Specifikation av examensarbete

Emil Wall

5 april 2013

1 Preliminär titel

Engelska: *Practices and Benefits of Javascript Testing*

Svenska: Testning av Javascript - Fördelar och Tillvägagångssätt

Rapporten kommer med allra största sannolikhet att skrivas på engelska, trots att denna specifikation är skriven på svenska. Några anledningar till detta är att det i många avseenden underlättar att kunna använda termer som är etablerade på engelska men som saknar svensk motsvarighet samt att rapporten blir mer tillgänglig och användbar som merit om den även kan läsas av personer som saknar kunskaper om det svenska språket.

2 Bakgrund

Javascript är ett skriptspråk som främst används i webbläsare för att utföra mer komplicerad logik på klientsidan än vad som är möjligt med endast html och css. Det finns ett flertal testramverk för Javascript, men det är långt ifrån alltid som utvecklare utnyttjar dem. Detta innebär att otestad kod sätts i produktion, som kan orsaka stora ekonomiska förluster i samband med driftstopp på grund av oupptäckta buggar, förkortad livstid för produkten och förhöjda kostnader i samband med vidareutveckling och underhåll av koden.

Särskilt allvarligt blir det när Javaskript-koden utgör en del av förutsättningarna för den affärskritiska verksamheten, vilket blir allt vanligare. Det kan till exempel röra sig om stora mängder uteblivna beställningar från en webbshop eller att varumärket skadas då en hemsida upplevs som trasig. Fel uppstår lätt vid ändringar om automatiska tester saknas och kan till exempel ha att göra med inkompatibilitet med olika versioner av webbläsare eller okända beroenden mellan olika delar av Javaskript-koden, saker som skulle gå att upptäcka innan produktionssättning med bra tester. Om enhetstesterna kompletteras med integrationstester och körs automatiskt i ett CI-bygge (på en byggserver som används för kontinuerlig integration) så kan man undvika fel som annars lätt uppstår till följd av oväntade konsekvenser av ändringar på annat håll i applikationen, som påverkar den funktionalitet som javascriptet är tänkt att bidra med.

Att använda sig av tester vid programmering med Javaskript breddar även för testdriven utveckling, vilket för med sig fördelar i form av att designen blir mer genomtänkt och ökar underhållbarheten, både genom att testerna fungerar som dokumentation för koden och genom att koden görs testbar vilket i sig tenderar att innebära efterlevnad av viktiga principer såsom separation av beroenden och att varje funktion gör exakt en sak.

Målet med examensarbetet är att utreda varför testning av Javascript utförs i såpass liten utsträckning idag, undersöka vilka konsekvenser en ökad mängd testning skulle kunna

ge för utvecklingsarbetet och affärsvärdet gentemot kund samt att redogöra för några möjliga tillvägagångssätt för testning av Javascript under olika förutsättningar.

Att skriva tester för Javascript är inget nytt, det första kända testramverket JsUnit skapades 2001 av Edward Hieatt[1][2] och sedan dess har ett flertal andra testramverk tillkommit såsom QUnit[5] och JsUnits uppföljare Jasmine[3], samt verktyg för mockning[1] som till exempel Sinon.JS[9]. Däremot tycks kunskapen om hur man på ett smidigt sätt kommer igång, hur man undviker att testerna blir icke-deterministiska och tidskrävande och vad det egentligen är man ska testa vara sällsynt. Att sätta upp strukturen som krävs för att kunna skriva tester är en tröskel som de flesta Javascript-programmerare inte tar sig över och därmed går de miste om de vinster, såväl kortsiktiga som långsiktiga, som väl utförd testning ger.

I guider för hur man använder testramverk för Javascript är exemplen ofta frikopplade från det typiska användningsområdet av Javascript - webben. Istället tenderar de att innehålla tester av funktioner utan sidoeffekter och beroenden. Under dessa förhållanden blir testningen trivial och de allra flestaJavascript-programmerare skulle säkerligen klara av att sätta upp en testmiljö för sådan enkel kod, men det är alltså inte det problemdomänet som kommer att beröras av det här arbetet utan fokus kommer istället att ligga på hur man testar beteendet hos Javascript som manipulerar DOM-element (Document Object Model, de element som html-kod består av), samt när och varför man bör göra det.

Arbetet kommer att utföras i Valtechs lokaler i Stockholm, ett företag med cirka 180 anställda. Valtech ägnar sig åt konsultverksamhet med fokus på agil utveckling av moderna hemsidor, webbapplikationer och intranät. Namnet Valtech härrör från *value through technology* och grundtanken med företagets verksamhet är att skapa affärsnytta genom att applicera ny teknik och skapa användarvänliga system. Under examensarbetet kommer Valtech att tillhandahålla arbetsplats med teknisk utrustning och handledare. Enligt separat avtal har Valtech gemensam äganderätt till resultatet och immateriella rättigheter som följer av samarbetet. Valtech har inte haft några specifika önskemål utöver att arbetet har genomförts på ett professionellt sätt och håller hög kvalitet. Vilket område eller problemdomän som arbetet ska röra sig inom har lämnats upp till författaren av examensarbetet att besluta. Att arbetet kommer att handla om något som är relevant för Valtechs verksamhet beror alltså på välvilja snarare än påtvingade idéer.

3 Uppgiftsbeskrivning

Här beskrivs mer detaljerat innehållet i examensarbetet: vad som skall göras och vilka moment som ingår. Speciellt skall det beskriva vad som är den intressanta delen i problemet, samt hur detta skall analyseras och lösas. Här bör det alltså framgå att arbetet uppfyller kraven som universitetet har på ett examensarbete på denna nivå.

[1]Mockning och stubbning innebär simulering av beteenden hos verkliga objekt i syfte att isolera systemet under test från yttre beroenden

Att utreda orsaker till dagens begränsade testning av Javascript kan göras utifrån olika infallsvinklar. Det finns mjuka aspekter att ta hänsyn till såsom:

- Skillnader i attityder gentemot testning mellan olika gemenskaper, kretsar och yrkesgrupper
- Synen på Javascript som språk och hur det vanligtvis används
- Kunskap om testning bland de som utvecklar i Javascript
- Ekonomisk lönsamhet

Det finns även mer tekniska aspekter att analysera:

- Testbarhet hos den Javascript-kod som skrivs
- Testverktygens användbarhet
- Begränsningar i vad som går att testa

Att undersöka vad testning av Javascript kan medföra i olika sammanhang kan ske både utifrån ett kortsiktigt perspektiv och ett långsiktigt perspektiv. Utifrån ett kortsiktigt perspektiv lämpar det sig att analysera påverkan på utvecklingsprocessen samt hur testerna kan användas för att skapa korta feedback-loopar, som dokumentation och som en del av kommunikationen med kund. Ur ett långsiktigt perspektiv blir det fokus på det slutgiltiga affärsvärdet och hur testningen i slutändan påverkar kvaliteten på den kod som skrivs.

En redogörelse för hur man praktiskt kan gå till väga för att testa Javascript utgår lämpligen från både de vanligaste och svåraste fallen och innebär samtidigt en utvärdering av de ramverk och verktyg som finns att tillgå. Här finns en skillnad mot de flesta introduktioner som finns för testramverken idag, som tenderar att fokusera på så enkla fall som möjligt. Denna del av arbetet har som mål att ge läsaren konkret vägledning i hur en testmiljö kan sättas upp för att testa Javascript och hur testerna kan utformas, allt beroende på vad det är som ska testas. Utformningen av testerna sker lämpligen utifrån tidigare resonemang kring vad testningen har för konsekvenser, så att mängden onödigt arbete som går åt till att underhålla testerna minimeras medan nyttan och värdet som testerna är tänkta att tillföra maximeras.

4 Tillvägagångssätt

Vilka system, verktyg och metoder som skall användas. Relevant litteratur (det är oftast en del av arbetet att ta fram ytterligare litteratur). Hur resultatet skall utvärderas och dokumenteras. Studenten ska bidra med information om vilka kurser han/hon har tagit som är relevanta och viktiga för examensarbetet.

TODO: Beskriv hur testramverken ska undersökas

Arbetet kommer i första hand att utgå från kvalitativa metoder eftersom insikt i problemdomänen väger tyngre för de mål som satts upp än vad kvantitativt verifierade slutsatser gör. Chansen att hitta de verkliga skälen till varför Javascript testas i så liten

utsträckning ökar med öppna frågeställningar. Dessa metoder kommer att involvera litteraturstudier, intervjuer av gränssnittsprogrammerare vars arbete involverar kodande i Javascript och analys av kod och verktyg. Det kommer även att ingå praktiskt arbete vid utvärdering av de olika testramverken samt tillämpning av testning i ett projekt som författaren tidigare varit delaktig i på företaget, i syfte att kunna beskriva en process för att skriva tester och ge inblick i typiska problem utifrån praktisk erfarenhet.

Preliminärt så kommer följande testramverk att undersökas: Jasmine[3] (+ Jasmine-species[4]), qUnit[5], Mocha[6], JsTestDriver[7], Buster.JS[8] och Sinon.JS[9].

Rapporten kommer att skrivas i LaTeXoch versionshanteras med Git, med ett centralt repo (*repository*, förvaringsplats) på Github för att ge handledare, ämnesgranskare och andra som är intresserade av arbetet tillgång till det under arbetets gång. Adressen till repot är `https://github.com/emilwall/exjobb`, där även denna specifikation och övriga projekt-relaterade filer som är relevanta för utomstående kan tas del av. Varje ändring i rapporten kommer att dokumenteras och motiveras i commit-meddelanden. Potentiellt kommer även kod att versionshanteras publikt, men troligtvis i ett separat repo.

Det finns sedan tidigare akademiskt arbete som behandlar testning av webb-applikationer och även specifikt Javascript. Dessa arbeten kommer att redogöras för i den slutgiltiga rapporten. De arbeten som behandlar automatisk generering av tester[13] kommer att vara av begränsat intresse för examensarbetet eftersom sådana tekniker är mer relevanta för att uppnå olika grader av *code coverage* snarare än testdriven utveckling. Andra är i allra högsta grad relevanta, till exempel ett som behandlar enhetstestning av Javascript som manipulerar en underliggande webbsida[14] och en empirisk studie av hur vanligt det är med buggar i Javascript-kod på produktionssatta webbsidor[15].

En av de viktigaste källorna kommer att vara boken *Test-Driven JavaScript Development*[16] av Christian Johansen, som tar upp testning av Javascript ur ett TDD-perspektiv. Författaren ligger bakom Sinon.JS[9] och är även delaktig i utvecklingen av ett flertal av ramverken som nämnts ovan.

4.1 Relevanta kurser

Följande är ett axplock av de för arbetet mest relevanta kurser som författaren har läst. Utöver dessa har författaren anskaffat sig kunskap om ämnet genom egna studier och i projekt genom anställningar.

1DL250 Programvaruteknik, 5hp Ingående om olika utvecklingsprocesser, design, testning, validering och verifiering.

1DL410 Storskalig programmering, 10hp Projektbaserad kurs med fokus på att skriva kod av hög kvalitet och med tester.

1DT053 Testmetodik, 5hp Fokus på flera olika aspekter av testning: TDD, kriterier för kodtäckning och modellering.

1DT002 Uppsatsmetodik, 5hp Substitut till kandidatexamensarbete, mycket fokus på skrivandeprocessen och källgranskning.

1DL220 Imperativ och objektorienterad programmering, 10hp Gav grundläggande kunskap om objektorienterad programmering i imperativa språk, vilket är relevant om man ska skriva testbar kod i Javascript.

1DL200 Programkonstruktion, 10hp Första programmeringskursen i programmet, mycket fokus på specifikationer av kod.

5 Avgränsningar

Det är också viktigt att skriva ner vad som inte ingår i uppdraget. Detta förebygger att examensarbetet sväller ut okontrollerat. Med fördel kan du skriva ner några punkter som ingår i mån av tid, men räkna med att tiden ofta inte räcker.

Arbetet kommer att fokusera på testning av kod som körs på klientsidan, därmed kommer ramverk såsom vows[10] och cucumis[11] inte att ingå. Detta beror inte på att testning av klientkod på något sätt är viktigare än serverkod, utan är enbart ett sätt att hålla arbetet på en lagom omfattande nivå. Testramverk som inte längre underhålls, såsom JsUnit[2] och JSpec[12], kommer inte heller att behandlas.

Vissa ramverk har även sorterats bort för att de inte bedömts ha lika många användare eller unik funktionalitet för att vara värda att undersöka. Bland dessa finner vi Karma (tidigare känt som Testacular), TestSwarm, YUI Yeti och RhinoUnit. De är säkert användbara, men någonstans behöver en gräns dras för vad som ska tas med och inte. Om det blir tid över så kommer även dessa att undersökas.

6 Tidsplan

Här framgår det när jobbet ska göras och hur mycket tid som allokerats till varje moment (noggrannheten kan variera, men inget moment ska vara större än 4 veckor). Tidplanen skall ta hänsyn till faktorer som gör att man inte jobbar heltid med examensarbetet (semester, kurser som ska avslutas, andra åtaganden). Ofta genomförs moment parallellt, t.ex. rapportskrivningen pågår under hela arbetet. Rita gärna en bild som förtydligar sammanhanget mellan olika moment. Rapporteringsmöten bokas in med ämnesgranskaren.

De första förberedelserna för examensarbetet påbörjades den 2 april 2013, då denna specifikation började skrivas och handledare och ämnesgranskare utseddes. Arbetet är tänkt att pågå på heltid med undantag för viss ledighet under sommaren. De olika delarna av projektet kommer till viss del att ske parallellt, men fokus i början av projektet kommer att ligga på litteraturstudier och att genomföra intervjuer medan det kommer att övergå till mer implementationsarbete längre fram mot slutet.

Skrivandet av rapporten kommer att ske kontinuerligt genom hela projektet, och driva det övriga arbetet.

Referenser

[1] Edward Hieatt and Robert Mee, *Going Faster: Testing The Web Application.* IEEE Software, p. 63 March/April 2002.

[2] Github, *pivotal/jsunit.* `https://github.com/pivotal/jsunit` Read on April 3, 2013.

[3] Running documentation, *Jasmine is a behavior-driven development framework for testing JavaScript code.* `http://pivotal.github.com/jasmine/` Read on April 3, 2013.

[4] Rudy Lattae, *Jasmine-species: Extended BDD grammar and reporting for Jasmine.* `http://rudylattae.github.com/jasmine-species/` Read on April 5, 2013.

[5] The jQuery Foundation, *QUnit: A JavaScript Unit Testing framework.* `http://qunitjs.com/` Read on April 3, 2013.

[6] TJ Holowaychuk, *mocha - simple, flexible, fun javascript test framework for node.js & the browser.* `http://visionmedia.github.com/mocha/` Read on April 3, 2013.

[7] Cory Smith, Robert Dionne, Vojta Jína, Sergey Simonchik, *JsTestDriver.* `https://code.google.com/p/js-test-driver/` Read on April 5, 2013.

[8] August Lilleaas, Christian Johansen, et al., *BusterJS: A powerful suite of automated test tools for JavaScript* `http://docs.busterjs.org/` Read on April 5, 2013.

[9] Christian Johansen, *Sinon.JS: Standalone test spies, stubs and mocks for JavaScript.* `http://sinonjs.org/` Read on April 5, 2013.

[10] Vows: Asynchronous behaviour driven development for Node. `http://vowsjs.org/` Read on April 5, 2013.

[11] Cucumis: BDD Cucumber Style Asynchronous Testing Framework for node.js `https://github.com/noblesamurai/cucumis` Read on April 5, 2013.

[12] JSpec on Github no longer supported `https://github.com/liblime/jspec` Read on April 5, 2013.

[13] Shay Artzi, Julian Dolby, Simon Holm Jensen, Anders Møller, Frank Tip, *A Framework for Automated Testing of Javascript Web Applications.* ICSE '11, Honolulu, Hawaii, USA. May 21–28, 2011.

[14] Phillip Heidegger, Annette Bieniusa, and Peter Thiemann, *DOM Transactions for Testing JavaScript.* Albert-Ludwigs-Universität Freiburg, Germany. Proceeding TAIC PART'10 Proceedings of the 5th international academic and industrial conference on Testing - practice and research techniques. Pages 211-214. 2010.

[15] Frolin S. Ocariza, Jr., Karthik Pattabiraman, Benjamin Zorn *JavaScript Errors in the Wild: An Empirical Study.* 2011 IEEE 22nd International Symposium on Software Reliability Engineering (ISSRE). Pages 100-109. Nov. 29 2011-Dec. 2 2011.

[16] Christian Johansen, *Test-Driven JavaScript Development* ADDISON-WESLEY, ISBN 9780321683915, 2010.