Unit-Tests für JavaScript und AJAX

Peter Krenn

9. Januar 2011

Zusammenfassung

Zusammenfassung

Inhaltsverzeichnis

1 Einleitung

2 QUnit

QUnit[5] wird von Mitgliedern des *jQuery* Teams[3] entwickelt und ist auch die offizielle *test suite* des *jQuery* Projektes. Es zählt zur Gruppe der *xUnit test frameworks*[2], die einem Design folgen, das erstmals mit *SUnit*[1] für *Smalltalk* formuliert wurde und als *JUnit* für *Java* große Verbreitung fand.

2.1 Assertions

Der zentrale Bestandteil eines *unit tests* ist die *assertion*. Eine *assertion* ist eine Aussage über den Zustand einer bestimmten Stelle in einem Programm[4]. Der

Entwickler macht eine Zusicherung, dass eine bestimmte Bedingung unabhänging von den Laufzeitumständen immer wahr ist.

QUnit stellt eine Reihe von allgemeinen *assertions* zur Verfügung, die *JUnit* nachempfunden sind, und auch solche, die speziell an JavaScript und asynchrone Entwicklung angepasst sind[5]:

`ok(state, message)`

`ok()` ist eine boolsche *assertion*, die positiv terminiert, wenn der erste Parameter wahr ist. Den zweiten, optionalen Parameter `message` haben alle *assertion*-Funktionen gemein: dieser wird gemeinsam mit dem Ergebnis ausgegeben.

```
ok(true, "terminiert immer positiv");
ok("", "Ein leerer string entspricht false, die assertion schlägt fehl");
```

`equal(actual, expecteded, message)`

Diese *assertion* ist `ok()` sehr ähnlich, allerdings werden die `actual` und `expecteded` Werte mit dem Ergebnis ausgegeben. Dadurch werden die Tests aussagekräftiger und *debugging* einfacher. `equal` terminiert positiv, wenn `actual == expecteded`.

```
var actual = 1;
equal(actual + 1, 2, "terminiert positiv");
equal(actual, 2, "schlägt fehl");
```

`strictEqual(actual, expected, message)`

Im Gegensatz zu `equal()` werden bei dieser *assertion* die Werte mit `actual === expecteded` verglichen. Dieser Operator vergleicht zusätzlich die Typengleichheit der beiden Parameter.

```
var actual = 0;
equal(actual, false, "terminiert positiv, da == den Datentyp ignoriert");
strictEqual(actual, false, "schlägt fehl, da 0 Number und false boolean ist");
```

`deepEqual(actual, expected, message)`

`deepEqual()` überprüft primitve Datentypen, *arrays* und Objekte rekursiv auf Gleichheit. Sie ist dabei auch strikter als `equal()`, da der === Operator verwendet wird.

```
var actual = {a: 1};
equal(actual, {a: 1}, "schlägt mit verschiedenen Objekten fehl");
deepEqual(actual, {a: "1"}, "schlägt fehl, da die Datentypen verschieden sind");
deepEqual(actual, {a: 1}, "terminiert positiv, da die Inhalte der Objekte
gleich sind");
```

`notEqual(actual, expected, message)`

Diese *assertion* ist die invertierte Version von `equal()`. Entsprechende Funktionen existieren auch für `strictEqual()` und `deepEqual()`: `notStrictEqual()` und `notDeepEqual`.

```
var actual = 0;
notEqual(actual, false, "schlägt fehl");
notStrictEqual(actual, false, "terminiert positiv");

var actual = {a: 1};
notDeepEqual(actual, {a: "1"}, "terminiert positiv");
```

`raises(state, message`

`raises()` überprüft, ob die übergebene *callback* Funktion eine *exception* wirft. Die Funktion wird ohne Parameter im *default scope* aufgerufen.

```
raises(function() {
  throw new Error("Fehlerfall");
}, "terminiert positiv, falls der gewünschte Fehler auftritt");
```

2.2 Testfunktionen

Werden die vorgestellten *assertions* direkt ausgeführt, unterbrechen sie im Falle eines Fehlers den Testdurchlauf. Deswegen werden sie in Testfunktionen eingebettet.

Es ist auch üblich, dass jede Testfunktion nur eine spezifische Eigenschaft testet. Dies geschieht oft mit mehreren *assertions* und die Testfälle bleiben dabei überschaubar und einfach zu verstehen.

QUnit macht das mit der Funktion `test(name, expected, test)`. `name` ist eine Bezeichnung des Tests und `test` ist eine Funktion, die die *assertions* enthällt. `expect` ist optional und erlaubt es die Anzahl der zu erwartenden *assertions* festzulegen. Dies ist bei asynchronen Tests von Bedeutung.

2.3 Test Suites

Mit der `module(name, lifecycle)` Anweisung ist es möglich, die Testfälle weiter in Module aufzuteilen. Dies dient einerseits der Strukturierung der Tests und andererseits wird es auch möglich, für mehrere Testfälle relevante Daten vor deren Ausführung zu definieren.

Im Parameter `lifecycle` kann man optional die Funktionen `setup()` und `tearDown()` übergeben, die vor beziehungsweise nach jedem Test ausgeführt werden. Sie teilen den `this`-*scope* der Testfunktionen, wodurch sogenannte *fixtures* erstellt werden können. Die Daten werden nach jedem Test zurückgesetzt, was in `tearDown()` auch manuell geschehen kann.

Es stellt die Dateien `qunit.js` und `qunit.css` zur Verfügung, die gemeinsam mit dem zu testenden JavaScript Code und den Tests in eine HTML-Datei eingebunden werden müssen.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script type="text/javascript" src="qunit.js"></script>
    <link rel="stylesheet" href="qunit.css" type="text/css" media="screen" />

    <!-- project file -->
    <script type="text/javascript" src="my_project.js"></script>

    <!-- tests file -->
    <script type="text/javascript" src="my_tests.js"></script>
  </head>
  <body>
    <h1 id="qunit-header">QUnit example</h1>
    <h2 id="qunit-banner"></h2>
    <div id="qunit-testrunner-toolbar"></div>
    <h2 id="qunit-userAgent"></h2>
    <ol id="qunit-tests"></ol>
    <div id="qunit-fixture">test markup, will be hidden</div>
  </body>
</html>
```

Zum Ausführen der Tests muss diese Datei dann in einem Web-Browser geöffnet werden. *QUnit* befüllt die Elemente im `body` mit den Ergebnissen der Tests.

Wie auch bei anderen *xUnit test frameworks* sind die zentralen Elemente eines *QUnit* Tests die *assertions*.

Literatur

[1] Kent Beck. Simple smalltalk testing: With patterns. *Smalltalk Report*, 4(2):16–18, 1994.

[2] Martin Fowler. xUnit. http://www.martinfowler.com/bliki/Xunit.html, 2010.

[3] John Resig. jQuery. http://jquery.com/, 2011.

[4] Wikipedia. Assertion (computing). http://en.wikipedia.org/wiki/Assertion_(computing), 2011.

[5] Jörg Zaefferer and John Resig. QUnit. http://docs.jquery.com/Qunit, 2011.