



React Testing Library

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

Simple and complete cheat sheet

render a component

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

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

search the DOM

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

render(
  <label>
    Remember Me <input type="checkbox" />
  </label>,
)

const checkboxInput = screen.getByRole('checkbox', {
  name: /remember me/i,
})
```

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(screen.getByRole('textbox'), 'Hello World')
```

screen

<code>debug(element)</code>	Pretty print the DOM
<code>...queries</code>	Functions to query the DOM

search variants (result)

<code>getBy</code>	Element or Error
<code>getAllBy</code>	Element[] or Error
<code>queryBy</code>	Element or null
<code>queryAllBy</code>	Element[] or []
<code>findBy</code>	Promise<Element> or Promise<rejection>
<code>findAllBy</code>	Promise<Element[]> or Promise<rejection>

search types (result)

<code>Role</code>	<code><div role='dialog'>...</div></code>
<code>LabelText</code>	<code><label for="element" /></code>
<code>PlaceholderText</code>	<code><input placeholder="username" /></code>
<code>Text</code>	<code>About</code>
<code>DisplayValue</code>	<code><input value="display value" /></code>
<code>AltText</code>	<code></code>
<code>Title</code>	<code></code> or <code><title /></code>
<code>TestId</code>	<code><input data-testid='username-input' /></code>

text matches

```
render(<label>Remember Me <input type="checkbox" /></label>)

screen.getByRole('checkbox', {name: /remember me/i}) // ✓
screen.getByRole('checkbox', {name: 'remember me'}) // ✗
screen.getByRole('checkbox', {name: 'Remember Me'}) // ✓

// other queries accept text matches as well
// the text match argument can also be a function
screen.getByText((text, element) => { /* return true/false */ })
```

wait for appearance

```
test('movie title appears', async () => {
  render(<Movie />)

  // the element isn't available yet, so wait for it:
  const movieTitle = await screen.findByText(
    /the lion king/i,
  )

  // the element is there but we want to wait for it
  // to be removed
  await waitForElementToBeRemoved(() =>
    screen.getByLabelText(/loading/i),
  )

  // we want to wait until an assertion passes
  await waitFor(() =>
    expect(mockFn).toHaveBeenCalledWith('some arg'),
  )
})
```

render() options

<code>hydrate</code>	If true, will render with ReactDOM.hydrate
<code>wrapper</code>	React component which wraps the passed ui

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