Extracted from:

React for Real

Front-End Code, Untangled

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

React for Real

Front-End Code, Untangled

Ludovico Fischer

edited by Brian P. Hogan

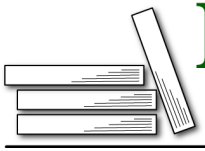
React for Real

Front-End Code, Untangled

Ludovico Fischer

The Pragmatic Bookshelf

Raleigh, North Carolina



Pragmatic Bookshelf

Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Development Editor: Brian P. Hogan
Copy Editor: Nicole Abramowtiz
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-68050-263-3
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—August 2017

Set Up the Test Framework

Let's add some tests to the word counter project. We'll start by installing a test framework that runs our tests and reports the test results. We'll use Jest[2] because it offers an excellent out-of-the-box experience, but you can apply the techniques you'll learn to other test frameworks.

Navigate to the word counter project directory and install Jest as a development dependency:

```
$ npm i --save-dev jest
```

Since your code uses JSX, you also need to transpile the code when you run the tests. A Jest plugin called babel-jest ensures Jest transpiles the test code with the Babel configuration it finds in the package root. Install the babel-jest plugin as a development dependency:

```
$ npm i --save-dev babel-jest
```

We'll use npm scripts to run our tests via the npm command. That way, you'll avoid conflicts between different versions of Jest in different projects.

The jest utility scans the directory for files with names that end in test or spec, or files in a directory named __tests__. It then executes the tests they contain. Since there's no single dominant testing tool for JavaScript, the npm test command is the standard way to run tests in projects managed with npm. Open package.json and modify the test entry in the scripts section in package.json to run the local version of Jest with the npm test command:

wordcounter/package.json

```
"scripts": {
  "test": "jest --watch",
```

The --watch flag lets Jest automatically re-run the tests when you edit a file. To check that your setup works, run:

```
$ npm test
```

Jest reports the regular expression used to find test files. It automatically ignores node_modules. Since you haven't written any tests yet, Jest hasn't run any tests.

```
No tests found
14 files checked.
roots: wordcounter - 14 matches
testMatch: **/__tests__/**/*.js?(x),**/?(*.)(spec|test).js?(x) - 0 matches
testPathIgnorePatterns: /node_modules/ - 14 matches
```

2. https://facebook.github.io/jest/

Once Jest finishes its run, it presents you with a prompt:

```
Watch Usage
› Press a to run all tests.
› Press p to filter by a filename regex pattern.
› Press t to filter by a test name regex pattern.
› Press q to quit watch mode.
› Press Enter to trigger a test run.
```

You don't need to choose any of these options: by default, Jest will start running new tests as soon as you add them.

Now that you have installed Jest and verified that you're able to run it, let's write our first test.