Preface

One of the biggest challenges of many web applications is being supported by different browsers with different versions. JavaScript code that runs on the Safari browser will not necessarily run correctly on Internet Explorer (IE), Firefox, or Google chrome browsers. This challenge is caused by the lack of unit testing of the JavaScript code that has lived in the web application from day one. Without unit testing the JavaScript code, more money will have to be spent for testing and retesting the application's web pages after deciding to upgrade to current, supported browsers (or after updating the JavaScript code of the web pages with non-trivial features).

The *JavaScript Unit Testing* book is a comprehensive practical guide that illustrates in detail how to efficiently create and automate JavaScript tests for web applications using popular, JavaScript unit testing frameworks, such as Jasmine, YUI Test, QUnit, and JsTestDriver.

This book explains the concept of JavaScript unit testing and explores the bits of an interactive Ajax web application (the weather application). Throughout the book, the JavaScript part of the weather application is tested using different JavaScript unit testing frameworks. The book illustrates how to generate test and code coverage reports of developed JavaScript tests. It also explains how to automate the running of JavaScript tests from build and continuous integration tools. The book shows how to integrate different JavaScript unit testing frameworks with each other in order to test web applications in the most efficient way.

What this book covers

Chapter 1, *Unit Testing JavaScript Applications*, helps you understand what unit testing is, the requirements of a good unit test, and why unit testing is needed. You will also learn the difference between Test-Driven Development and traditional unit testing. You will understand the complexities of testing JavaScript code, and the requirements of good, JavaScript unit testing tools. In this chapter, we will explore the weather web application's JavaScript section which we will unit test in the next chapters.

Chapter 2, *Jasmine*, helps you learn what Jasmine is and how to use it for testing synchronous JavaScript code. You will learn how to test asynchronous (Ajax) JavaScript code using the Jasmine Spies, waitsFor, and runs mechanisms. You will learn how to perform mock Ajax testing using Jasmine. You will learn about the various matchers provided by the framework, and how to load HTML fixtures in your Jasmine tests. In this chapter, you will learn how to use Jasmine for testing the weather application's JavaScript section.

Chapter 3, *YUI Test*, helps you to learn what YUI Test is and how to use this JavaScript unit testing framework for testing synchronous JavaScript code. You will learn how to test asynchronous (Ajax) JavaScript code using the YUI Test's wait and resume mechanisms. You will learn about the various assertions provided by the framework, how to display XML and JSON test reports using framework reporter APIs, and how to generate test reports automatically using the YUI Test Selenium Driver. You will learn how to automate running YUI tests using the YUI Test Selenium Driver, and how to integrate an automation script with build management and continuous integration tools. In this chapter, you will learn how to use YUI Test for testing the weather application's JavaScript section.

Chapter 4, *QUnit*, helps you to understand what QUnit is and how to use it for testing synchronous JavaScript code. You will learn how to test asynchronous (Ajax) JavaScript code using the QUnit test mechanism and the QUnit asyncTest mechanism. You will also learn the different assertions provided by the framework, and how to develop your own assertion in order to simplify your test code. You will learn how to load HTML fixtures in your QUnit tests. In this chapter, you will learn how to use the framework for testing the weather application's JavaScript section.

Chapter 5, *JsTestDriver*, helps you to learn what JsTestDriver (JSTD) is, the JSTD architecture, the JSTD configuration, and how to use JSTD for testing synchronous JavaScript code. You will learn how to test asynchronous (Ajax) JavaScript code using the JSTD AsyncTestCase object. You will learn the various assertions provided by the framework, and how to generate test and code coverage reports using the framework's code coverage plugin. You will learn how to use JSTD as a test runner for the other JavaScript unit testing frameworks mentioned in the book, such as

Jasmine and QUnit, in order to enable the execution of the tests of these frameworks from the command-line interface. You will learn how to integrate the tests of JSTD (and the tests of the JavaScript frameworks on top of JSTD) with build and continuous integration tools. You will learn how to work with the JSTD framework in one of the most popular integrated development environments (IDEs) which is Eclipse. In this chapter, you will learn how to use JSTD for testing the weather application's JavaScript section.

What you need for this book

You will need the following software in order to run all of the examples in this book:

- Apache Tomcat 6, which can be found at `http://tomcat.apache.org/download-60.cgi`
- Java Development Kit (JDK) Version 5.0 or later, which can be found at `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
- The Selenium Server version 2.25.0 (for *Chapter 3*, *YUI Test* only), which can be found at `http://seleniumhq.org/download/`
- Eclipse IDE (for *Chapter 5*, *JsTestDriver* only), which can be found at `http://www.eclipse.org/downloads/packages/release/indigo/sr2`

Who this book is for

The target audience for this book is developers, designers, and architects of web applications.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "The `validateLoginForm` function calls the `LoginClient` JavaScript object, which is responsible for validating the login form."

A block of code is set as follows:

```
function validateLoginForm() {
  var loginClient = new weatherapp.LoginClient();
```

```
    var loginForm = {
      "userNameField" : "username",
      "passwordField" : "password",
      "userNameMessage" : "usernameMessage",
      "passwordMessage" : "passwordMessage"
    };

    return loginClient.validateLoginForm(loginForm);
  }
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>QUnit test runner</title>
  <link rel="stylesheet" href="lib/qunit-1.10.0.css">
</head>
<body>
  <div id="qunit"></div>
  <div id="qunit-fixture"></div>
  <script src="lib/qunit-1.10.0.js"></script>

  ...The test code here...
</body>
</html>
```

Any command line input or output is written as follows:

```
java -jar JsTestDriver-1.3.4.b.jar --port 9876 --browser [firefoxpath],
[iepath],[chromepath]
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes, for example, appear in the text like this: "In this application, the user enters his/her name and then clicks on the **Welcome** button."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.PacktPub.com`. If you purchased this book elsewhere, you can visit `http://www.PacktPub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.