Unit Tests with Enzyme

In this lesson, we'll discuss unit testing in React using a testing utility called Enzyme.

Introduction to Enzyme

Enzyme is a testing utility by Airbnb to assert, manipulate, and traverse React components. It is used to conduct unit tests to complement snapshot tests in React. First we have to install it along with its extension, since it doesn't come with *create-react-app*.

Installing Enzyme

```
npm install --save-dev enzyme react-addons-test-utils enzyme-adapter-react-16
```

Second, we include it in the test setup and we initialize its adapter.

```
import React from 'react';
import ReactDOM from 'react-dom';
import renderer from 'react-test-renderer';
import Enzyme from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';
import App, { Search, Button, Table } from './App';

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

Now you can write your first unit test in the Table "describe"-block. You will use `shallow()` to render your component and assert that the Table was passed two items. The assertion simply checks if the element has two elements with the class `table-row`.

```
import React from 'react';
import ReactDOM from 'react-dom';
import renderer from 'react-test-renderer';
import Enzyme, { shallow } from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';
import App, { Search, Button, Table } from './App';

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

```
describe('Table', () => {


  const props = {
    list: [
      { title: '1', author: '1', num_comments: 1, points: 2, objectID: 'y' },
      { title: '2', author: '2', num_comments: 1, points: 2, objectID: 'z' },
    ],
  };

  ...

  it('shows two items in list', () => {
    const element = shallow(
      <Table { ...props } />
    );

    expect(element.find('.table-row').length).toBe(2);
  });

});
```

Shallow renders the component without its child components, so you can dedicate the test to one component.

Enzyme has three rendering mechanisms in its API. You already know `shallow()`, but there is also `mount()` and `render()`. Both instantiate instances of the parent component and all child components. `mount()` gives you access to the component lifecycle methods.

When to use what

- Always begin with a shallow test
- If `componentDidMount()` or `componentDidUpdate()` should be tested, use `mount()`
- If you want to test component lifecycle and children behavior, use `mount()`
- If you want to test a component's children rendering with less overhead than `mount()` and you are not interested in lifecycle methods, use `render()`

Continue to unit test your components, but be sure to keep the tests simple and maintainable. Otherwise, you will have to refactor them once you change your components. This is one the main reason Facebook introduced snapshot tests with Jest.

Exercises

- Write a unit test with Enzyme for your Button component
- Keep your unit tests up to date during the following chapters

Further Reading:

- Read about Enzyme and its rendering API
- Read about testing React applications