



render a component

```
import { render } from '@testing-library/react'
const result = render(<MyComponent />)
```

search the DOM

```
import { screen } from '@testing-library/react'
render(<span>hello</span>)
const element = screen.getByText('hello')
```

interact with element

```
import userEvent from '@testing-library/react'
// 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'))
```

screen

debug(element)	Pretty print the DOM
...queries	Queries for the baseElement

search variants (result)

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

search types (result)

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

text matches

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

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 />)

  // element is initially not present...
  // wait for appearance
  const movieTitle = await findByText(/the lion king/i)
})
```

wait for element to be removed

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

  // Element is initially present...
  // Wait for element to be removed
  await waitForElementToBeRemoved(
    screen.getByRole('button', {name: /submit/i})
  )
})
```

wait for something

```
import { screen, waitFor } from '@testing-library/react'
// Retry search every 50ms for 4500ms

await waitFor(() =>
  expect(mockFn).toHaveBeenCalledTimes('some argument'))
```

render() options

container	Element the component will render into
baseElement	App wrapper (usually <body> tag)
hydrate()	If true, will render with ReactDOM.hydrate
wrapper	React component which wraps the passed ui