strategy | consulting | digital | technology | operations

REACTJS – MODULE 12

USER INTERFACE TESTING

accenture

AGENDA

- Introduction to User Interface Testing
- Introduction to enzyme
- Testing using enzyme
- Introduction to snapshot testing
- Demo
- Activity

accenture

Introduction to User Interface Testing

Testing the front end of the application with respect to UI elements is known as User Interface testing

User Interface testing is a time consuming process, as it involves checking every individual element in the web page

React provides the facility to perform testing of a User interface with respect to react component

Introduction to Enzyme

Enzyme is a javascript testing utility, which is used to test react components with respect to UI

Enzyme provides easy way to use assertions, it also provides easy way to traverse react components

Enzyme is compatible with most of the test runners like karma, mocha etc.

Testing Using Enzyme

In order to use enzyme for the react project, the first thing is to do installation of enzyme and the enzyme adapter with respect to react

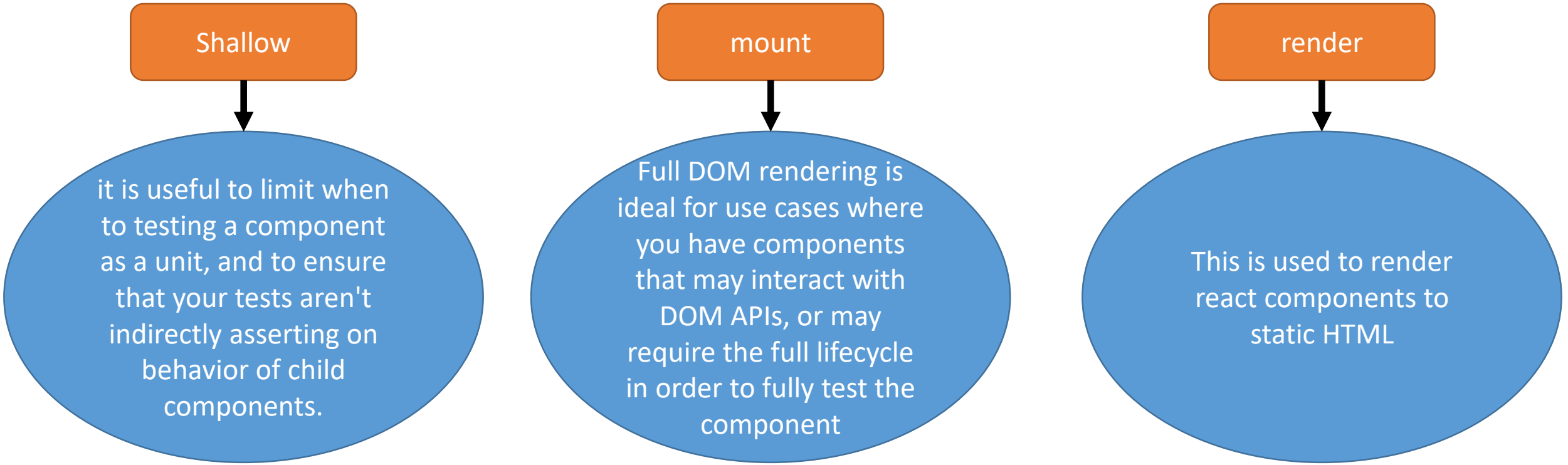
```
C:\reactdemos\enzymeapp>npm i --save-dev enzyme enzyme-adapter-react-16
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.2 (node_modules\react-scripts\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.1.2: wanted {"os":"darwin","arch":"any
"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.3 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.1.3: wanted {"os":"darwin","arch":"any
"} (current: {"os":"win32","arch":"x64"})

+ enzyme-adapter-react-16@1.1.0
+ enzyme@3.2.0
added 255 packages in 71.47s
```

Note

Get information about installation
http://airbnb.io/enzyme/docs/installation/index.html

Testing Using Enzyme

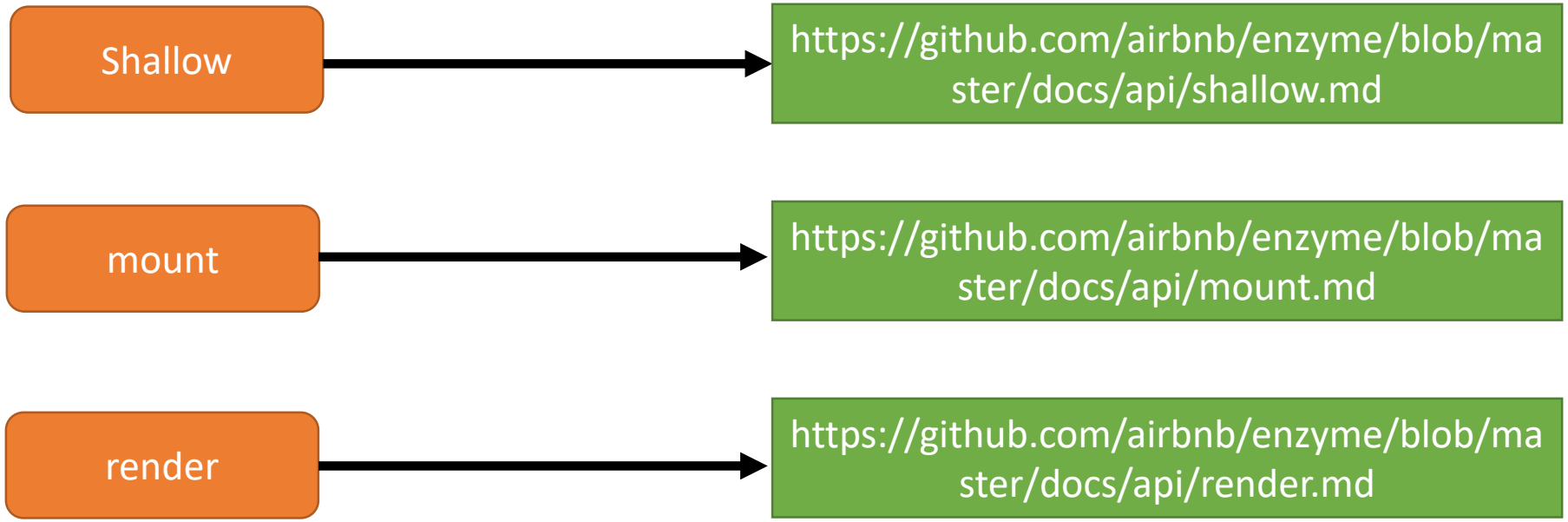
Enzyme uses some API like shallow, mount and a render function to render each element and capture its value



accenture

Testing Using Enzyme

Get more information about enzyme API's by visiting respective links



Demo – User Interface Testing Using Enzyme

Create new react project “enzymeapp”

Open the project in Webstorm and provide some additional information about enzyme configuration inside App.test.js file

```
import { configure } from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';

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

Demo – User Interface Testing Using Enzyme

Open App.js and code it as follows

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

class App extends Component {
  render() {
    return (
      <div className="App">
          Hello
      </div>
    );
  }
}
export default App;
```

accenture

Demo – User Interface Testing Using Enzyme

Before writing test case import the enzyme with respect to shallow, mount and render

```
import { shallow, mount, render } from 'enzyme';
```

Write test case to test whether “Hello” is present in a react component

```
describe('A suite', function() {
    it('renders without crashing', () => {
        const div = document.createElement('div');
        ReactDOM.render(<App/>, div);
    });
    it('should render to static HTML', function() {
        expect(render(<App />).text()).toEqual('Hello');
    });
});
```

accenture

Demo – User Interface Testing Using Enzyme

Before writing test case import the enzyme with respect to shallow, mount and render

```
import { shallow, mount, render } from 'enzyme';
```

Write test case to test whether “Hello” is present in a react component

```
describe('A suite', function() {
    it('renders without crashing', () => {
        const div = document.createElement('div');
        ReactDOM.render(<App/>, div);
    });
    it('should render to static HTML', function() {
        expect(render(<App />).text()).toEqual('Hello');
    });
});
```

Describe is to start the test suite

Test case to check successful rendering of App

Test case to check Hello text is present inside App component

Demo – User Interface Testing Using Enzyme

In the terminal execute the test using
npm test

```
 PASS  src\App.test.js
  A suite
    √ renders without crashing
    √ should render to static HTML

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        3.903s, estimated 4s
Ran all test suites.
```

accenture

Demo 2 – Testing React UI

Use existing react project “enzymeapp” and do App.js code modification as follows

```
import React, { Component } from 'react';
import './App.css';
class App extends Component {
  render() {
    return (
      <div className="App">
          Hello
          <p>Accenture</p>
          <h3>

          </h3>
          <form>
              <input type="text"/>
              <button>Click</button>
          </form>
      </div>
    );
  }
}
export default App;
```

accenture

Demo 2 – Testing React UI

Write test cases in App.test.js

Test case to check whether App component have class selector as “App”

```
it('should be selectable by class selector', function() {
    expect(shallow(<App />).is('.App')).toBe(true);
});
```

accenture

Demo 2 – Testing React UI

Test case to check whether App component mounted properly in DOM

```
it('should mount in a full DOM', function() {
    expect(mount(<App />).find('.App').length).toBe(1);
});
```

Demo 2 – Testing React UI

Test case to check whether App component have single paragraph tag

```
it('should check number of p tag', function() {
    const shal=mount(<App />).find('p').length;
    expect(shal).toBe(1);
});
```

accenture

Demo 2 – Testing React UI

Test case to check whether paragraph have the text as “Accenture”

```
it('should check for text of p tag', function() {
    const shal=mount(<App />).find('p').text();
    expect(shal).toEqual('Accenture');
});
```

Demo 2 – Testing React UI

Test case to check whether h3 tag is empty

```
it('should check for h3 tag text to be empty',function () {
    const shal=mount(<App />).find('div').childAt(1).text();
    expect(shal).toBe('');
});
```

Demo 2 – Testing React UI – Test Result

```
 PASS  src\App.test.js (5.064s)
  A suite
    √ renders without crashing
    √ should be selectable by class selector (15ms)
    √ should mount in a full DOM
    √ should check number of p tag
    √ should check for text of p tag
    √ should check for h3 tag text to be empty (16ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        5.173s, estimated 6s
Ran all test suites.
```

Activity

1. Use existing project and write test cases as mentioned below
2. Test Case 1 :

   Write a test case which will check whether form is present in a App component
3. Test Case 2:

   Write a test case whether textbox and button controls are child element of form

Introduction to Snapshot Testing

Some time there may be a requirement to keep track on UI, like a developer may have to ensure that the application is working fine before and after some changes

This could be done using snapshot testing

Snapshot testing is also a kind of regression testing which will ensure that previously developed application works fine even after some change

In react we can use testing utility “jest” to perform snapshot testing

accenture

How Snapshot Testing Works

When snapshot testing is executed for the first time it takes the snapshot of a component and saves it

Next time when you run snapshot testing again, it compares recent rendered component to the snapshot which it has taken earlier

In case it is differed which means test is fail

In such scenario either developer will update snapshot or fix the component to make proper match

accenture

Demo – Snapshot Testing

Create new react project “snapshottest” and open it in Webstorm

In this module we are executing snapshot testing with the help of jest, hence we need to install “npm install - - save-dev react-test-renderer”

```
C:\reactdemos\snapshottest>npm install --save-dev react-test-renderer
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.2 (node_modules\react-scripts\node_modules\fsevents):
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: Error: EPERM: operation not permitted, rename 'C:\reactdemos\snapshottes
t\node_modules\.staging\fsevents-2e8aecbe\node_modules\asynckit' -> 'C:\reactdemos\snapshottest\node_modules\.staging\as
ynckit-7b1c27f2'
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.3 (node_modules\fsevents):
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: Error: EPERM: operation not permitted, rename 'C:\reactdemos\snapshottes
t\node_modules\.staging\fsevents-7735b202\node_modules\ajv' -> 'C:\reactdemos\snapshottest\node_modules\.staging\ajv-336
cedc0'

+ react-test-renderer@16.2.0
added 230 packages and updated 1 package in 121.621s
```

Demo – Snapshot Testing

Keep the code inside App.js unchanged but open App.test.js and write below code

```
it('renders a snapshot', () => {
    const tree = renderer.create(<App/>).toJSON();
    expect(tree).toMatchSnapshot();
});
```

Demo – Snapshot Testing

```
it('renders a snapshot', () => {
    const tree = renderer.create(<App/>).toJSON();
    expect(tree).toMatchSnapshot();
});
```

renderer.create will render the element. This will use method toJSON()

toJSON() will convert component representation into JSON

Demo – Snapshot Testing

```
it('renders a snapshot', () => {
    const tree = renderer.create(<App/>).toJSON();
    expect(tree).toMatchSnapshot();
});
```

This line will either create new snapshot or compare with existing snapshot to find the difference

Demo – Snapshot Testing

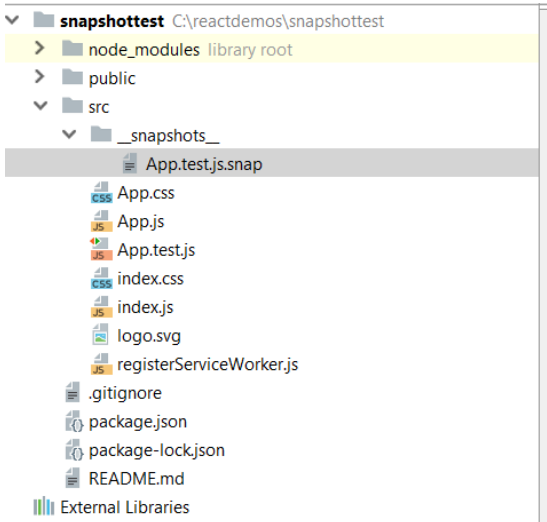
Run the test using
npm test

```
PASS  src\App.test.js
  √ renders a snapshot (16ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   1 passed, 1 total
Time:        1.959s, estimated 4s
Ran all test suites.
```

accenture

Demo – Snapshot Testing

When the snapshot test executed for the first time it will create _snapshot_ folder inside src with App.test.js.snap file




```
// Jest Snapshot v1, https://goo.gl/fbAQLP
exports[`renders a snapshot 1`] = `
<div
  className="App"
>
  <header
    className="App-header"
  >
    <img
      alt="logo"
      className="App-logo"
      src="logo.svg"
    />
    <h1
      className="App-title"
    >
      Welcome to React
    </h1>
  </header>
  <p
    className="App-intro"
  >
    To get started, edit
    <code>
      src/App.js
    </code>
     and save to reload.
  </p>
</div>
```

Demo – Snapshot Testing

Do not change code inside App.test.js but modify App.js by adding paragraph tag as mentioned below

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
          <p>Hello Snapshot</p>
      </div>
    );
  }
}
export default App;
```

accenture

Demo – Snapshot Testing

Run the test using
npm test

```
  × renders a snapshot

Snapshot Summary
 › 1 snapshot test failed in 1 test suite. Inspect your code changes or press `u` to update them.

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   1 failed, 1 total
Time:        1.938s, estimated 2s
Ran all test suites.
```

accenture

Demo – Snapshot Testing

Run the test using
npm test

```
  × renders a snapshot

Snapshot Summary
 › 1 snapshot test failed in 1 test suite. Inspect your code changes or press `u` to update them.

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   1 failed, 1 total
Time:        1.938s, estimated 2s
Ran all test suites.
```

Test is fail because paragraph is added extra which was not present when the first time snapshot was taken

Demo – Snapshot Testing

Run the test using
npm test

```
  × renders a snapshot

Snapshot Summary
 › 1 snapshot test failed in 1 test suite. Inspect your code changes or press `u` to update them.

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   1 failed, 1 total
Time:        1.938s, estimated 2s
Ran all test suites.
```

To update the snapshot press “u” in the terminal

Demo – Snapshot Testing

This time snapshot test is pass because snapshot is updated

```
 PASS  src\App.test.js
  √ renders a snapshot

Snapshot Summary
 › 1 snapshot updated in 1 test suite.

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   1 updated, 1 total
Time:        1.859s, estimated 2s
Ran all test suites.
```

accenture

MODULE SUMMARY

- Understand about Front end testing
- How to use enzyme to perform UI testing
- Understand snapshot testing
- Benefit of snapshot testing
- How to perform snapshot testing

accenture

THANK YOU

accenture