14.Tester une application React

14.1. Que tester dans une application React ?

Sur une application front-end, qu'elle soit en React ou non, la question se pose régulièrement de savoir quoi tester unitairement. En effet, si l'on prend par exemple un composant qui n'est responsable que de présentation en générant du HTML, il serait laborieux de tester précisément le rendu du composant dans le navigateur. De plus, il est possible que le moindre changement dans le CSS associé au composant change le rendu au point de mettre le test en échec, ce qui n'est généralement pas le but.

Pas certains aspects, il est tout de même intéressant de tester quelques composants :

- pour tester les informations affichées
- pour tester le comportement du composant en réponse à des actions de l'utilisateur.

14.2. Test unitaire de composants avec Enzyme

Enzyme est un utilitaire de test JavaScript permettant de tester facilement les composants React. Il aide à rendre les composants React en mode test.

Pour démarrer avec Enzyme, installez-le via npm avec la commande suivante.

```
npm install --save-dev enzyme
npm install --save-dev enzyme-adapter-react-16 --force
```

Rédaction du premier cas de test

Étape 1 : Nous allons rendre un simple bouton nommé **`Click Me`** en utilisant le code suivant.

```
import React, { Component } from 'react';
import './App.css';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {};
  }
  render() {
    return (
      <div>
        <button className="click-me" id="ClickMe">Click Me</button>
      </div>
    )
  }
};

export default App;
```

Étape 2 : Ajoutez le code suivant dans le fichier **App.test.js**, qui est le fichier dans lequel nous écrivons les cas de test.

```
import React from 'react'
import Enzyme, { shallow } from 'enzyme'
import Adapter from 'enzyme-adapter-react-16'
import App from './App'

Enzyme.configure({ adapter: new Adapter() })

describe('Test Case For App', () => {
  it('should render button', () => {
    const wrapper = shallow(<App />)
    const buttonElement  = wrapper.find('#ClickMe');
    expect(buttonElement).toHaveLength(1);
    expect(buttonElement.text()).toEqual('Click Me');
  })
})
```

Étape 3 : Utilisez la commande suivante pour exécuter les scénarios de test.

```
npm test
```

Les résultats du test seront affichés comme dans la capture d'écran suivante.

```
PASS src/App.test.js
  Test Case For App
    √ should render button (9 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.526 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Scénario de test pour la variable d'état

Créons un nouveau cas de test pour vérifier l'état désactivé/activé du bouton à l'aide de la variable d'état.

Étape 1 : Ajoutez l'extrait de code ci-dessous dans le fichier **App.js**.

```
import React, { Component } from 'react';
import './App.css';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      ClickCount:0,
      IamDisabled : true
    };
    this.ClickMe = this.ClickMe.bind(this);
  }
  ClickMe(){
    this.setState({
      ClickCount:this.state.ClickCount + 1
    });
  }
  render() {
    return (
      <div>
        <button  className="click-me" id="ClickMe"
onClick={this.ClickMe}>Click Me</button>
        <p>You clicked me :: {this.state.ClickCount}</p>
        <button  className="click-me"
disabled={this.state.IamDisabled}>Disabled</button>
      </div>
    )
  }
};
export default App;
```

Étape 2 : Ajoutez l'extrait de code suivant dans le fichier App.test.js.

```
import React from 'react'
import Enzyme, { shallow } from 'enzyme'
import Adapter from 'enzyme-adapter-react-16'
import App from './App'

Enzyme.configure({ adapter: new Adapter() })

describe('Test Case For App', () => {
  it('should render button', () => {
    const wrapper = shallow(<App />)
    const buttonElement  = wrapper.find('#ClickMe');
    expect(buttonElement).toHaveLength(1);
    expect(buttonElement.text()).toEqual('Click Me');
  }),

  it('increments count by 1 when button is clicked', () => {
    const wrapper = shallow(<App />);
    const buttonElement  = wrapper.find('#ClickMe');
    buttonElement.simulate('click');
    const text = wrapper.find('p').text();
    expect(text).toEqual('You clicked me :: 1');
  });
})

describe('Test Case for App Page', () => {
    test('Validate Disabled Button disabled', () => {
        const wrapper = shallow(
            <App />
        );
        expect(wrapper.state('IamDisabled')).toBe(true);
    });
});
```

Les résultats du test seront affichés comme dans la capture d'écran suivante.

```
PASS src/App.test.js
  Test Case For App
    √ should render button (12 ms)
    √ increments count by 1 when button is clicked (3 ms)
  Test Case for App Page
    √ Validate Disabled Button disabled (1 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        2.637 s
Ran all test suites related to changed files.
```