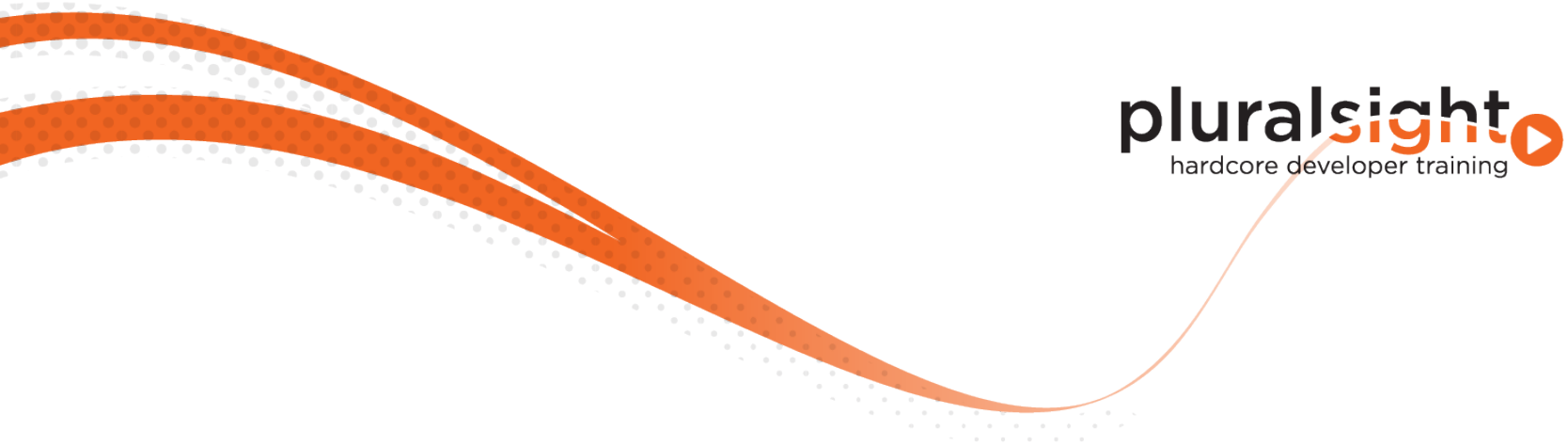
Testing Clientside JavaScript

Joe Eames
@josepheames
www.testdrivenjs.com
www.pluralsight.com



About the Author

- Joe Eames
- @josepheames
- www.testdrivenjs.com
- www.javascriptjabber.com

Agenda

- **Testing Frameworks**
  - QUnit
  - Jasmine
  - Mocha
- **Mocking**
  - Mocking in JavaScript
  - Jasmine Spies
  - Sinon
- **Testing Utilities**

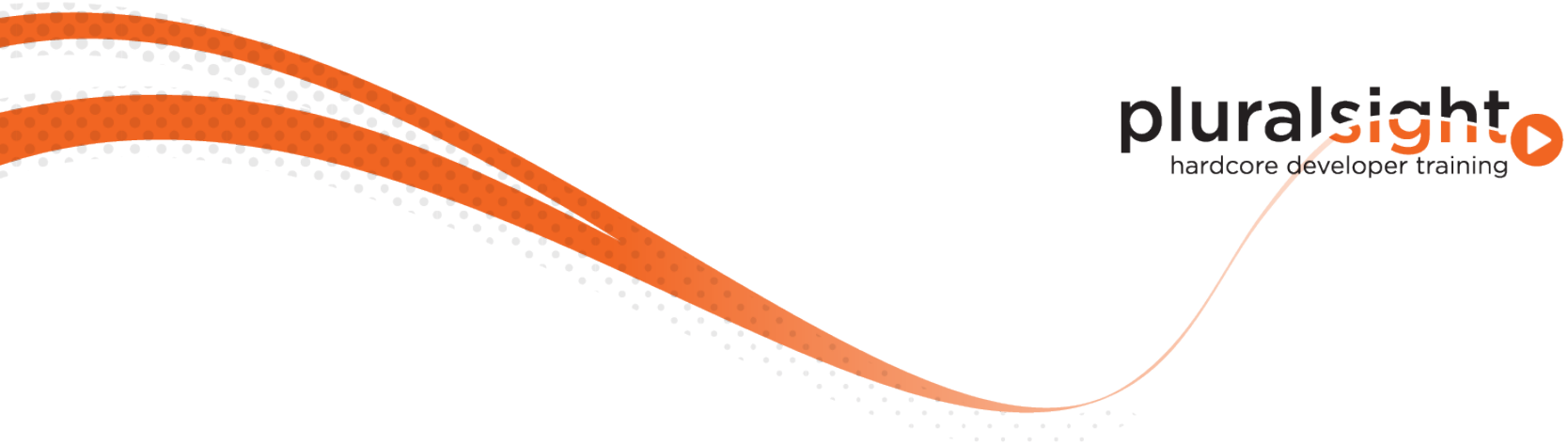
Testing Clientside JavaScript
QUnit

Joe Eames

@josepheames

www.testdrivenjs.com

www.pluralsight.com



pluralsight
hardcore developer training

Overview

- Introduction to Qunit
- Organizing Tests
- Running Tests
- Integrating with the DOM
- Integrating with CI
- Asynchronous Tests
- QUnit Tidbits

Introduction

- Similar to server-side frameworks (JUnit, Nunit)
- Built by the jQuery team
- Used to test jQuery's features
- No dependencies
- Can test server-side JavaScript

Setting Up QUnit

- **Get Source Code**
  - https://github.com/jquery/qunit
- **Create Test File**
- **Create Test Runner File**

Assertions

- ok(state, message)
- equal(actual, expected, message)
- notEqual(actual, expected, message)
- deepEqual(actual, expected, message)
- notDeepEqual(actual, expected, message)
- strictEqual(actual, expected, message)
- notStrictEqual(actual, expected, message)
- raises(actual, expected, message)

Organizing Tests

- **Server-Side Organization**
  - Folders
  - Files
  - Fixtures

Organizing Tests – Folders & Files

```
Project Root
  Scripts
    Module1
      Source1.js
    Module2
      Source2.js
```

```
Project Root
  Scripts
    Module1
      Source1.js
    Module2
      Source2.js
  Tests
    Libs
    Module1
      Source1Tests.js
    Module2
      Source2Tests.js
```

Organizing Tests - Modules

- **Purposes of Modules**
  - Group Tests Logically
  - Group Setup & Teardown
- **Purposes of Common Setup**
  - **Common Objects**
  - **Setting up the DOM**
- **Purposes of Common Teardown**
  - **Cleaning up the DOM**
  - **Generic cleanup**
- **Reasons for Using Multiple Modules**
  - **Logical grouping**
  - **Grouping by component**
  - **Grouping by common setup**

Grouping With Testrunner files

TestFile1.Html
- TestFile1.js
- SourceFile1.js

TestFile2.Html
- TestFile2.js
- SourceFile2.js

TestRunner1.Html
- TestFile1.js
- SourceFile2.js
- SourceFile1.js

TestRunner2.Html
- TestFile2.js
- TestFile3.js
- SourceFile3.js

TestRunner3.Html
- TestFile4.js
- TestFile5.js
- SourceFile4.js
- SourceFile5.js

Running Qunit Tests

1. Running all tests within a test runner file
2. Running a single test
3. Custom filter
4. Composite addon
5. Resharper

Using ReSharper

- **Benefits**
  - Simple to setup
  - Convenient within Visual Studio
- **Drawbacks**
  - Can’t refresh browser
  - Opens a new tab on every run
  - Requires that test files be in the same project as source code
  - Encourages poor project structure
- **Not Recommended**

Integrating QUnit with the DOM

- **Reasons to Test the DOM**
  - Test that the SUT correctly manipulates the DOM
  - Test that the SUT correctly reads from the DOM
- **Drawbacks to DOM testing**
  - Requires additional setup
  - Prone to brittleness

Integrating With CI

- **Basics: Use PhantomJS and capture output**
- **True Browser Testing Needs**
  - Difficult because of deployment (Multiple versions of I.E.)
  - Typically easier to use a 3rd party.

- **URLs**
  - PhantomJS
    - http://code.google.com/p/phantomjs/downloads/list
  - Qunit.teamcity.js
    - https://gist.github.com/1755675

Asynchronous Tests

- **Testing with setTimeout and setInterval**
- **Testing with UI effects**
- **Testing with ajax**
  - Avoid it at all costs
  - Write an abstraction layer
  - Use test doubles

QUnit Tidbits

- **Noglobals setting**
- **Notrycatch setting**
- **Expect() method**
- **Events**
  - Log
  - testStart
  - testDone
  - moduleStart
  - moduleDone
  - Begin
  - done

Summary

- TDD style tests
- Versatile HTML Interface
- Supports asynchronous tests
- Integrates with CI