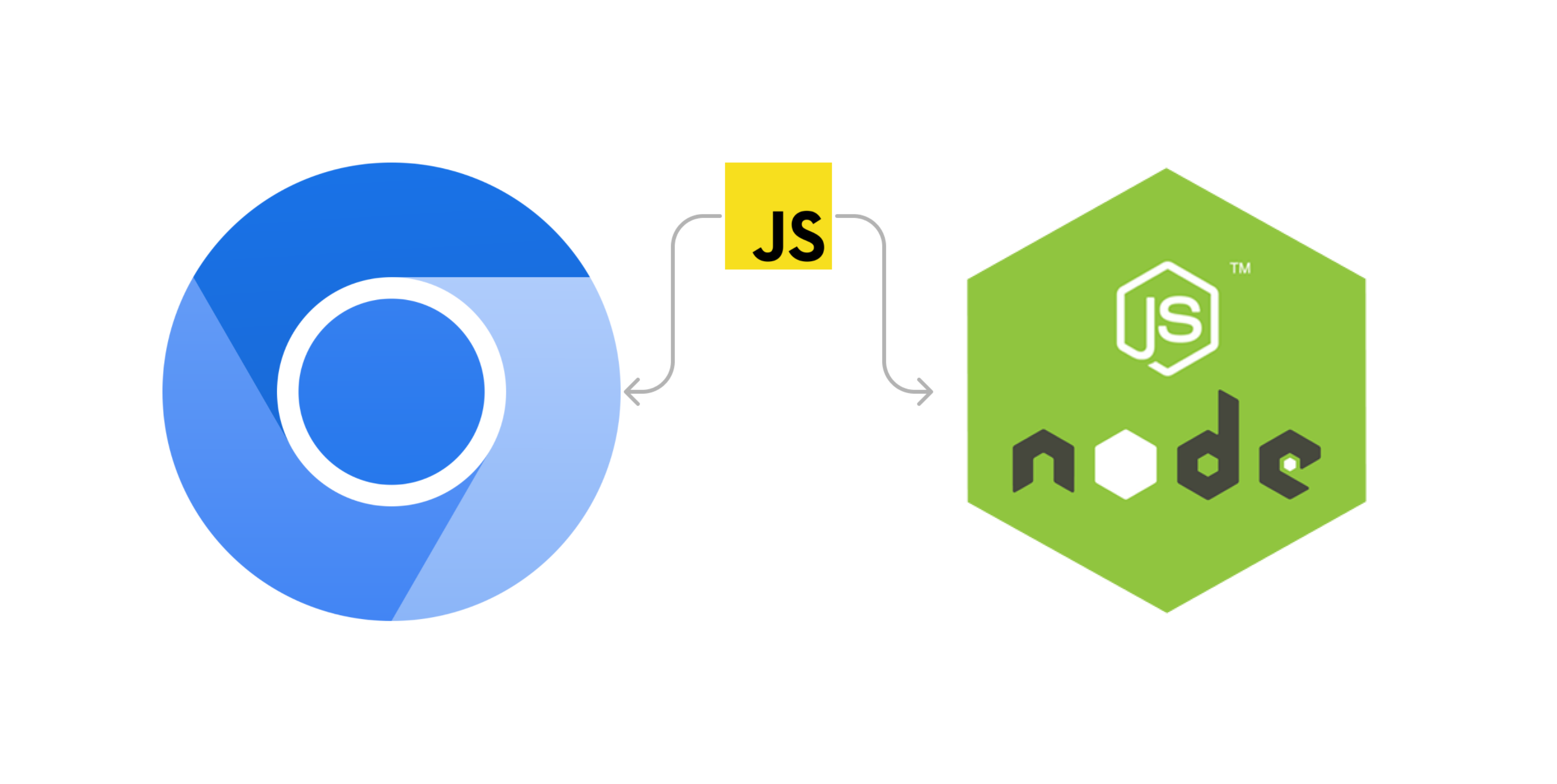
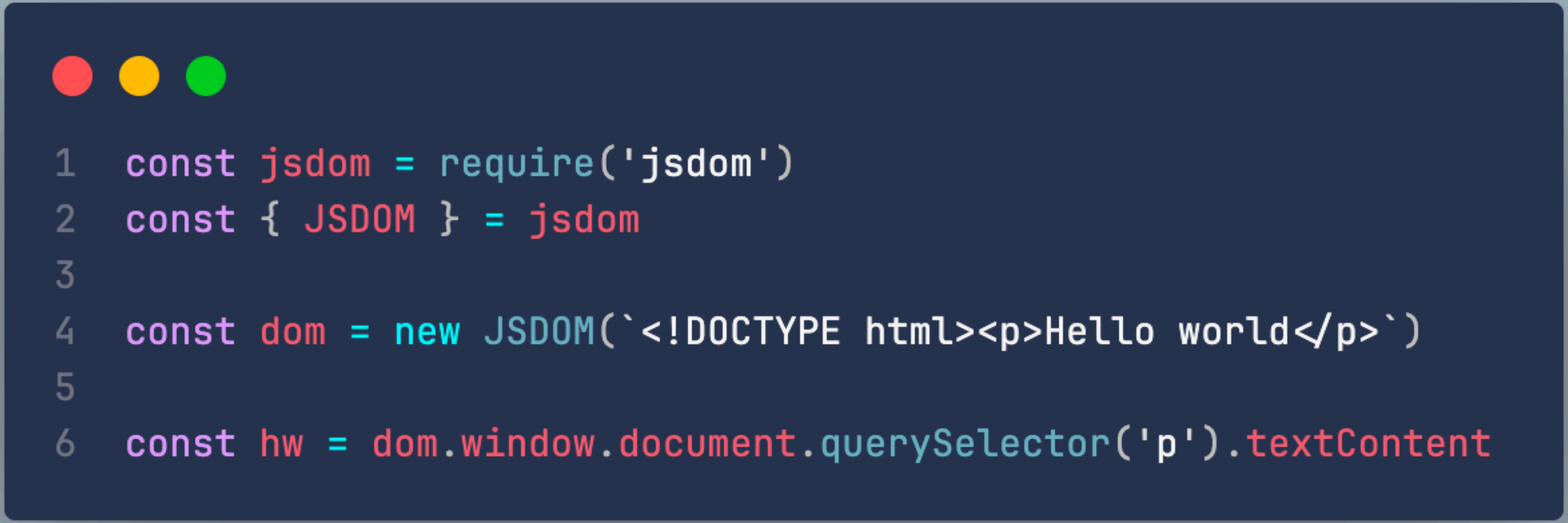
REACT TESTING

Example 4 (react setup)

Example 4 (react setup)



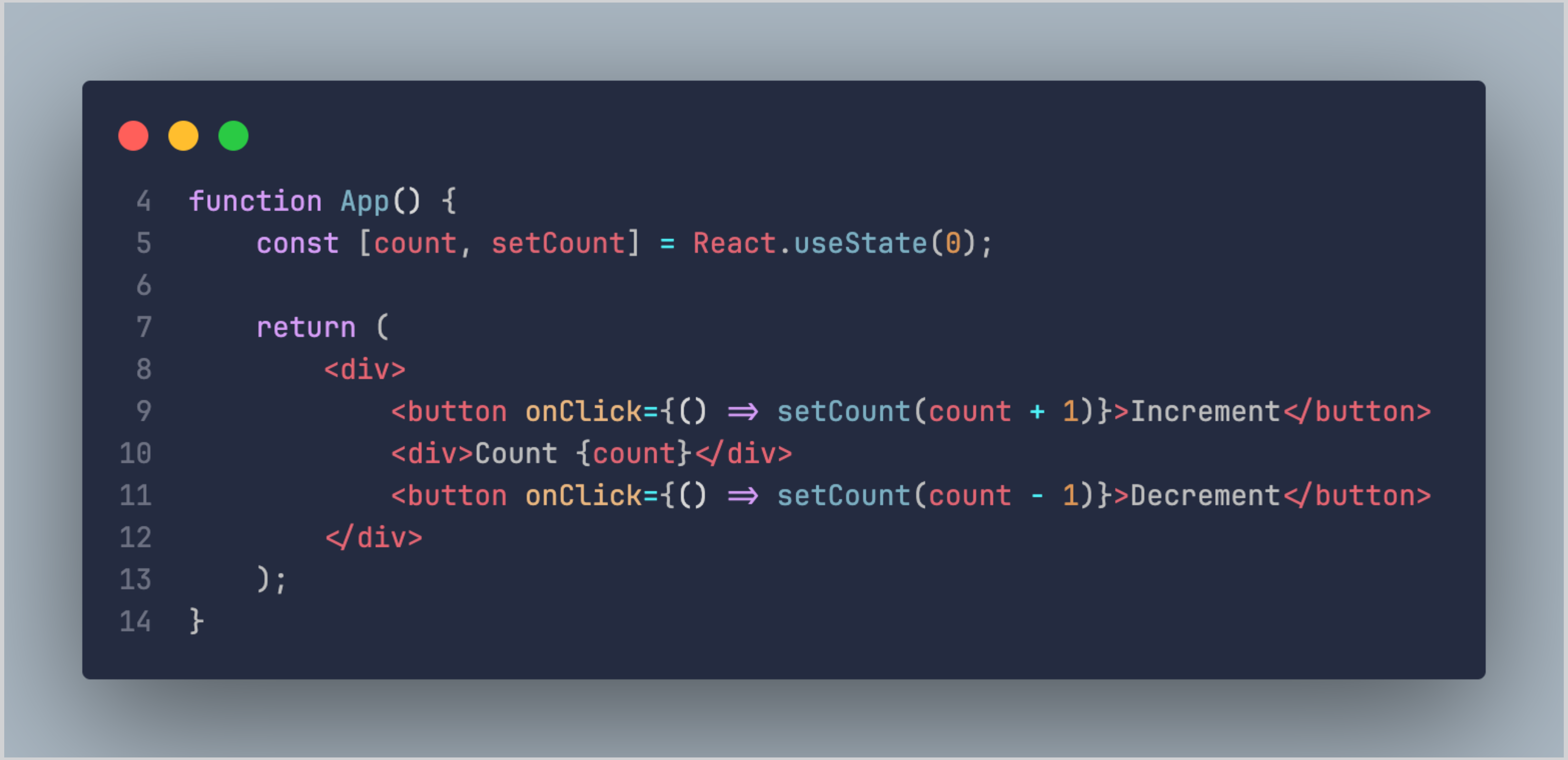


```
const jsdom = require('jsdom')
const { JSDOM } = jsdom

const dom = new JSDOM(`<!DOCTYPE html><p>Hello world</p>`)

const hw = dom.window.document.querySelector('p').textContent
```

React Component testing flow

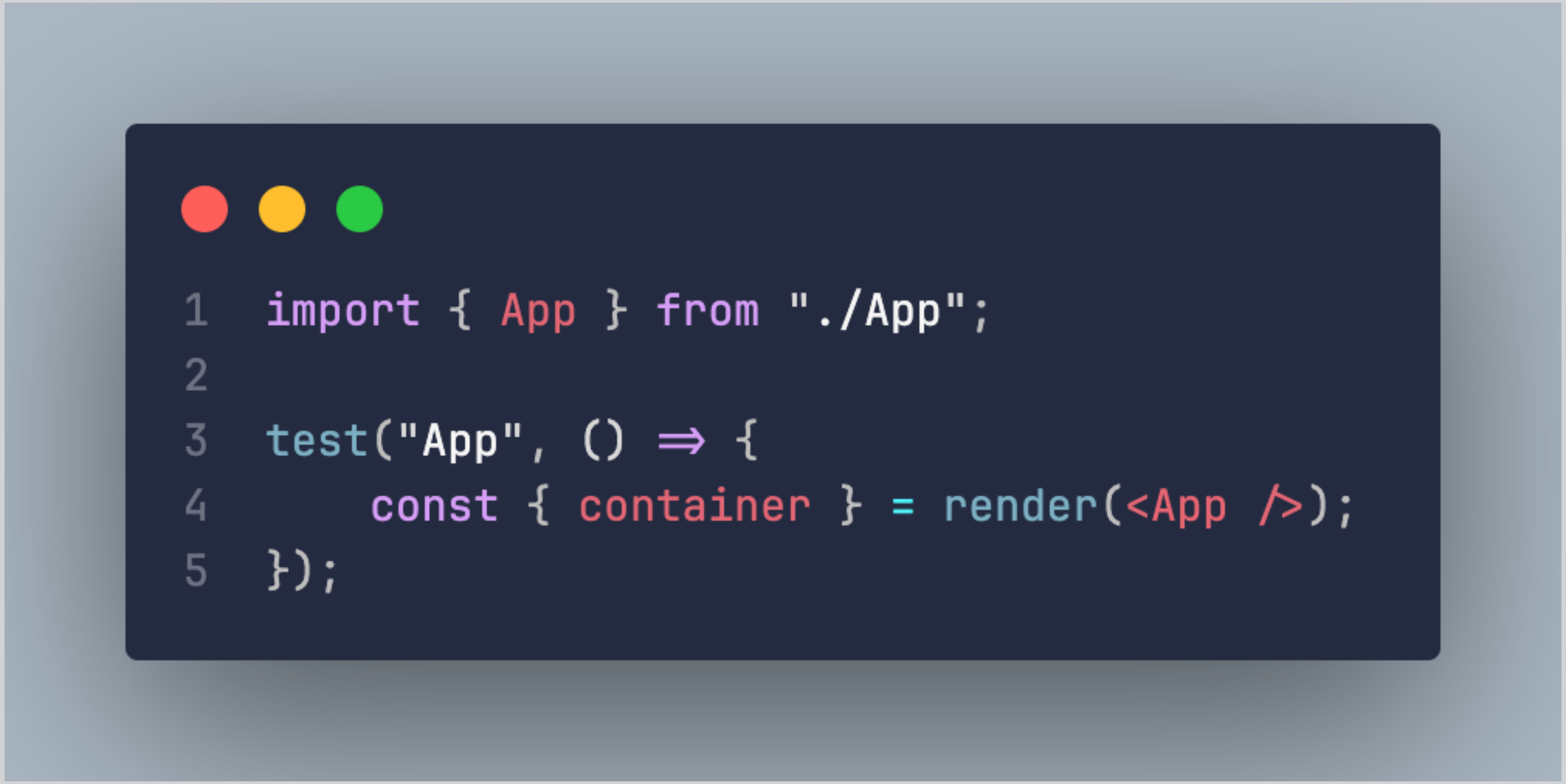
- render
- query an element
- fireEvent (“user-event”)
- *act
- waitFor
- match

React Component testing flow



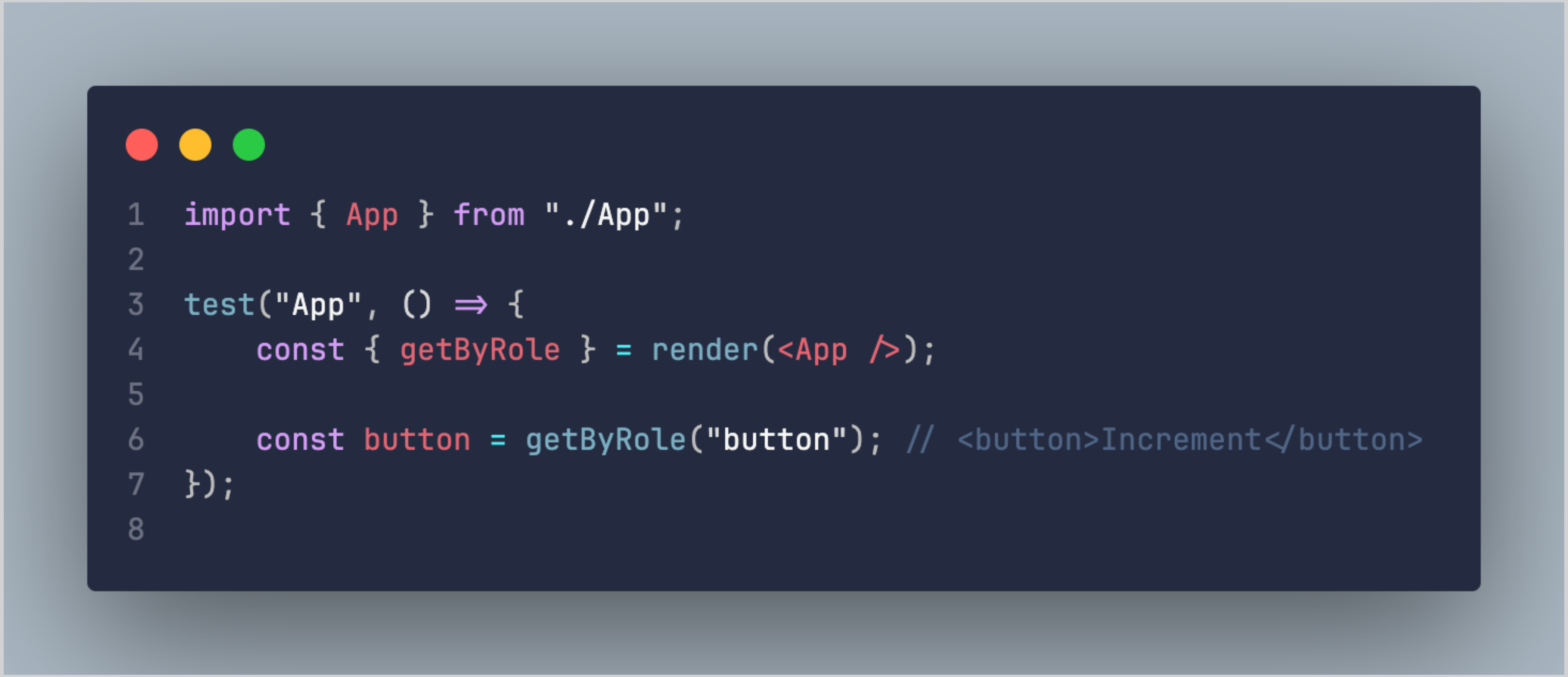
```
function App() {
    const [count, setCount] = React.useState(0);

    return (
        <div>
            <button onClick={() => setCount(count + 1)}>Increment</button>
            <div>Count {count}</div>
            <button onClick={() => setCount(count - 1)}>Decrement</button>
        </div>
    );
}
```

Render



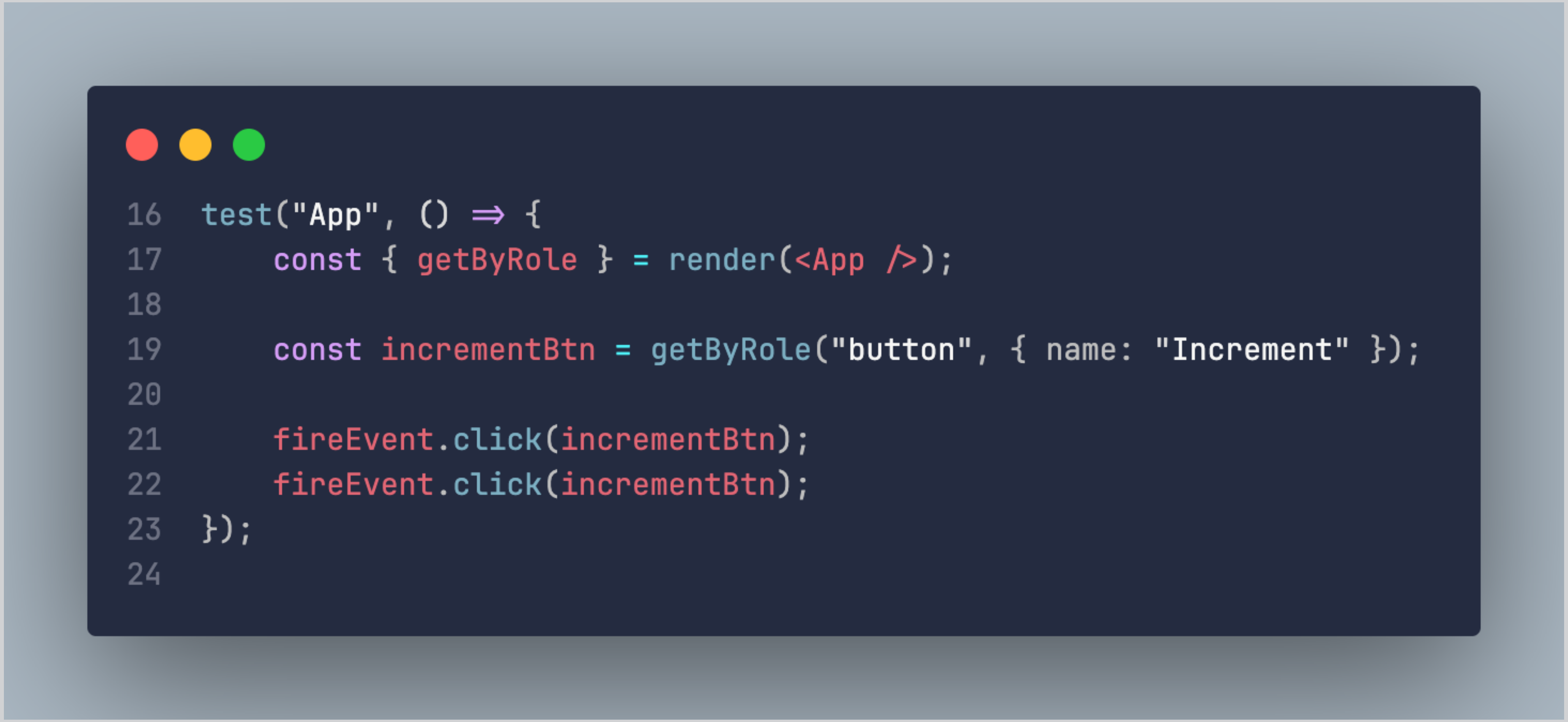
```
import { App } from "./App";

test("App", () => {
    const { container } = render(<App />);
});
```

Query an Element



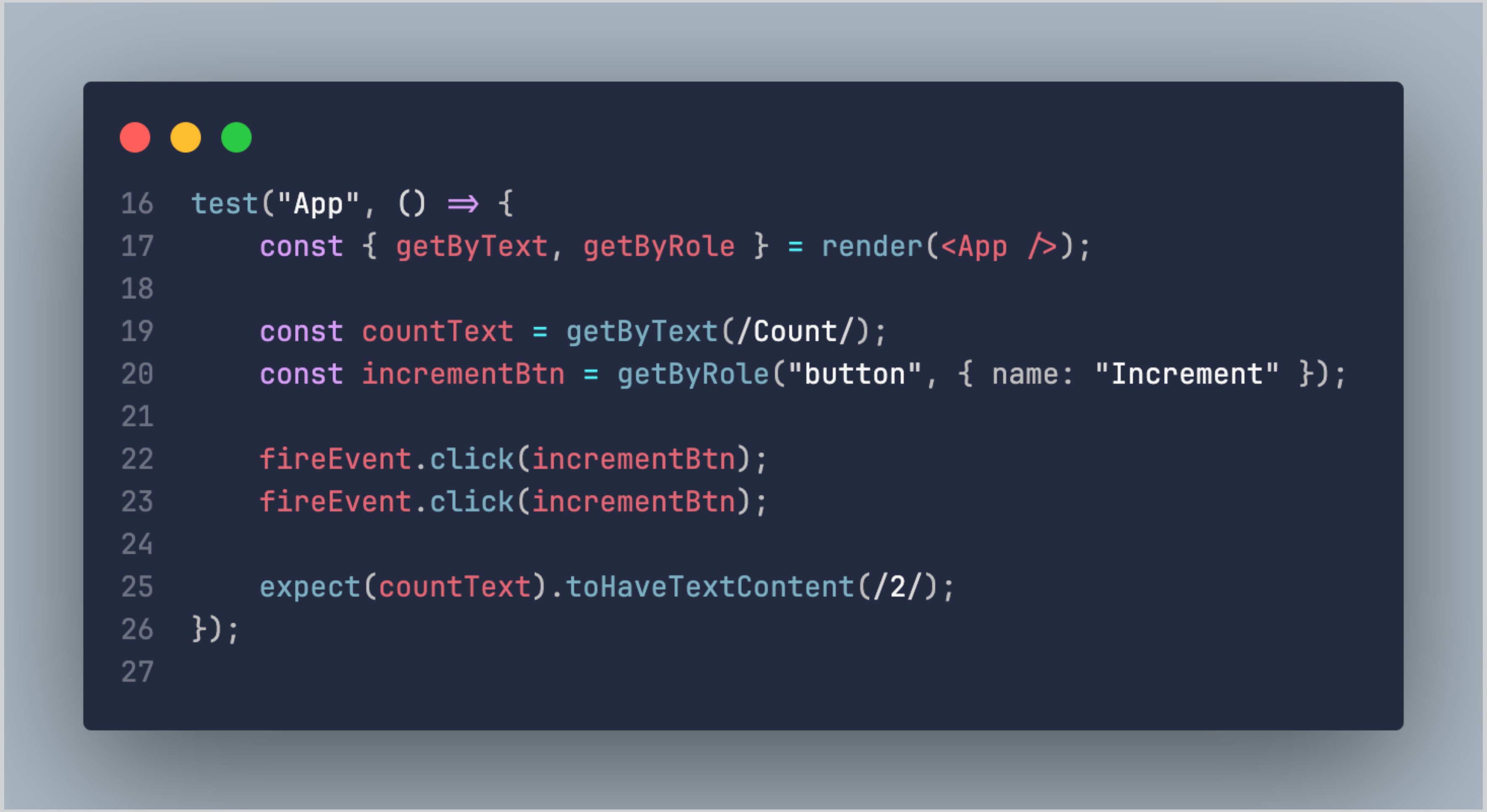
```
import { App } from "./App";

test("App", () => {
    const { getByRole } = render(<App />);

    const button = getByRole("button"); // <button>Increment</button>
});

```

Fire an event



```
test("App", () => {
    const { getByRole } = render(<App />);

    const incrementBtn = getByRole("button", { name: "Increment" });

    fireEvent.click(incrementBtn);
    fireEvent.click(incrementBtn);
});
```

Match the Result



```
test("App", () => {
    const { getByText, getByRole } = render(<App />);

    const countText = getByText(/Count/);
    const incrementBtn = getByRole("button", { name: "Increment" });

    fireEvent.click(incrementBtn);
    fireEvent.click(incrementBtn);

    expect(countText).toHaveTextContent(/2/);
});
```

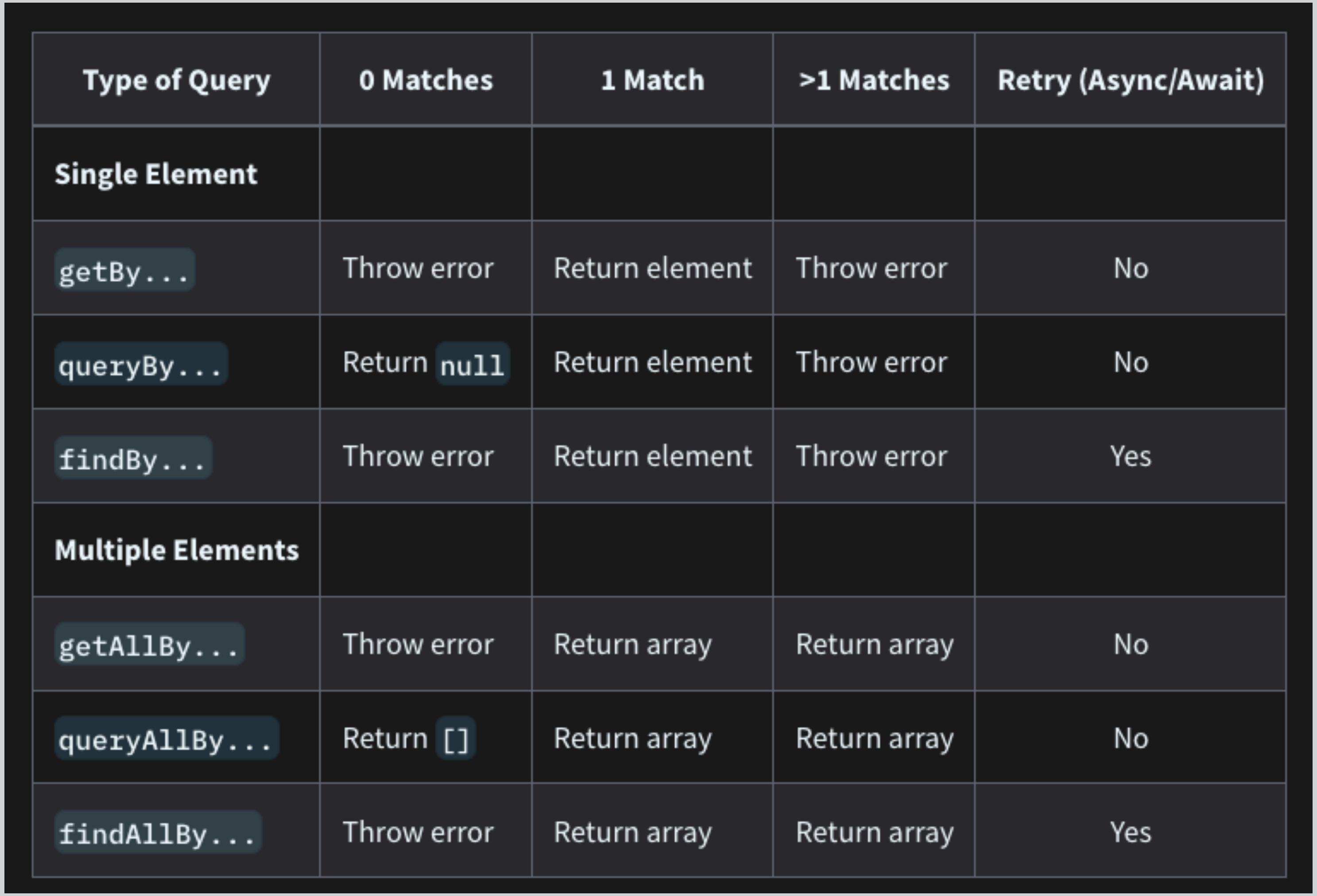
Querying

- getBy, queryBy, findBy
- role, label, placeholder, text,

Example 4 (react setup)

- getBy - fails with message if the element is not found or if multiple elements are found
- queryBy - returns null if the element is not found, does not fail
- findBy - rejects a Promise if there are multiple elements or element isn’t found within default 1000ms

Example 4 (react setup)



Type of Query	0 Matches	1 Match	>1 Matches	Retry (Async/Await)
Single Element				
`getBy...`	Throw error	Return element	Throw error	No
`queryBy...`	Return `null`	Return element	Throw error	No
`findBy...`	Throw error	Return element	Throw error	Yes
Multiple Elements				
`getAllBy...`	Throw error	Return array	Return array	No
`queryAllBy...`	Return `[]`	Return array	Return array	No
`findAllBy...`	Throw error	Return array	Return array	Yes

Roles

- heading - <h1>
- link - <a href=“…”>
- img - <img>
- button - <button>
- list - <ul> <ol>
- listitem - <li>
- checkbox - <input type=“checkbox”>
- …others

Why not query class?

- Class names are an implementation detail that will make your unit tests less robust.
- (className changes - test changes)

act()

- You don’t need to use act(), most of the cases are already wrapped in act() (“user-event”)

act()

- You don’t need to use act(), most of the cases are already wrapped in act() (“user-event”)
- Most common use case for act is jest’s fake timers.

act()

- You don’t need to use act(), most of the cases are already wrapped in act() (“user-event”)
- Most common use case for act is jest’s fake timers.
- Dispatching actions is another way of using act

Example 5 (Testing Redux)

1. Create store
2. Dispatch some actions
3. Verify the state
4. Don’t use mocked store, us your own real store! (If possible)

Example 5 (Testing React with Redux)

1. Initialise store
2. Wrap your component with Provider
3. Dispatch some actions
4. Verify the state is correct
5. Verify the component shows correct things

- https://github.com/threepointone/react-act-examples/blob/master/sync.md
- https://bestofjs.org/
- https://jestjs.io/
- https://vitest.dev/
- https://testing-library.com/
- https://www.youtube.com/watch?v=Bf7vDBBOBUA
- https://www.youtube.com/watch?v=5iJWOPaNZDA&t=1731s
- https://www.youtube.com/watch?v=Eu35xM76kKY
- https://en.wikipedia.org/wiki/Inattentional_blindness
- https://en.wikipedia.org/wiki/Test-driven_development
- https://dev.to/thejaredwilcurt/why-you-should-never-use-tobe-in-jest-48ca
- https://www.w3.org/TR/html-aria/#docconformance

Evolution Engineering | ACCELERATING EVOLUTION