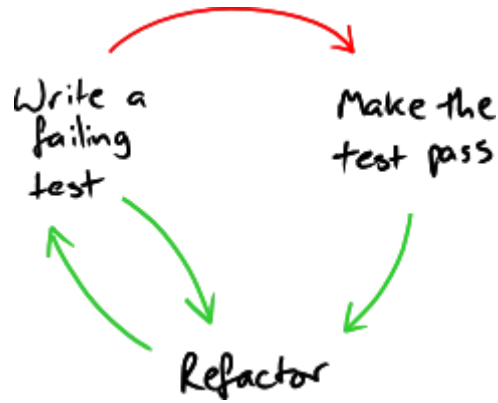


React Testing

React Component'ları için Nasıl Test Yazılır?



Jest: Facebook React component'larının testi için kendi geliştirdiği javascript test framework'ünü kullanmamızı öneriyor. Test için renderer package'ına sahip. Import ettiğiniz component için bir renderer create ettikten sonra test case'lerinizi yazmaya başlayabiliyorsunuz.

Enzyme: AirBnB tarafından geliştirilen bu kütüphane ile React component'larını test edebiliyoruz. Shallow Rendering özelliği ile aynı jest gibi bir render simülasyonu yaparak test yazmamızı sağlıyor. Enzyme'in güzel yanı ise sadece test framework'lerine React component'larının testi için yardımcı bir araç olarak tasarlanmış olması. Bu sayede ister Jest ile isterseniz Mocha veya Jasmine ile birlikte kullanabiliyorsunuz.

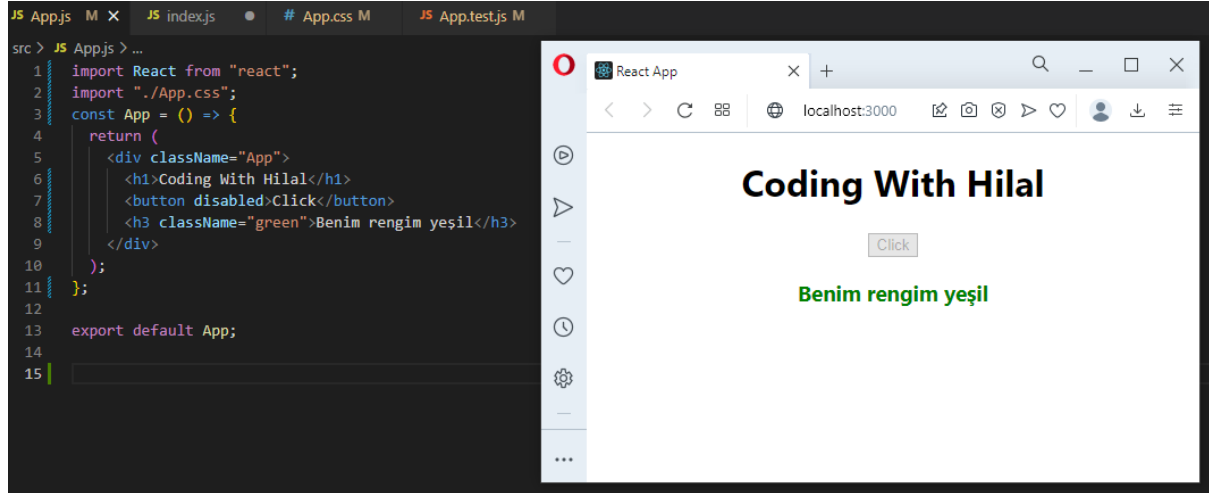
React Testing Library

React Testing Library Kent C. Dodds tarafından enzyme alternatifi olarak oluşturulmuş ve react komponentlerini test edebilmek için kullanabileceğimiz bir test kütüphanesidir.

React Testing with React Testing Library

Burada bir React App oluşturuluyor. Title belirleniyor.

<button> ekleniyor fakat bu buton 'disabled' olarak pasifleştiriliyor.



Testi gerçekleştirdiğimiz **App.test.js** klasörünü inceleyelim.

render, React bileşenlerini DOM'a render etmek için kullanılan bir fonksiyondur. Bu fonksiyon, React ağacını sanal bir DOM'a dönüştürür ve ardından bu sanal DOM'u gerçek DOM'a ekler. render fonksiyonu, React uygulamasını yalnızca bir kez oluşturur ve ardından DOM işlemleri gerçekleştirir. Bu, uygulamanın performansını artırır ve testlerin hızlı çalışmasını sağlar.

screen, DOM üzerindeki öğeleri seçmek için kullanılan bir araçtır. screen aracılığıyla öğeleri data-testid gibi özel özniteliklerini kullanarak veya sınıfları, metinleri vb. kullanarak seçebilirsiniz. screen ayrıca, öğelerin görünür olup olmadığını, etkinleştirilip etkinleştirilmediğini ve diğer durumları da kontrol edebilir.

```
JS App.js M JS index.js # App.css M JS App.test.js M X
src > JS App.test.js > ...
1 import { render, screen } from '@testing-library/react';
2 import App from './App';
3
4 test("Header renders correctly", () => {
5
6   render(<App />);
7
8   const headerEl = screen.getByText(/coding with Hilal/i);
9   expect(headerEl).toBeInTheDocument();
10  expect(headerEl).toHaveTextContent("Coding With Hilal");
11
12 });
13
14 test("Green Element Renders Correctly", () => {
15   render(<App />);
16   const greenElement = screen.getByText(/benim rengim yeşil/i);
17
18   expect(greenElement).toBeInTheDocument();
19   expect(greenElement).toHaveTextContent("Benim rengim yeşil");
20   expect(greenElement).toHaveClass("green");
21
22 });
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PASS src/App.test.js
✓ Header renders correctly (24 ms)

PASS src/App.test.js
✓ Header renders correctly (29 ms)

PASS src/App.test.js
✓ Header renders correctly (23 ms)
✓ Green Element Renders Correctly (6 ms)

Test Suites: 1 passed, 1 total
Tests: 2 passed, 2 total
Snapshots: 0 total
Time: 1.111 s
Ran all test suites related to changed files.

Burada butonlarımıza test-id atıyoruz. Bu sayede <enabled> ve <disabled> butonlarımızı birbirinden ayırt edip, kontrollerini sağlıyoruz.

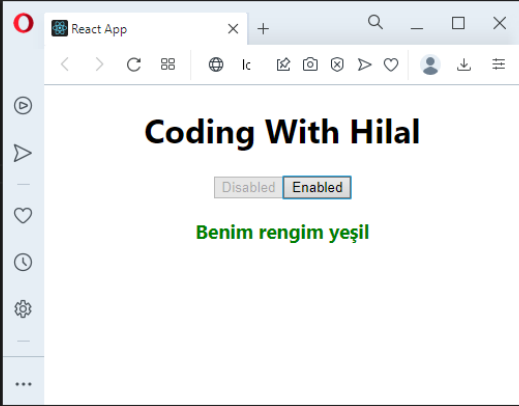
Testimizin gerçekleşmesi için her test case'inden sonra render() işlemini ekliyoruz.

toBeInTheDocument() : Varlığı kontrol edilir.

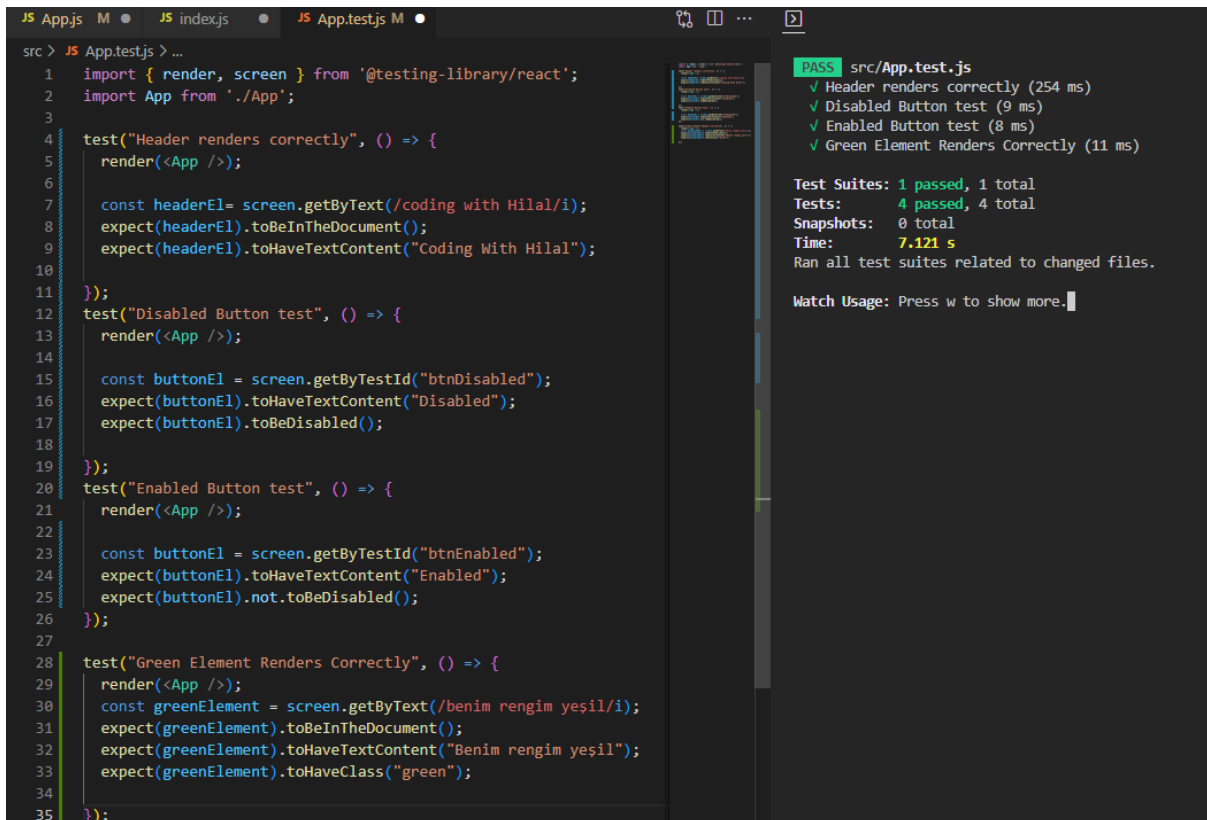
toHaveTextContent() : Verilen text'in doğruluğu test edilir.

toBeDisabled() : Butonun pasif olduğu test edilir.

not.toBeDisabled() : Butonun pasif olmadığı (enabled) olduğu durum kontrol edilir.



```
src > JS App.js > [App] App
1 import React from "react";
2 import "../App.css";
3 const App = () => {
4   return (
5     <div className="App">
6       <h1>Coding With Hilal</h1>
7       <button data-testid="btnDisabled" disabled> Disabled </button>
8       <button data-testid="btnEnabled" enabled> Enabled </button>
9       <h3 className="green">Benim rengim yeşil</h3>
10    </div>
11  );
12 };
13
14 export default App;
15
16
```



```
src > JS App.test.js > ...
1 import { render, screen } from '@testing-library/react';
2 import App from '../App';
3
4 test("Header renders correctly", () => {
5   render(<App />);
6
7   const headerEl = screen.getByText(/coding with hilal/i);
8   expect(headerEl).toBeInTheDocument();
9   expect(headerEl).toHaveTextContent("Coding With Hilal");
10 });
11
12 test("Disabled Button test", () => {
13   render(<App />);
14
15   const buttonEl = screen.getByTestId("btnDisabled");
16   expect(buttonEl).toHaveTextContent("Disabled");
17   expect(buttonEl).toBeDisabled();
18 });
19
20 test("Enabled Button test", () => {
21   render(<App />);
22
23   const buttonEl = screen.getByTestId("btnEnabled");
24   expect(buttonEl).toHaveTextContent("Enabled");
25   expect(buttonEl).not.toBeDisabled();
26 });
27
28 test("Green Element Renders Correctly", () => {
29   render(<App />);
30   const greenElement = screen.getByText(/benim rengim yeşil/i);
31   expect(greenElement).toBeInTheDocument();
32   expect(greenElement).toHaveTextContent("Benim rengim yeşil");
33   expect(greenElement).toHaveClass("green");
34 });
35
```

PASS src/App.test.js

- ✓ Header renders correctly (254 ms)
- ✓ Disabled Button test (9 ms)
- ✓ Enabled Button test (8 ms)
- ✓ Green Element Renders Correctly (11 ms)

Test Suites: 1 passed, 1 total

Tests: 4 passed, 4 total

Snapshots: 0 total

Time: 7.121 s

Ran all test suites related to changed files.

Watch Usage: Press w to show more.

beforeEach()

beforeEach() fonksiyonu, @testing-library/react kütüphanesi ile yazılan testlerde sıklıkla kullanılan bir Jest fonksiyonudur. beforeEach() fonksiyonu, test sürecinde önceden belirlenmiş işlemleri yapmak için kullanılır. Bu işlemler, her bir test çalıştırılmadan önce otomatik olarak gerçekleştirilir.

beforeEach() fonksiyonu içerisine alınan render() sayesinde , her test case render edilir.

```
JS App.js M ● JS index.js ● JS App.test.js M X
src > JS App.test.js > ...
1 import { render, screen } from '@testing-library/react';
2 import App from './App';
3 import { scryRenderedComponentsWithType } from 'react-dom/test-utils';
4
5 beforeEach(() => {
6   render(<App />);
7 });
8
9 test("Header renders correctly", () => {
10
11   const headerEl= screen.getByText(/coding with Hilal/i);
12   expect(headerEl).toBeInTheDocument();
13   expect(headerEl).toHaveTextContent("Coding With Hilal");
14
15 });
16 test("Disabled Button test", () => {
17
18
19   const buttonEl = screen.getByTestId("btnDisabled");
20   expect(buttonEl).toHaveTextContent("Disabled");
21   expect(buttonEl).toBeDisabled();
22
23 });
24 test("Enabled Button test", () => {
25
26
27   const buttonEl = screen.getByTestId("btnEnabled");
28   expect(buttonEl).toHaveTextContent("Enabled");
29   expect(buttonEl).not.toBeDisabled();
30 });
31
32 test("Green Element Renders Correctly", () => {
33
34   const greenElement = screen.getByText(/benim rengim yeşil/i);
35   expect(greenElement).toBeInTheDocument();
36   expect(greenElement).toHaveTextContent("Benim rengim yeşil");
37   expect(greenElement).toHaveClass("green");
38
39
40 });
41
```

- **beforeAll():** Test dosyasındaki tüm testler çalıştırılmadan önce yalnızca bir kez çağrılır. Bu fonksiyon, genellikle test için gereken verilerin veya bileşenlerin önceden hazırlanması için kullanılır.
- **beforeEach():** Test dosyasındaki her bir test çalıştırılmadan önce çağrılır. Bu fonksiyon, her bir test için gerekli verilerin ve bileşenlerin önceden hazırlanması için kullanılır.
- **afterAll():** Test dosyasındaki tüm testler tamamlandıktan sonra yalnızca bir kez çağrılır. Bu fonksiyon, genellikle test sırasında oluşturulan kaynakların temizlenmesi için kullanılır.
- **afterEach():** Test dosyasındaki her bir test tamamlandıktan sonra çağrılır. Bu fonksiyon, her bir test sırasında oluşturulan kaynakların temizlenmesi için kullanılır.

The screenshot shows a code editor with Jest test code on the left and the corresponding console output on the right.

Code Editor (Left):

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

beforeAll(() => {
  console.log("Before All");
});

beforeEach(() => {
  render(<App />);
  console.log("Before Each");
});

afterAll(() => {
  console.log("After All");
});

afterEach(() => {
  console.log("After Each");
});

test("Header renders correctly", () => {
  const headerEl = screen.getByText(/coding with Hilal/i);
  expect(headerEl).toBeInTheDocument();
  expect(headerEl).toHaveTextContent("Coding With Hilal");
});

test("Disabled Button test", () => {
  const buttonEl = screen.getByTestId("btnDisabled");
  expect(buttonEl).toHaveTextContent("Disabled");
  expect(buttonEl).toBeDisabled();
});

test("Enabled Button test", () => {
  const buttonEl = screen.getByTestId("btnEnabled");
  expect(buttonEl).toHaveTextContent("Enabled");
  expect(buttonEl).not.toBeDisabled();
});

test("Green Element Renders Correctly", () => {
  const greenEl = screen.getByText(/green/i);
  expect(greenEl).toBeInTheDocument();
  expect(greenEl).toHaveTextContent("Green Element");
});
```

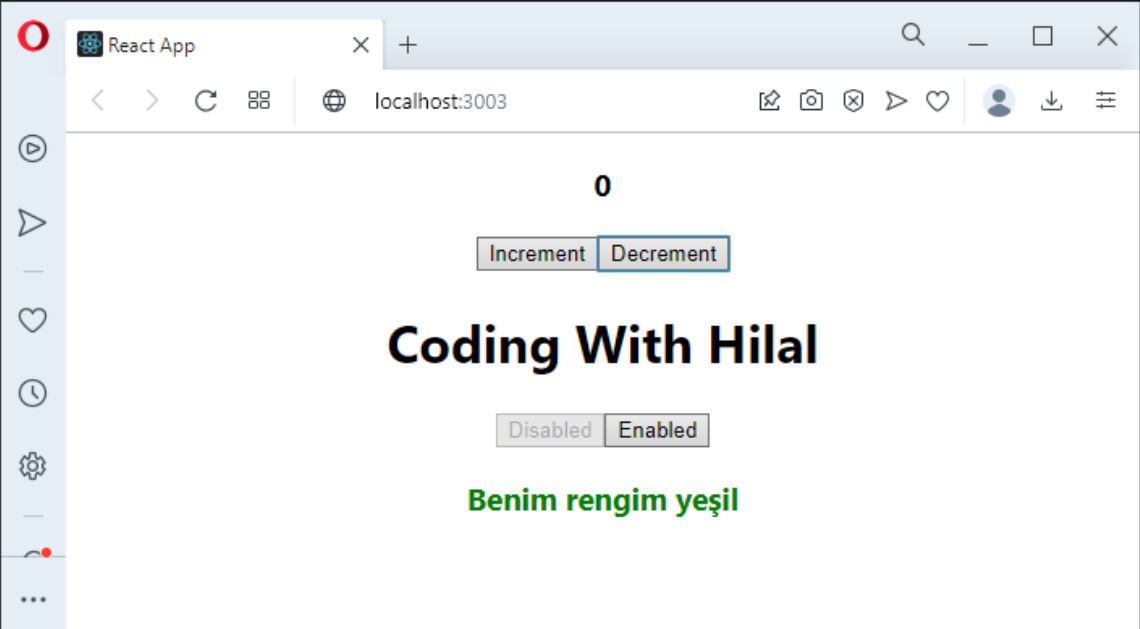
Console Output (Right):

```
Before All
at Object.<anonymous> (src/App.test.js:11:11)
console.log
After All
at Object.<anonymous> (src/App.test.js:11:11)
Before Each
at Object.<anonymous> (src/App.test.js:11:11)
console.log
Before Each
at Object.<anonymous> (src/App.test.js:11:11)
console.log
After Each
at Object.<anonymous> (src/App.test.js:11:11)
console.log
Before Each
at Object.<anonymous> (src/App.test.js:11:11)
console.log
After Each
at Object.<anonymous> (src/App.test.js:11:11)
console.log
Before Each
at Object.<anonymous> (src/App.test.js:11:11)
console.log
After Each
at Object.<anonymous> (src/App.test.js:11:11)
console.log
After All
at src/App.test.js:15:11
PASS src/App.test.js
✓ Header renders correctly (69 ms)
✓ Disabled Button test (11 ms)
✓ Enabled Button test (6 ms)
✓ Green Element Renders Correctly (10 ms)
```

Counter

Bu projede bir Counter bulunmaktadır. Counter'ın 'Increment' ve 'Decrement' olmak üzere iki adet butonu vardır. Increment butonu sayıyı 1 arttırırken, Decrement butonu 1 azaltmaktadır.

```
src > components > JS Counter.js > [default]
1  import React, {useState} from "react";
2
3  const Counter = () => {
4    const [counter, setCounter] = useState(0);
5    return(
6      <div>
7        <h3 data-testid="counter">{counter}</h3>
8        <button onClick={() => setCounter((prevCounter) => prevCounter + 1)}>
9          Increment
10       </button>
11       <button onClick={() => setCounter((prevCounter) => prevCounter - 1)}>
12         Decrement
13       </button>
14     </div>
15   );
16 };
17
18 export default Counter;
```



Counter.test

userEvent kütüphanesi, kullanıcının gerçekleştirdiği etkileşimleri taklit eden bir fonksiyon seti sağlar. Böylece, uygulamanızda bulunan bileşenlere gerçek kullanıcılar gibi etkileşimlerde bulunarak test senaryolarınızı gerçek hayata daha yakın hale getirebilirsiniz.

Burada öncelikle Counter'ın başlangıç değerinin 0 olması isteniyor.

Daha sonra Increment butonunun çalışabilirliği test ediliyor. Buton ilk olarak 0 değeri döndürüyor , kullanıcı butonu bastıktan sonra counter'ın 1 artması test edilir.

```
JS App.js M JS index.js JS App.test.js M JS Counter.js U JS Counter.test.js U ●
src > components > JS Counter.test.js > ...
1  import {render, screen} from "@testing-library/react";
2  import userEvent from "@testing-library/user-event";
3  import Counter from "../Counter";
4
5  beforeEach(() => {
6    render(<Counter />);
7  });
8
9
10 test("Counter is initially 0", () => {
11   const counterEl = screen.getByTestId("counter");
12
13   expect(counterEl).toHaveTextContent(0);
14 });
15
16 test("Increment button works correctly", () => {
17   const counterEl = screen.getByTestId("counter");
18   const incrementBtn = screen.getByText('Increment');
19   expect(counterEl).toHaveTextContent(0);
20
21   userEvent.click(incrementBtn);
22
23   expect(counterEl).toHaveTextContent(1);
24
25
26 });
27
```

PROBLEMS OUTPUT DEBUG CONSOLE

✓ **TERMINAL**

```
at onClick (src/components/Counter.js:8:36)
at HTMLUnknownElement.callCallback (node_modules/react-dom/cjs/react-dom.development.js:187:14)
at HTMLUnknownElement.callTheUserObjectsOperation (node_modules/jsdom/lib/jsdom/living/events/HTMLUnknownElement.js:26:30)

Test Suites: 2 passed, 2 total
Tests: 6 passed, 6 total
Snapshots: 0 total
Time: 21.516 s
Ran all test suites related to changed files.
```