WORKSHOP

REACT/REDUX TESTING

TEXT

REPOSITORY

- https://gitlab.thinknet.co.th/kunapot/react-redux-testing-workshop
- yarn test or npm run test

HOW TO TEST

- Presentational Component
- Container Component
- Reducer
- Action
- Store

PRESENTATIONAL COMPONENT

- Define how things look
- Contain markup and styles
- Don’t specify how the data is loaded
- Receive data and callback via props
- Rarely have own state
- May written as functional component

EXAMPLE

- src/components/CommentItem.js

```
import React from 'react'

const CommentItem = (props) => (
  <div className="box">
    <div className="media-content">
      <div className="content">
        <p>
          <strong>{props.title}</strong> <small>{props.author}</small>
          <br />
          {props.body}
        </p>
      </div>
    </div>
  </div>
)

export default CommentItem
```

HOW TO TEST

- id, className, element
- Render data props correctly
- Snapshot (optional)
- Simulate event
- State and callback (optional)
- Lifecycle (optional)

TESTING TOOLS

TEST RUNNER

- Jest - https://facebook.github.io/jest

REACT TESTING UTILITY

- Enzyme - http://airbnb.io/enzyme/index.html

PRESENTATIONAL COMPONENT

- src/components/CommentItem.js

```
import React from 'react'

const CommentItem = (props) => (
  <div className="box">
    <div className="media-content">
      <div className="content">
        <p>
          <strong>{props.title}</strong> <small>{props.author}</small>
          <br />
          {props.body}
        </p>
      </div>
    </div>
  </div>
)

export default CommentItem
```

PRESENTATIONAL COMPONENT

- src/components/CommentItem.test.js

```
describe('<CommentItem />', () => {
  it('render correctly', () => {

  })
})
```

PRESENTATIONAL COMPONENT

- src/components/CommentItem.test.js

```
import React from 'react'
import { shallow } from 'enzyme'
import CommentItem from './CommentItem'

describe('<CommentItem />', () => {
  it('render correctly', () => {
    const props = {
      title: 'title',
      body: 'body',
      author: '@john doe',
      postat: '31m',
    }

    const wrapper = shallow(<CommentItem {...props} />)
    expect(wrapper.html()).toMatchSnapshot()
  })
})
```

PRESENTATIONAL COMPONENT

- src/components/CommentList.js

```
import React from 'react'
import CommentItem from './CommentItem'

const CommentList = props => (
  <div>
    <div id="comment-list">
      {
        props.comments.map((comment, index) => (
          <CommentItem
            key={`comment-${index}`}
            title={comment.title}
            body={comment.body}
            author={comment.author}
            postat={comment.postat}
          />
        ))
      }
    </div>
    <button id="load-more" className="button is-primary load-more" onClick={props.loadMore}>Load More</button>
  </div>
)

export default CommentList
```

PRESENTATIONAL COMPONENT

- src/components/CommentList.test.js

```
describe('<CommentList />', () => {
  it('render correct ids', () => {
    const props = {
      comments: [
        { title: 'title1', body: 'body1', author: 'author1', postat: '1m ago' },
        { title: 'title2', body: 'body2', author: 'author2', postat: '2m ago' },
      ],
      loadMore: () => {},
    }

    const wrapper = shallow(<CommentList {...props} />)
    expect(wrapper.find('#comment-list')).toHaveLength(1)
    expect(wrapper.find('#load-more')).toHaveLength(1)
  })
})
```

PRESENTATIONAL COMPONENT

- src/components/CommentList.test.js

```
describe('<CommentList />', () => {
  it('render comments correctly', () => {
    const props = {
      comments: [
        { title: 'title1', body: 'body1', author: 'author1', postat: '1m ago' },
        { title: 'title2', body: 'body2', author: 'author2', postat: '2m ago' },
      ],
      loadMore: () => {},
    }

    const wrapper = shallow(<CommentList {...props} />)
    expect(wrapper.find('CommentItem')).toHaveLength(2)
  })
})
```

- src/components/CommentList.test.js

```
describe('<CommentList />', () => {
  it('render correct ids', () => {
    const props = {
      comments: [
        { title: 'title1', body: 'body1', author: 'author1', postat: '1m ago' },
        { title: 'title2', body: 'body2', author: 'author2', postat: '2m ago' },
      ],
      loadMore: () => {},
    }

    const wrapper = shallow(<CommentList {...props} />)
    expect(wrapper.find('#comment-list')).toHaveLength(1)
    expect(wrapper.find('#load-more')).toHaveLength(1)
  })

  it('render comments correctly', () => {
    const props = {
      comments: [
        { title: 'title1', body: 'body1', author: 'author1', postat: '1m ago' },
        { title: 'title2', body: 'body2', author: 'author2', postat: '2m ago' },
      ],
      loadMore: () => {},
    }

    const wrapper = shallow(<CommentList {...props} />)
    expect(wrapper.find('CommentItem')).toHaveLength(2)
  })
})
```

- src/components/CommentList.test.js

```
describe('<CommentList />', () => {
  it('render correct ids', () => {
    const props = {
      comments: [
        { title: 'title1', body: 'body1', author: 'author1', postat: '1m ago' },
        { title: 'title2', body: 'body2', author: 'author2', postat: '2m ago' },
      ],
      loadMore: () => {},
    }

    const wrapper = shallow(<CommentList {...props} />)
    expect(wrapper.find('#comment-list')).toHaveLength(1)
    expect(wrapper.find('#load-more')).toHaveLength(1)
  })

  it('render comments correctly', () => {
    const props = {
      comments: [
        { title: 'title1', body: 'body1', author: 'author1', postat: '1m ago' },
        { title: 'title2', body: 'body2', author: 'author2', postat: '2m ago' },
      ],
      loadMore: () => {},
    }

    const wrapper = shallow(<CommentList {...props} />)
    expect(wrapper.find('CommentItem')).toHaveLength(2)
  })
})
```

TEXT

JEST LIFECYCLE

- beforeEach
- afterEach
- beforeAll
- afterAll

JEST LIFECYCLE

```
beforeEach(() => {
  console.log('happen before every tests run')
})
```

PRESENTATIONAL COMPONENT

- src/components/CommentList.test.js

```
describe('<CommentList />', () => {
  const props = {
    comments: [
      { title: 'title1', body: 'body1', author: 'author1', postat: '1m ago' },
      { title: 'title2', body: 'body2', author: 'author2', postat: '2m ago' }
    ],
    loadMore: () => {},
  }
  let wrapper

  beforeEach(() => {
    wrapper = shallow(<CommentList {...props} />)
  })

  it('render correct ids', () => {
    expect(wrapper.find('#comment-list')).toHaveLength(1)
    expect(wrapper.find('#load-more')).toHaveLength(1)
  })

  it('render comments correctly', () => {
    expect(wrapper.find('CommentItem')).toHaveLength(2)
  })
})
```

TEXT

PRESENTATIONAL COMPONENT

- src/components/CommentList.test.js

```
it('match its snapshot', () => {
  expect(wrapper.html()).toMatchSnapshot()
})
```

PRESENTATIONAL COMPONENT

- src/components/CommentList.test.js

```
const props = {
  comments: [
    { title: 'title1', body: 'body1', author: 'author1', postat: '1m ago' },
    { title: 'title2', body: 'body2', author: 'author2', postat: '2m ago' }
  ],
  loadMore: jest.fn(),
}
```

PRESENTATIONAL COMPONENT

- src/components/CommentList.test.js

```
it('call loadMore function when click loadmore button', () => {
  wrapper.find('#load-more').simulate('click')
  expect(props.loadMore).toHaveBeenCalledTimes(1)
})
```

CONTAINER COMPONENT

- Define how things work
- Usually don’t have markup and styles
- Provide data and behavior to other component

TEXT

HOW TO TEST

- Initial state
- State behavior
- Child component
- Lifecycle
- Map state and dispatch from redux

- src/containers/CommentList.js

```
import React, { Component } from 'react'
import { connect } from 'react-redux'
import CommentList from '../components/CommentList'
import { getComments } from '../actions/comments'

export class CommentListContainer extends Component {
  componentDidMount() {
    this.props.getComments()
  }

  render() {
    return (
      <CommentList
        comments={this.props.comments}
        loadMore={this.props.getComments}
      />
    )
  }
}

const mapStateToProps = state => ({
  comments: state.comments,
})

const mapDispatchToProps = dispatch => ({
  getComments: () => dispatch(getComments()),
})

export default connect(
  mapStateToProps,
  mapDispatchToProps,
)(CommentListContainer)
```

PRESENTATIONAL COMPONENT

- src/containers/CommentList.test.js

```
import React from 'react'
import { shallow } from 'enzyme'
import { CommentListContainer } from './CommentList'

describe('<CommentListContainer />', () => {
  it('contain CommentList component', () => {
    const wrapper = shallow(<CommentListContainer />)
    expect(wrapper.find('CommentList')).toHaveLength(1)
  })
})
```

PRESENTATIONAL COMPONENT

- src/containers/CommentList.test.js

```
import React from 'react'
import { shallow, mount } from 'enzyme'
import { Provider } from 'react-redux'
import configureMockStore from 'redux-mock-store'
import { CommentListContainer, mapDispatchToProps, mapStateToProps } from './CommentList'

describe('<CommentListContainer />', () => {
  it('call getComments once after mount', () => {
    const props = {
      comments: [],
      getComments: jest.fn(),
    }

    const createStore = configureMockStore()
    const store = createStore({})
    const wrapper = mount(
      <Provider store={store}>
        <CommentListContainer {...props} />
      </Provider>
    )

    expect(props.getComments).toHaveBeenCalledTimes(1)
  })
})
```

PRESENTATIONAL COMPONENT

- src/containers/CommentList.test.js

```
it('subscribe state correctly', () => {
  const state = {
    comments: [
      { title: 'title1', body: 'body1', author: 'author1', postat: '1m' },
      { title: 'title2', body: 'body2', author: 'author2', postat: '2m' },
    ],
  }
  const subscribeState = mapStateToProps(state)
  expect(subscribeState).toEqual({
    comments: [
      { title: 'title1', body: 'body1', author: 'author1', postat: '1m' },
      { title: 'title2', body: 'body2', author: 'author2', postat: '2m' },
    ],
  })
})
```

PRESENTATIONAL COMPONENT

- src/containers/CommentList.test.js

```
import thunk from 'redux-thunk'
import nock from 'nock'

it('map dispatch to props correctly', () => {
  const createStore = configureMockStore([thunk])
  const store = createStore({})
  const { getComments } = mapDispatchToProps(store.dispatch)
  nock('http://localhost:3000/').get('/comments').reply(200, [])
  const expectedActions = [{ comments: [], type: GET_COMMENTS_SUCCESS }]

  getComments().then(() => {
    expect(store.getActions()).toEqual(expectedActions)
  })
})
```

REDUCER

- Define how the application's state changes
- (previousState, action) => newState

REDUCER

- src/reducers/comments.js

```
const comments = (state = [], action) => {
  switch (action.type) {
    case GET_COMMENTS_SUCCESS:
      return [...state].concat(action.comments)

    case GET_COMMENTS_FAILED:
      return []

    default:
      return state
  }
}
```

REDUCER

- src/reducers/comments.test.js

```
import { comments } from './comments'

describe('comments reducer', () => {
  const initialState = []

  it('when action type is GET_COMMENTS_SUCCESS', () => {
    const action = {
      type: 'GET_COMMENTS_SUCCESS',
      comments: [
        { title: 'title1', body: 'body1' }
      ]
    }
    const expected = [
      { title: 'title1', body: 'body1' }
    ]

    const actual = comments(initialState, action)
    expect(actual).toEqual(expected)
  })
})
```

REDUCER

- src/reducers/comments.test.js

```
it('when action type is GET_COMMENTS_FAILED', () => {
  const action = {
    type: 'GET_COMMENT_FAILED',
  }
  const expected = []

  const actual = comments(initialState, action)
  expect(actual).toEqual(expected)
})
```

REDUCER

- src/reducers/comments.test.js

```
  it('when action type is OTHER_ACTION_TYPE', () => {
    const action = {
      type: 'OTHER_ACTION_TYPE',
    }

    const actual = comments(initialState, action)
    expect(actual).toEqual(initialState)
  })
```

TEXT

ACTION

- Object describing what happened
- Action creator - function that create action
- Async action creator

ACTION

- src/actions/comments.js

```
const receiveComments = comments => ({
  type: GET_COMMENTS_SUCCESS,
  comments: comments,
})

const getComments = () => (
  dispatch => (
    fetch('http://localhost:3000/comments')
    .then(res => res.json())
    .then(comments => (dispatch(receiveComments(comments))))
    .catch(() => (dispatch({ type: GET_COMMENTS_FAILED })))
  )
)
```

ACTION

- src/actions/comments.test.js

```
import { receiveComments, getComments } from './comments'

describe('comments action', () => {
  const comments = [
    { title: 'title1', body: 'body1', author: 'author1', postat: '1m' },
    { title: 'title2', body: 'body2', author: 'author2', postat: '2m' },
  ]

  it('receiveComments action creator return correct action', () => {
    const expected = { type: 'GET_COMMENTS_SUCCESS', comments: comments }

    const actual = receiveComments(comments)

    expect(actual).toEqual(expected)
  })
})
```

ACTION

- src/actions/comments.test.js

```
import configureMockStore from 'redux-mock-store'
import thunk from 'redux-thunk'
import nock from 'nock'
import { receiveComments, getComments } from './comments'

describe('comments action', () => {
  const comments = [
    { title: 'title1', body: 'body1', author: 'author1', postat: '1m' },
    { title: 'title2', body: 'body2', author: 'author2', postat: '2m' },
  ]

  afterEach(() => {
    nock.cleanAll()
  })

  it('getComments dispatch action correctly', () => {
    const mockStore = configureMockStore([thunk])
    const store = mockStore({ comments: [] })
    nock('http://localhost:3000/').get('/comments').reply(200, comments)

    store.dispatch(getComments()).then(() => {
      expect(store.getActions()).toEqual([{ type: 'GET_COMMENTS_SUCCESS', comments: comments }])
    })
  })
})
```

STORE

- Object storage
- Access initial state by getState()

HOW TO TEST

- Correct initial state

TEXT

STORE

- src/store/index.js

```
import { createStore, applyMiddleware } from 'redux'
import thunk from 'redux-thunk'
import rootReducer from '../reducers'

const store = createStore(rootReducer, applyMiddleware(thunk))

export default store
```

STORE

- src/store/index.test.js

```
import store from './index'

describe('store', () => {
  it('get initial state correctly', () => {
    const actual = store.getState()
    expect(actual).toEqual({ comments: [] })
  })
})
```

FEEDBACK

- https://docs.google.com/forms/d/1jTsTEJDkB5NyAOhE3JHKwfkMHL2aI2Q9YOKPnXneAbM