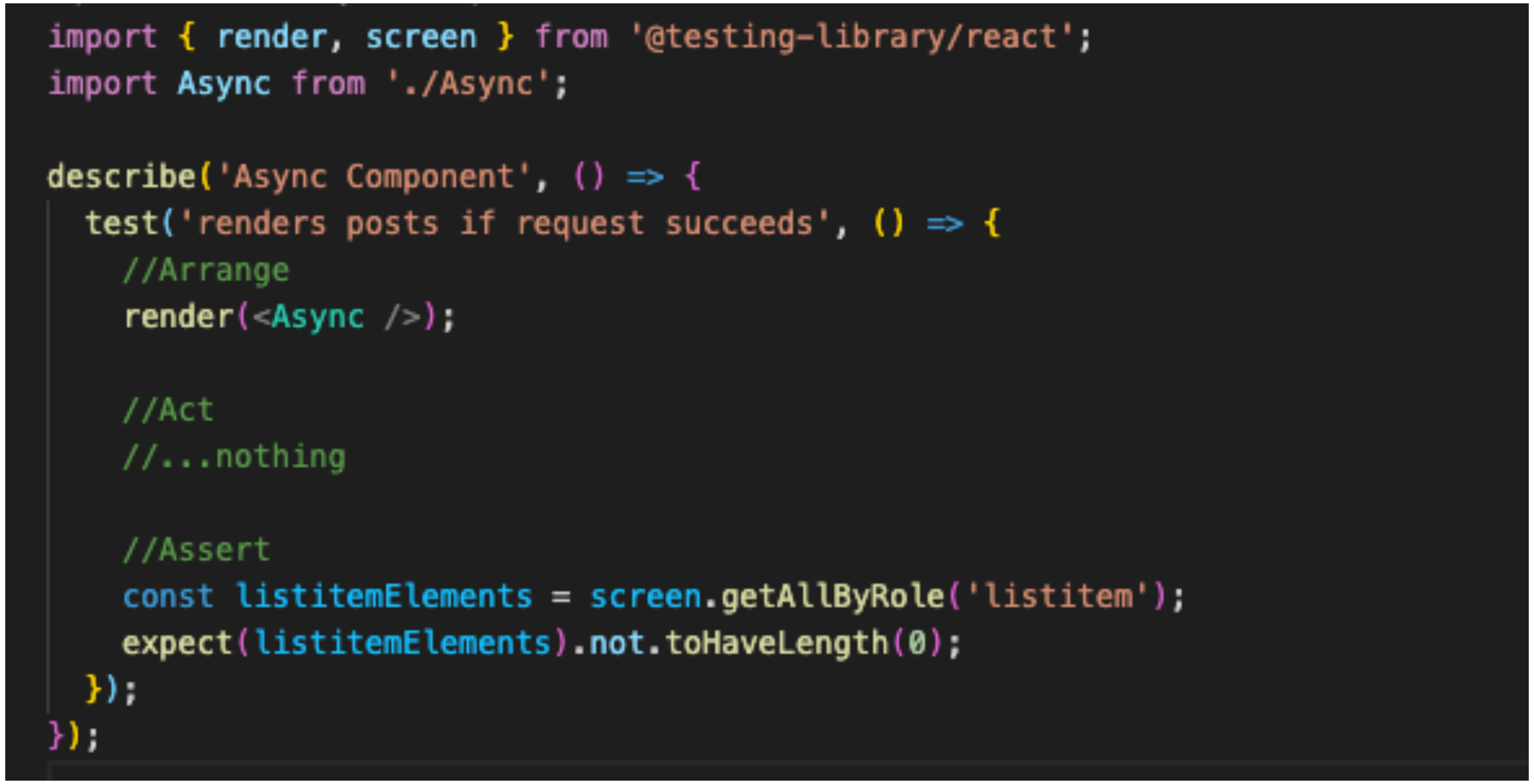
```
 PASS  src/components/Greeting.test.js
  Greeting component
    ✓ renders Hello World as a text (30 ms)
    ✓ renders It's good to see you when button was NOT clicked (4 ms)
    ✓ renders Changed! when button IS clicked (57 ms)
    ✓ does not render It's good to see you when button IS clicked (22 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        0.615 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Testing Async Code



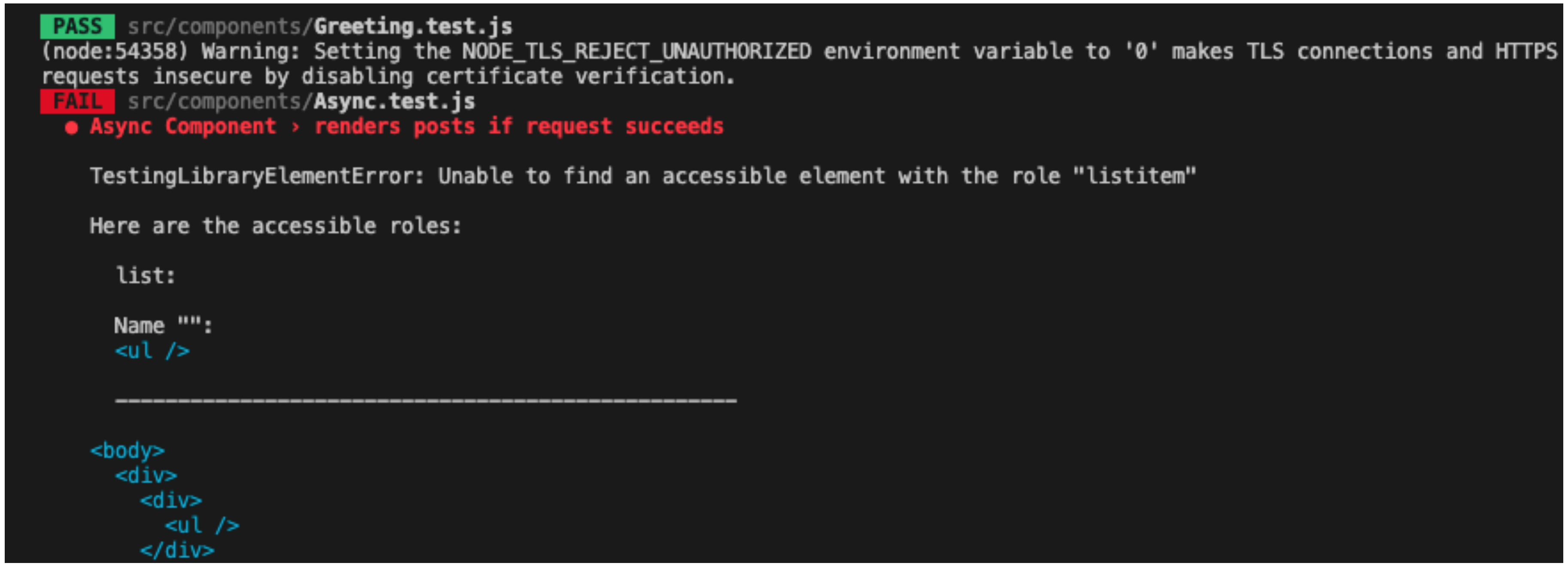
```
import { useEffect, useState } from 'react';

const Async = () => {
  const [posts, setPosts] = useState([]);

  useEffect(() => {
    fetch('https://jsonplaceholder.typicode.com/posts')
      .then((response) => response.json())
      .then((data) => {
        setPosts(data);
      });
  }, []);

  return (
    <div>
      <ul>
        {posts.map((post) => (
          <li key={post.id}>{post.title}</li>
        ))}
      </ul>
    </div>
  );
};

export default Async;
```

Testing Async Code



```
import { render, screen } from '@testing-library/react';
import Async from './Async';

describe('Async Component', () => {
  test('renders posts if request succeeds', () => {
    //Arrange
    render(<Async />);

    //Act
    //...nothing

    //Assert
    const listitemElements = screen.getAllByRole('listitem');
    expect(listitemElements).not.toHaveLength(0);
  });
});
```

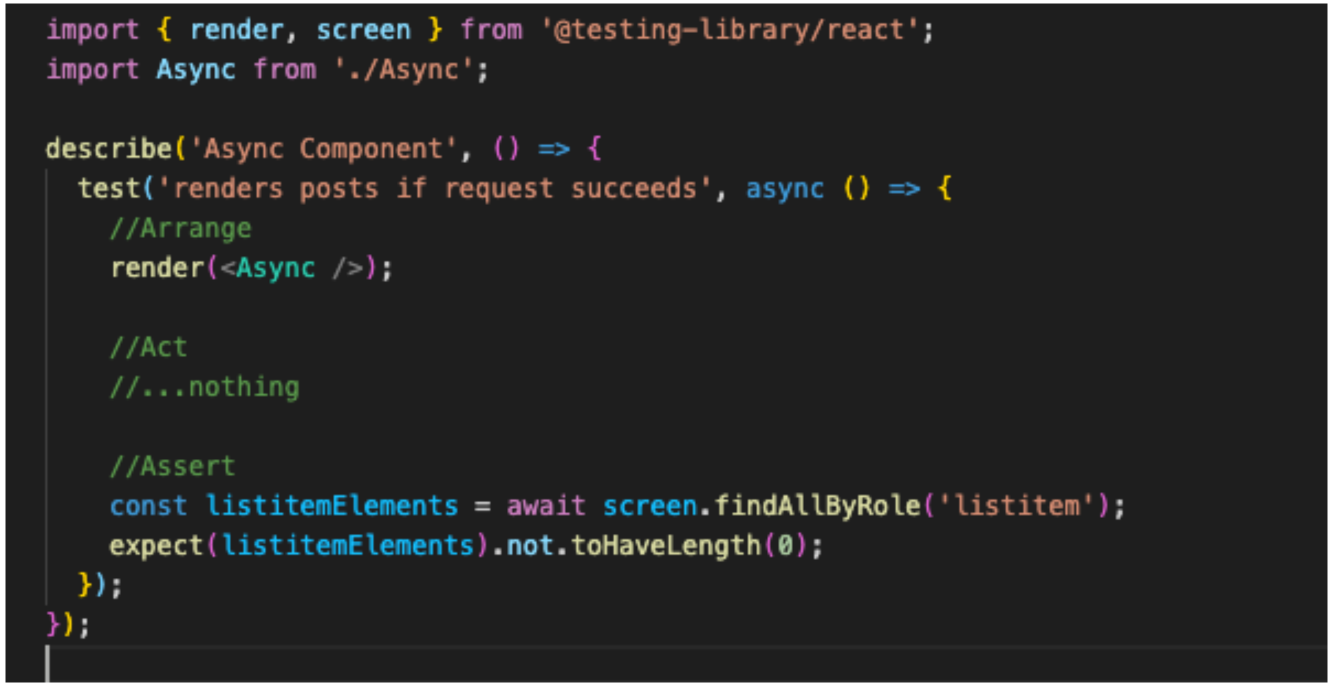
Testing Async Code



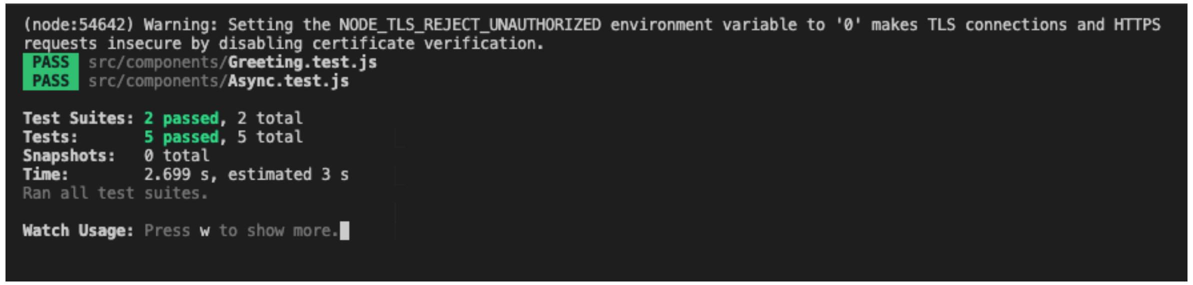
```
 PASS  src/components/Greeting.test.js
(node:54358) Warning: Setting the NODE_TLS_REJECT_UNAUTHORIZED environment variable to '0' makes TLS connections and HTTPS
requests insecure by disabling certificate verification.
 FAIL  src/components/Async.test.js
  ● Async Component › renders posts if request succeeds

    TestingLibraryElementError: Unable to find an accessible element with the role "listitem"

    Here are the accessible roles:

      list:

      Name "":
      <ul />

      --------------------------------------------------

    <body>
      <div>
        <div>
          <ul />
        </div>
```

Initially there are no list items as it’s an async call.
So, we need to alter our tests as fetching data takes time.

Testing Async Code



```
import { render, screen } from '@testing-library/react';
import Async from './Async';

describe('Async Component', () => {
  test('renders posts if request succeeds', async () => {
    //Arrange
    render(<Async />);

    //Act
    //...nothing

    //Assert
    const listitemElements = await screen.findAllByRole('listitem');
    expect(listitemElements).not.toHaveLength(0);
  });
});
```



```
(node:54642) Warning: Setting the NODE_TLS_REJECT_UNAUTHORIZED environment variable to '0' makes TLS connections and HTTPS
requests insecure by disabling certificate verification.
 PASS  src/components/Greeting.test.js
 PASS  src/components/Async.test.js

Test Suites: 2 passed, 2 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        2.699 s, estimated 3 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Testing Async Code

Not ideal as we don’t want to send http requests to test out Async code.
We don’t want to create network traffic
Don’t want to insert data into database etc. for unit testing
We don’t want to send requests to server to add data.

Fake server or don’t send request as two options for testing.
For example, we don’t want to unit test code we haven’t written.

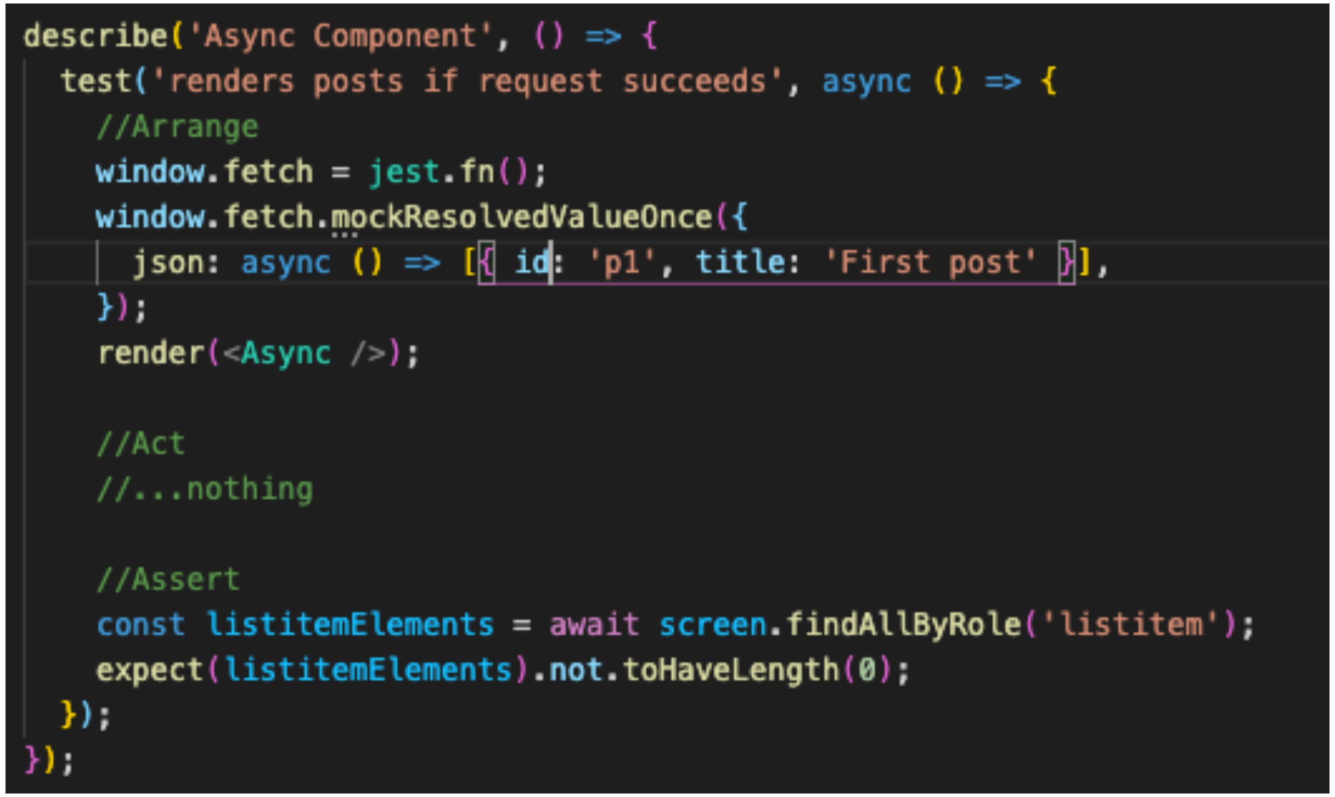
We want to test whether our component behaves correctly when I get data/ has an error.

We can create a dummy function (mock) instead of built-in fetch function.

Very common scenario where we are working with local Storage as well.
Jest has built in support for mocking as well.

Testing Async Code

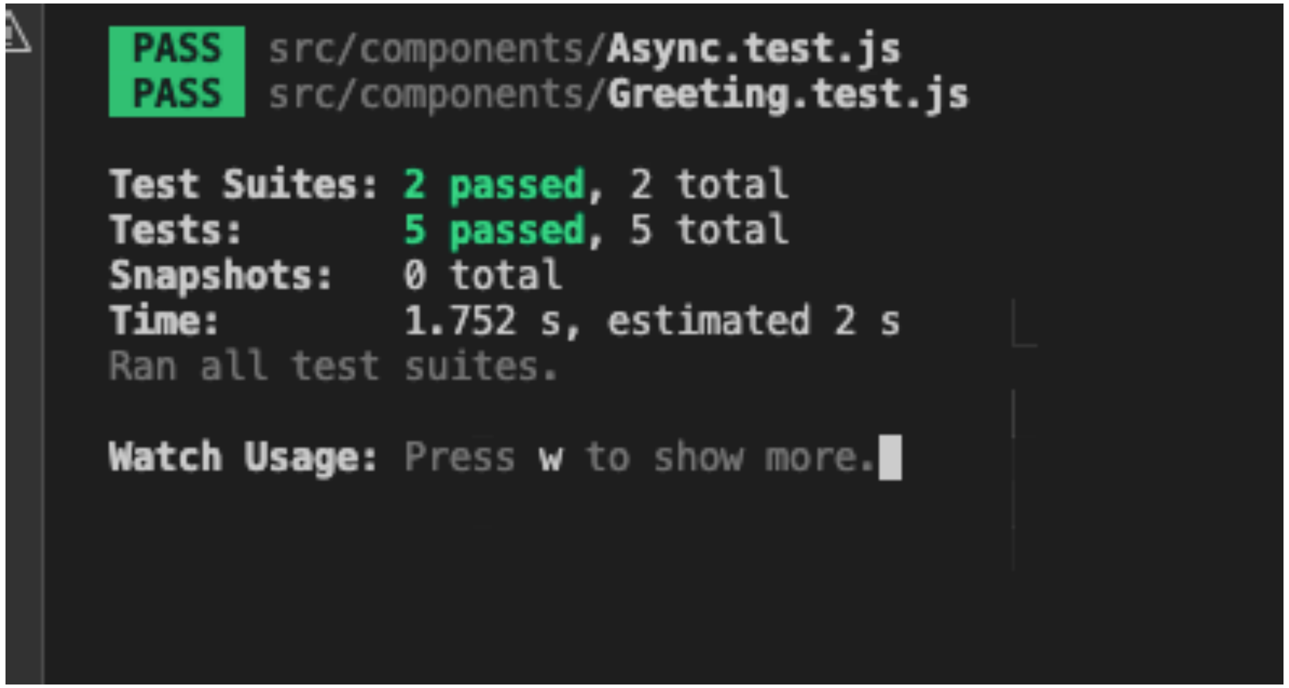
jest.fn() creates a mock function.



```
describe('Async Component', () => {
  test('renders posts if request succeeds', async () => {
    //Arrange
    window.fetch = jest.fn();
    window.fetch.mockResolvedValueOnce({
      json: async () => [{ id: 'p1', title: 'First post' }],
    });
    render(<Async />);

    //Act
    //...nothing

    //Assert
    const listitemElements = await screen.findAllByRole('listitem');
    expect(listitemElements).not.toHaveLength(0);
  });
});
```

Testing Async Code

jest.fn() creates a mock function.



```
PASS  src/components/Async.test.js
PASS  src/components/Greeting.test.js

Test Suites: 2 passed, 2 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        1.752 s, estimated 2 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Summary

https://jestjs.io/

https://jestjs.io/docs/getting-started

https://testing-library.com/docs/react-testing-library/intro/