# Here's the plan

**Testing Technologies**
- Testing frameworks
- Assertion libraries
- Helper libraries

**We'll test**
- React presentation components

# Testing Frameworks

# Test Frameworks

**Mocha**

**Jasmine**

**Jest**

**Tape**

**AVA**

|                Mocha                | AVA                             |
| ----------------------------------: | ------------------------------- |
|                              Serial | Concurrent                      |
|                No assertions built in | Assertions built in           |
|                         Uses globals | No globals                     |
|              No built-in ES6 support | Built-in ES6 support           |
|                  Runs all tests on save | Runs only impacted tests     |
|           Long stack trace upon error | Short specific error with marker |
| Proven, mature, with huge ecosystem | New                            |

# Test Frameworks



**Cory House**
@housecor

What JavaScript test framework should I use in my new "React and Redux in ES6" Pluralsight course?

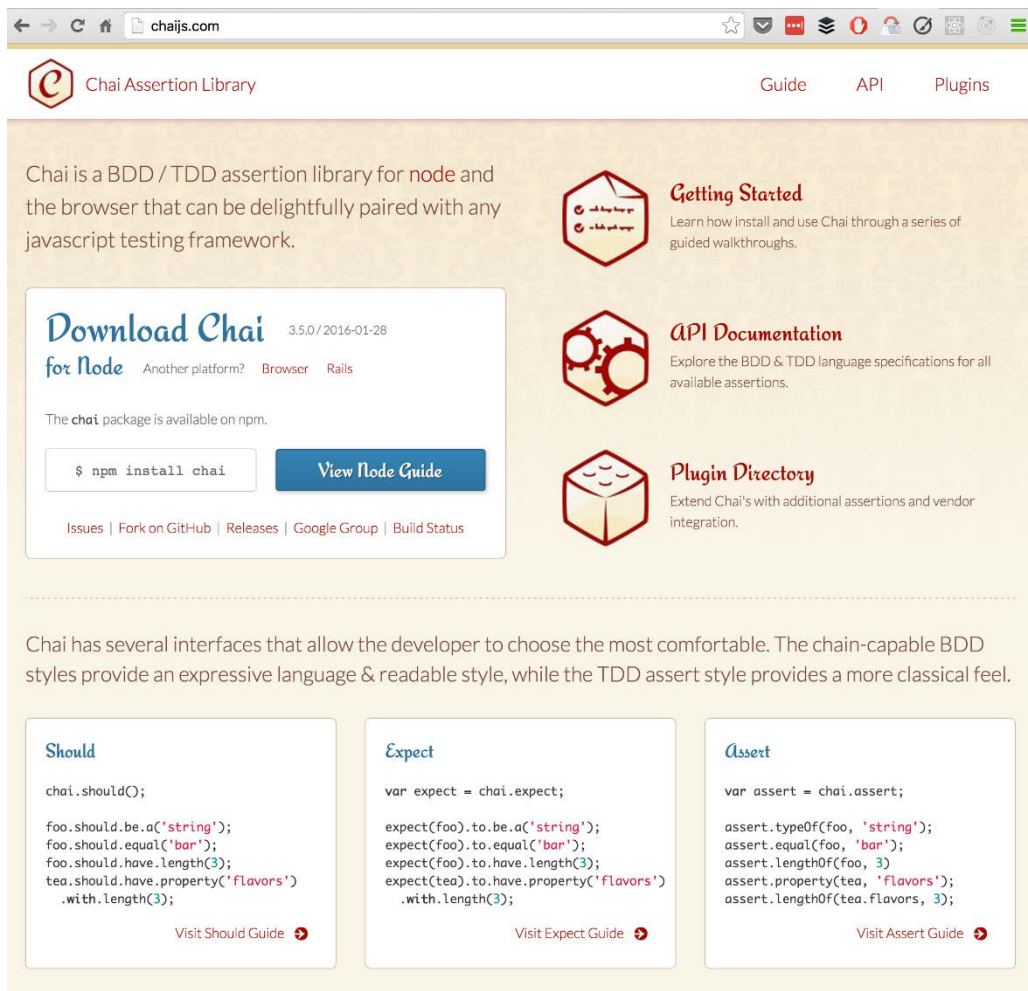| | |
|---|---|
| **59%** Mocha | |
| **13%** AVA | |
| **19%** Tape | |
| **9%** Jest | |

78 votes • Final results

# Assertion Libraries

**npm**    find packages    🔍    Greetings, housecor ⌄

# ⭐ expect `public`
## Write better assertions

**expect** lets you write better assertions.

When you use `expect`, you write assertions similarly to how you would say them, e.g. "I expect this value to be equal to 3" or "I expect this array to contain 3". When you write assertions in this way, you don't need to remember the order of actual and expected arguments to functions like `assert.equal`, which helps you write better tests.

You can think of `expect` as a more compact alternative to **Chai** or **Sinon.JS**, just without the pretty website. ;)

## Installation

Using **npm**:

```
$ npm install --save expect
```

Then, use as you would anything else:

```
// using ES6 modules
import expect, { createSpy, spyOn, isSpy } from 'expect'

// using CommonJS modules
var expect = require('expect')
var createSpy = expect.createSpy
var spyOn = expect.spyOn
var isSpy = expect.isSpy
```

The UMD build is also available on **npmcdn**:

```
<script src="https://npmcdn.com/expect/umd/expect.min.js"></script>
```

You can find the library on `window.expect`.

---

### Working with private modules

With npm private modules, you can use the npm registry to host your own private code and the npm command line to manage it. **Learn more…**

---

⬇ `npm install expect`

**how? learn more**

---

👤 **mjackson** published a week ago

---

**1.18.0** is the latest of 34 releases

---

**github.com/mjackson/expect**

---

**MIT** 🔓®

---

### Collaborators

---

### Stats

**7,548** downloads in the last day

**42,701** downloads in the last week

**180,647** downloads in the last month

**5 open issues** on GitHub

**2 open pull requests** on GitHub

| Chai | Expect |
|---|---|
| to.equal | toBe |
| to.deep.equal | toEqual |
| to.exist | toExist |
| to.not.exist | toNotExist |
| to.be.above | ToBeGreaterThan |
| | |
| No spy | Spy built in |

# Helper Libraries

# Helper Libraries

**React Test Utils**

**Enzyme**

# React Test Utils
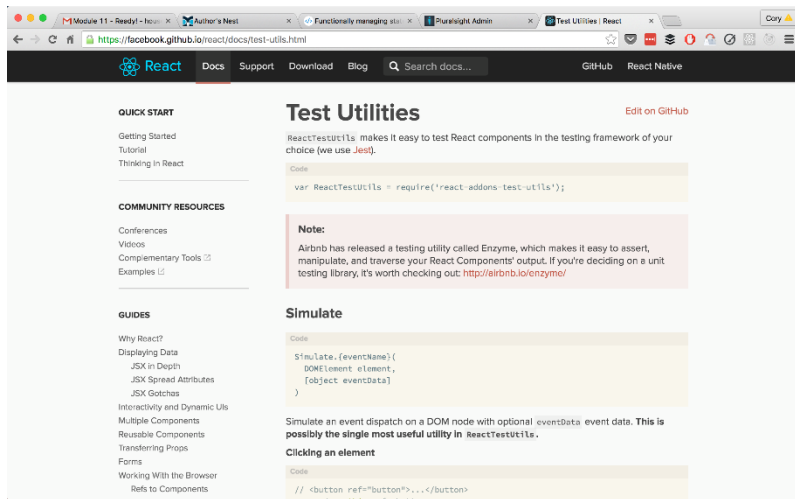
React testing library

Built by Facebook

Verbose API

# React Test Utils: Two Rendering Options

| shallowRender | renderIntoDocument |
|---|---|
| Render single component | Render component and children |
| No DOM Required | DOM Required |
| Fast and Simple | Supports simulating interactions |

# React Test Utils: DOM Interactions

- **findRenderedDOMComponentWithTag**

- **scryRenderedDOMComponentsWithTag**

- **Simulate**
  - **Clicks**
  - **Keypresses**
  - **Etc.**

Much more: [facebook.github.io/react/docs/test-utils.html](facebook.github.io/react/docs/test-utils.html)

React   Docs   Support   Download   Blog   🔍 Search docs...   GitHub   React Native

# Test Utilities

Edit on GitHub

`ReactTestUtils` makes it easy to test React components in the testing framework of your choice (we use Jest).

**QUICK START**

Getting Started
Tutorial
Thinking in React

**Code**

```
var ReactTestUtils = require('react-addons-test-utils');
```

**COMMUNITY RESOURCES**

Conferences
Videos
Complementary Tools ⧉
Examples ⧉

> **Note:**
>
> Airbnb has released a testing utility called Enzyme, which makes it easy to assert, manipulate, and traverse your React Components' output. If you're deciding on a unit testing library, it's worth checking out: http://airbnb.io/enzyme/

**GUIDES**

Why React?
Displaying Data
    JSX in Depth
    JSX Spread Attributes
    JSX Gotchas
Interactivity and Dynamic UIs
Multiple Components
Reusable Components
Transferring Props
Forms
Working With the Browser
    Refs to Components

## Simulate

**Code**

```
Simulate.{eventName}(
  DOMElement element,
  [object eventData]
)
```

Simulate an event dispatch on a DOM node with optional `eventData` event data. **This is possibly the single most useful utility in `ReactTestUtils`.**

**Clicking an element**

**Code**

```
// <button ref="button">...</button>
```

# Why Enzyme?

**React Test Utils**                              **Enzyme**

**findRenderedDOMComponentWithTag**               **find**

**scryRenderedDOMComponentsWithTag**              **find**

**scryRenderedDOMComponentsWithClass**            **find**

# Enzyme Is an Abstraction



**Behind the scenes**
- React Test Utils
- JSDOM (In-memory DOM)
- Cheerio (Fast jQuery style selectors)

# Comparison

| | React Test Utils | JSDOM | Cheerio | Enzyme |
|---|---|---|---|---|
| Run tests in | Node | Node | Node | Node |
| Dependencies | None | None | None | JSDOM, Cheerio, React Test Utils |
| Selectors | Wordy | DOM | jQuery | Terse |
| Interactions | Yes | Yes | No | Yes |
| Ease | Clunky | Okay | Easy | Easy |
| Speed | Varies | Okay | Fast | Varies |

# Where to Test

# Where to Test

```
Author Actions
  ✓ should create CREATE_AUTHOR_SUCCESS action

Async Actions
  ✓ should create LOAD_AUTHORS_SUCCESS when authors have been loaded (1006ms)

Course Actions
  ✓ should create a END_CREATE_COURSE action

Async Actions
  ✓ should create END_LOAD_COURSES when courses have been loaded (1006ms)

AJAX Call Status Reducer
  ✓ should increment the number of calls in progress
  ✓ should decrement the number of calls in progress when any action ending in
  ✓ should decrement the number of calls in progress when API_CALL_ERROR is di

Author Reducer
  ✓ should add author
  ✓ should create a new object when creating a new author
  ✓ should remove author
  ✓ should update author

Course Reducer
  ✓ should add course
  ✓ should create a new object when creating a new course
  ✓ should remove course
  ✓ should update course
```

**Browser**

**Headless Browser**

**In-memory DOM**

# Naming Test Files

**Cory House**
@housecor

How do you prefer to name your #JavaScript test files?

46%  fileName.spec.js

39%  fileName.test.js

15%  Other - Please reply

180 votes • 2 hours left

# Where Do Test Files Belong?

**Mocha default: /test**

**We'll place tests alongside the file under test. Why?**

- Easy imports. Always ./filenameUnderTest
- Clear visibility
- Convenient to open
- Move files and tests together

# Our Plan

| | |
|---|---|
| **What** | **React components and Redux** |
| **How** | **Mocha with Expect** |
| **Where** | **In-memory DOM via JSDOM** |
| **Helper** | **Enzyme** |

# Testing Presentation Components

# Rendering Options

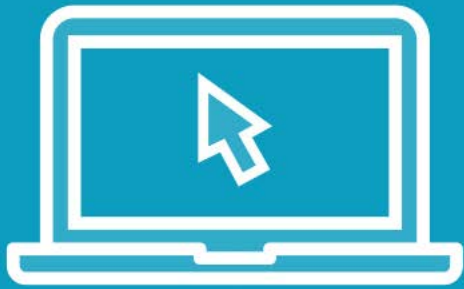| Shallow Render | Render into Document |
| --- | --- |
| Fast | Slower |
| Test in isolation | Test set of components |
| No refs or interactions yet | Test refs and interactions |
| Recommended for the future | |
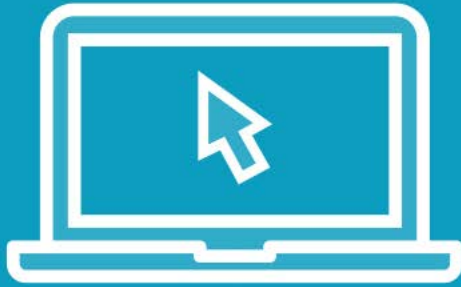
# Demo

**Test Component with Shallow Rendering**
- React Test Utils
- Enzyme

# Demo

**Enzyme**

# Wrap up

**Testing Approach**

- Mocha, Expect, React Test Utils, and Enzyme

- In-memory DOM via Node

**Tested**

- React presentation components

**Next up: Testing Redux**