JSHint und JSLint - JavaScript Codequality

Bei meinen ersten eigenen JavaScript Gehversuchen mit der Skillmatrix muss ich feststellen wie sehr sich das Entwickeln von JavaScript und Java unterscheidet. Die starre Typisierung und Java ermöglicht ein recht komfortables Refactoring, das somit toolgetützter Teil des Entwicklungsprozesses werden kann. Bei JavaScript scheint mir das schon ein wenig schwerer. In Eclipse kann das Ausmaß dieser Schwierigkeiten an dem Unterschied der Einträge im Refactoring-Contextmenü des JavaScript Editors gemessen werden.

Bei der Suche nach der Lösung dieses Problems stößt man innerhalb kürzester Zeit auf JSLint oder, wenn es etwas stylisher aussehen darf, auf den Doppelgänger JSHint. Hierbei handelt es sich weniger um eine IDE oder ein Refactoring-Tool, sondern um eine Art Stylecheck für JavaScript-Quellcode. JSLint testet die etablierten Codeconventions. Im einfachsten Fall kann man sein Skript einfach via Copy und Paste in das Online-Formular kopieren und auf unterschiedlichste Fehler prüfen lassen. Die Spannbreite reicht von vergessenen Leerzeichen und Kommas über Variablen, die nicht verwendet werden oder sich im Falschen Scope befinden bis hin zu missverständlichen Verzweigungen oder unerwünschten Notationen.

Somit hilft JSLint nicht direkt beim Entwickeln des Codes, meckert aber wenn das Ergebnis nicht ganz optimal ist. Schön wäre es, wenn diese Prüfung regelmäßig schon während der Entwicklung gemacht wird. Geht das? Klar geht das. Bei der Suche nach dem passenden Eclipse-Plugin wird man bei jshint-eclipse fündig. Wie erwartet wird man sofort nach der Installation in der gewohnten Strenge auf mögliche Probleme in seinem Script hingewiesen. Die Art der Prüfungen kann man über das Menü (Windows > Preferences > JSHint > Configuration) entsprechend er JSHint Options leicht anpassen.

An dieser Stelle möchte ich gestehen, dass die Codebeispiele aus der Skillmatrix bereits mehrere JSLint-Durchläufe hinter sich haben. Wie viele Punkte JSLint gefunden hat, bleibt aber mein Geheimnis.

Ich bin mir trotzdem noch nicht ganz sicher ob ich bei der Entwicklung von JavaScript bei Eclipse bleibe oder auf einen Texteditor umsteige. Deshalb probiere ich von Anfang an auch leichtgewichtigere Alternativen wie Sublime aus. Eine Anleitung zur Integration von JSHint in Sublime findet man im Blog von Exratione.

Die Refactoring Tools von Eclipse aus der Java-Welt vermisse ich immer noch. Ich kann mir vorstellen, dass diese Sehnsucht verstärkt wird, wenn sich die Skripte über mehrer Dateien und Verzeichnisse erstrecken. Trotzdem beruhigt JSLint bzw. JSHint, da so Codequalitäts-Standards gesichert werden können, die über einfache Syntaxchecks hinausgehen. Hier hilft wahrscheinlich eine Art Continious Integration Produkt auf JavaScript-Ebene. Aber das ist ein Thema für einen anderen Post.