Тестирование React и Redux

|  |  |  |  |
| --- | --- | --- | --- |
| № урока: | 7 | Курс: | React Advanced |
|  | | |  |
| Средства обучения: | | | Текстовый редактор или IDE, браузер, Node.js, терминал |

# Обзор, цель и назначение урока

В этом уроке вы познакомитесь с основными инструментами для тестирования приложений на React и Redux, после чего сможете покрывать тестами любые участки вашего приложения.

**Изучив материал данного занятия, учащийся сможет:**

* Знать об основных инструментах тестирования и уметь их применять.
* Тестировать React-компоненты.
* Тестировать Redux staff: actions, reducers.

# Содержание урока

1. Обзор основных инструментов для тестирования
2. Тестирование React-компонентов: полное и юнит.
3. Тестирование React-компонентов с подключенным Redux.
4. Тестирование actions и reducers.

# Резюме

**Инструменты для тестирования React/Redux.**

Необходимый минимум для тестирования наших приложений:

1. Testing Framework
2. Assert library
3. Helper library

Фреймворк для тестирования является основным инструментом, который умеет находить тест-файлы по заданному паттерну (как правило это *Filename.test.js*), запускать их и показывать результат в консоль.

Assert-библиотека нужна нам для того, чтобы делать какие-то утверждения, от результата которых будет зависеть прошел наш тест либо же нет.

И библиотека-хелпер нужна нам для тестирования React-компонентов – в такой библиотеке есть весь необходимый API для работы с DOM, state, props, ref и т.д.

**Фреймворков для тестирования** очень много, но все они по-сути имеют схожий функционал, поэтому выбор конкретного фреймворка уже дело вкуса.

Некоторые фреймворки лучше подходят для каких-то определенных вещей, например qUnit отлично подходит для тестирования jQuery, но при этом используя какой-либо другой фреймворк особого дискомфорта вы не ощутите. К тому же фреймворки с большим количество комьюнити гарантируют вам, что если вы столкнетесь с какой-то проблемой, то вероятнее всего пути ее решения уже есть.

Список самых популярных тест-фреймворков:

1. Mocha
2. Karma
3. Jasmine
4. Jest
5. AVA
6. И другие: tape, sinon, qUnit, buster etc.

**Assert-библиотек** тоже несметное количество, но они тоже функционалом сильно не разнятся.

Вот принцип, по которым работают все assert-библиотеки:

equal( sum(3, 7), 10 )

т.е мы ожидаем, что если в функцию сложения передать аргументы 3 и 7, то мы получим число 10.

Единственное, чем отличаются эти библиотеки – это стиль синтаксиса.

Есть три основных стиля:

1: Assert ( .isTrue, .isFunction … )

2: Expect ( .toBe, .toEqual, .toHaveLength … )

3. Should ( .shouldHave, .should.fail )

В Node.JS есть встроенный Assert - <https://nodejs.org/api/assert.html>, который в большинстве случаев отлично подходит и его можно использовать без подключения сторонних библиотек

Но также вы можете подключить и другие:

1. <http://chaijs.com/>

2. <https://github.com/mjackson/expect>

3. <https://shouldjs.github.io/>

Chai.Js является наиболее популярной, т.к. в ней есть 3 стиля assertions.

В кач-ве helper-библиотеки мы попробуем React-Test-Utils от Facebook и Enzyme от Airbnb.

Enzyme на самом деле это абстракция и синтаксический сахара и «под капотом» он использует React-Test-Utils и библиотеку “cheerio”, за счет чего работать с DOM гораздо удобнее (как в jQuery) + количество бойлерплейта сравнительно меньше.

В нашем уроке мы будет использовать Jest в кач-ве основного фреймворка для тестирования, потому что он имеет ряд таких преимуществ:

* встроен в create-react-app
* простой
* распараллеливание
* возможность тестировать асинхронщину
* умеет «мокать»
* есть свой assert
* есть интерактивный watch-mode
* умеет собирать coverage (покрытие тестами)
* Snapshots

И в кач-ве хелпер-библиотеки мы попробуем как React-Test-Utils, т.к. и Enzyme, но основным выбором будет Enzyme, потому что:

* сравнительно мало бойлерплейт-кода
* jQuery like syntax
* есть shallow/full рендеринг
* Отлично подходит для React-компонентов

# Закрепление материала

* Чем отличаются full rendering от shallow rendering?
* Как протестировать компонент отдельно от Redux, если он обернут в connect?
* Почему тестирование каноничных actions и reducers очень простое?
* Какие преимущества дает «мок» функций?
* Как тестировать асинхронные actions?

# Самостоятельная деятельность учащегося

Найдите на гитхабе любое приложение на React+Redux и попробуйте написать к нему различные тесты (особенно будет хорошо, если вы найдете приложение с асинхронными экшенами).

# Рекомендуемые ресурсы

Jest – getting started

<https://facebook.github.io/jest/docs/en/getting-started.html>

Unit testing with Jest: Redux + async

<https://medium.com/@ferrannp/unit-testing-with-jest-redux-async-actions-fetch-9054ca28cdcd>

Тестирование React

<https://habrahabr.ru/company/infowatch/blog/315760/>

Testing Redux – manual

<https://github.com/reactjs/redux/blob/master/docs/recipes/WritingTests.md>