Writing JavaScript Unit Tests

Writing Tests

Fluid is using jqUnit - a wrapper of QUnit for writing javascript unit tests. There's more about fluid: jqUnit and QUnit below .

There are two parts to writing a javascript test - the test data and the test itself.

On This Page

Creating the test data

Wonderfully, when writing these tests the data is simply the markup in an HTML file. There are a few required parts when creating a test HTML file. As an example, take a look at the InlineEdit-test.html file.

1. The inclusion of the test framework:

```
    <!--  These are the jqUnit test framework files -->
<link rel="Stylesheet" type="text/css" media="screen" href="../../../lib/qunit/css/qunit.css" />
<script type="text/javascript" src="../../../lib/qunit/js/qunit.js"></script>
<script type="text/javascript" src="../../../test-core/jqUnit/js/jqUnit.js"></script>
```

2. The inclusion of the modules that will be tested:

```
    <!--  These are the required javascript modules for the Inline Edit -->
<script type="text/javascript" src="../../../../lib/jquery/core/js/jquery.js"></script>
<script type="text/javascript" src="../../../../lib/jquery/ui/js/jquery.ui.core.js"></script>
<script type="text/javascript" src="../../../../lib/jquery/plugins/delegate/js/jquery.delegate.js"><
/script>
<script type="text/javascript" src="../../../../framework/core/js/jquery.keyboard-a11y.js"></script>
<script type="text/javascript" src="../../../../framework/core/js/Fluid.js"></script>
<script type="text/javascript" src="../../../../components/inlineEdit/js/InlineEdit.js"></script>
<script type="text/javascript" src="../../../../components/undo/js/Undo.js"></script>
```

3. The inclusion of the tests and their supports:

```
    <!--  These are tests that have been written using this page as data as well as test supports -->
<script type="text/javascript" src="../../../lib/jquery-ui/js/jquery.simulate.js"></script>
<script type="text/javascript" src="../js/InlineEditTests.js"></script>
```

4. Markup that QUnit looks for when running the tests and displaying the test results:

```

<h1 id="qunit-header">InlineEdit Test Suite</h1>
<h2 id="qunit-banner"></h2>
<div id="qunit-testrunner-toolbar"></div>
<h2 id="qunit-userAgent"></h2>
<ol id="qunit-tests"></ol>
<div id="main">
    ...
</div>
```

5. Markup that the tests use as data - everything inside the div with the id 'main'

Creating a test

The tests themselves should live in a separate javascript file that is linked to in the HTML file. There are a few required parts when creating a test javascript file. As an example, take a look at the InlineEditTests.js file.

1. Wrap everything in a call to 'ready':

```
jQuery(document).ready (function () {
    ...
});
```

2. Create a new test case:

```
var inlineEditTests = new jqUnit.TestCase ("InlineEdit Tests");
```

3. Create:
    - tests and add them to the test case:

```
inlineEditTests.test("Minimal Construction", function () {
    ...
});
```

    - asynchronous tests and add them to the test case:

```
inlineEditTests.asyncTest("Minimal Construction", function () {
    ...
});
```

InlineEditTests.js uses the jqUnit API for its tests.

Running the tests

To run the tests, open up the HTML file in the browsers that you are testing. Failing tests will be red. Clicking on a failing test will show you the details of the test.

Background on QUnit and jqUnit

jQuery has a testing framework called QUnit http://docs.jquery.com/QUnit. Fluid uses a little wrapper for QUnit called jqUnit which allows us to write tests using a jUnit style API. The main difference between the QUnit and jqUnit are the names of the test functions and the order of parameters. In QUnit, the names are short and simple such as 'equals' and the order of parameters is "actual, expected, message". In jqUnit the names begin with assert such as 'assertEquals' and the order is "message, expected, actual". The wrapper was created (thanks Colin) because we had lots of jUnit style tests prior to using QUnit and we wanted to port them with the least amount of work. It's now up to each developer whether they want to use the QUnit API or the jqUnit API. People coming from a background of writing jUnit tests in java tend to prefer the jqUnit API simply because of habit and you will notice that most of the unit tests in Fluid use that format.