

# Query Functions

All query functions are accessed through the `screen` object in a test. These query functions *always* begin with one of the following names: `getBy`, `getAllBy`, `queryBy`, `queryAllBy`, `findBy`, `findAllBy`.

Start of Function Name	Examples
<code>getBy</code>	<code>getByRole</code> , <code>getByText</code>
<code>getAllBy</code>	<code>getAllByText</code> , <code>getByDisplayValue</code>
<code>queryBy</code>	<code>queryByDisplayValue</code> , <code>queryByTitle</code>
<code>queryAllBy</code>	<code>queryAllByTitle</code> , <code>queryAllByText</code>
<code>findBy</code>	<code>findByRole</code> , <code>findByText</code>
<code>findAllBy</code>	<code>findAllByText</code> , <code>findAllByDisplayValue</code>

These names indicate the following:

1. Whether the function will return an element or an array of elements
2. What happens if the function finds 0, 1, or > 1 of the targeted element
3. Whether the function runs instantly (synchronously) or looks for an element over a span of time (asynchronously)

## Looking for a Single Element?

Name	0 matches	1 match	> 1 match	Notes
<code>getBy</code>	Throw	Element	Throw	
<code>queryBy</code>	null	Element	Throw	
<code>findBy</code>	Throw	Element	Throw	Looks for an element over the span of 1 second

## Looking for Multiple Elements?

Name	0 matches	1 match	> 1 match	Notes
<code>getAllBy</code>	Throw	<code>[]Element</code>	<code>[]Element</code>	
<code>queryAllBy</code>	<code>[]</code>	<code>[]Element</code>	<code>[]Element</code>	
<code>findAllBy</code>	Throw	<code>[]Element</code>	<code>[]Element</code>	Looks for elements over the span of 1 second

## When to use each

Goal of test	Use
Prove an element exists	<code>getBy</code> , <code>getAllBy</code>
Prove an element does <b>not</b> exist	<code>queryBy</code> , <code>queryAllBy</code>
Make sure an element eventually exists	<code>findBy</code> , <code>findAllBy</code>

```
1 import { render, screen } from '@testing-library/react';
2
3 function ColorList() {
4   return (
5     <ul>
6       <li>Red</li>
7       <li>Blue</li>
8       <li>Green</li>
9     </ul>
10  );
11 }
12
13 render(<ColorList />);
```

+ Code

+ Text



```
1 test('getBy, queryBy, findBy finding 0 elements', async () => {
2   render(<ColorList />);
3
4   expect(
5     () => screen.getByRole('textbox')
6   ).toThrow();
7
8   expect(screen.queryByRole('textbox')).toEqual(null);
9
10  let errorThrown = false;
11  try {
12    await screen.findByRole('textbox');
13  } catch (err) {
```

+ Code

+ Text



```
1 test('getBy, queryBy, findBy when they find 1 element', async () => {
2   render(<ColorList />);
3
4   expect(
5     screen.getByRole('list')
6   ).toBeInTheDocument();
7   expect(
8     screen.queryByRole('list')
9   ).toBeInTheDocument()
10  expect(
11    await screen.findByRole('list')
12  ).toBeInTheDocument()
13 });
```

+ Code

+ Text



```
1 test('getBy, queryBy, findBy when finding > 1 elements', async () => {
2   render(<ColorList />);
3
4   expect(
5     () => screen.getByRole('listitem')
6   ).toThrow();
7
8   expect(
9     () => screen.queryByRole('listitem')
10  ).toThrow();
11
12  let errorThrown = false;
13  try {
```

+ Code

+ Text



```
1 test('getAllBy, queryAllBy, findAllBy', async () => {
2   render(<ColorList />);
3
4   expect(
5     screen.getAllByRole('listitem')
6   ).toHaveLength(3);
7
8   expect(
9     screen.queryAllByRole('listitem')
10  ).toHaveLength(3);
11
12  expect(
13    await screen.findAllByRole('listitem')
```

+ Code

+ Text

```
1 test('favor using getBy to prove an element exists', () => {
2   render(<ColorList />);
3
4   const element = screen.getByRole('list');
5
6   expect(element).toBeInTheDocument();
7 });
```

+ Code

+ Text

```
1 test('favor queryBy when proving an element does not exist', () => {
2   render(<ColorList />);
3
4   const element = screen.queryByRole('textbox');
5
6   expect(element).not.toBeInTheDocument();
7 });
```

+ Code

+ Text

```
1 import { useState, useEffect } from 'react';
2
3 function fakeFetchColors() {
4   return Promise.resolve(
5     ['red', 'green', 'blue']
6   );
7 }
8
9 function LoadableColorList() {
10   const [colors, setColors] = useState([]);
11
12   useEffect(() => {
13     fakeFetchColors()
```

+ Code

+ Text

```
1 test('Favor findBy or findAllBy when data fetching', async () => {
2   render(<LoadableColorList />);
3
4   const els = await screen.findAllByRole('listitem');
5
6   expect(els).toHaveLength(3);
7 });
```

+ Code

+ Text