React Testing Library

github.com/test-library/react-testing-library

Simple and complete cheat sheet v2

render a component

```
import { render } from '@testing-library/react'

// See render result for more details…

const result = render(<MyComponent />)
```

search the DOM

```
import { screen } from '@testing-library/react'

render(<span>hello</span>)

// Check out jest-dom on npm for handy assertions…

const element = screen.getByText('hello')
```

fire an event

```
import { fireEvent } from '@testing-library/react'

fireEvent.click(element)

// Or you can fire events manually with…
fireEvent(element, new MouseEvent('click')
```

interact with element

```
import UserEvent from '@testing-library/user-event'
// UserEvent simulates advanced browser interactions like
// clicks, type, uploads, tabbing etc


// Click on a button
UserEvent.click(screen.getByRole('button'))

// Types HelloWorld in a text field
UserEvent.type('Hello World',screen.getByRole('textbox')
```

wait for something

```
import { screen, waitFor } from '@testing-library/react'

// Retry search every 50ms for 4500ms

await waitFor(() => screen.getByText('async result'))
```

search variants (return value)

getBy	Element or Error
getAllBy	Element[] or Error
queryBy	Element or Error
queryAllBy	Element[] or []
findBy	Promise<Element> or Error
findAllBy	Promise<Element[]> or Error

(DOM attribute) search types

Role	...
LabelText	<label for="element" />
PlaceholderText	<input placeholder="username" />
Text	About
DisplayValue	Current value of input element
AltText	
Title	or <title />
TestId	<input data-testid="username-input" />

text matches

```
render(<div>Hello World</div>

screeen.getByText(/hello world/)                    // ❌
screen.getByText(/hello world/i)                    // ✅
screen.getByText('Hello World')                     // ✅

screen.getByTestId('username')                      // ❌
screen.getByRole('textbox', {name: /username/i})    // ✅
```

wait for appearance

```
test('movie title appears', async () => {
  // element is initially not present...

  // wait for appearance
  await waitFor(() => {
    expect(getByText('the lion
king')).toBeInTheDocument()
  })

  // wait for appearance and return the element
  const movie = await findByText('the lion king')
})
```

wait for element to be removed

```
test('submit button disappears', async () => {

    // Element is initially present...
    expect(() => screen.queryByRole('button',
                    {name: /submit/i})
    ).toBeDefined()

    // Wait for element to be removed
    await waitFor(() => {
       expect(screen.queryByRole(
          'button',
           { name: /submit/i})
       ).not.toBeInTheDocument()
   })
})
```

the screen object (description)

container	The target of ReactDOM.render()
baseElement	App wrapper (usually <body> tag)
debug(element)	Pretty print the DOM
unmount()	Unmount your component
rerender(ui)	Render again at the container
asFragment()	Return component as DocumentFragment
...queries	Queries for the baseElement

Visit our playground at: https://testing-playground.com and did you try out the eslint plugins?