Bachelor Thesis in Software Engineering
May 2013 15hp

Den Moderna Webben

Kan HTML5 konkurrera med traditionell programvara?

Josef Karlsson
Jesper Erlandsson Lähdevirta

Contact Information:
Author(s):
Josef Karlsson
E-mail: sphinxen83@gmail.com

Jesper Erlandsson Lähdevirta
E-mail: jesper_91@live.se

University advisor:
Kari Rönkkö

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstrakt

Vi ser en förändring i hur vi använder applikationer. Från att tidigare varit associerat till en dator och begränsat till ett par operativsystem finns de numera överallt. Fler plattformar innebär att man måste utveckla en applikation till var och en av dessa. Detta gör att utvecklingstiden och kostnaderna ökar och efterfrågan av crossplattformlösningar i form av webbapplikationer har därför ökat.
Vi tittar närmare på hur det går att använda sig av HTML5, och övriga tekniker som en modern webbläsare har integrerat stöd för, för att utveckla webbaserade applikationer. Vi tar reda på om det finns några hinder eller begränsningar i detta och om det finns några fördelar i att använda webben som plattform.
Resultatet indikerar att med de nya tekniker som är på ingång kan man utveckla webbbaserade applikationer med prestanda motsvarande Java eller C#.
Nya tekniker är ständigt på ingång. Genom att uppdatera kunskapen om dessa, tillför vi ny kunskap om hurvida HTML5, tillsammans med dessa nya tekniker kan utveckla webbaserade applikationer som är kraftfulla nog att börja ersätta plattformsspecifik programvara.

Innehåll

Introduktion

Vi ser en förändring i hur människor använder applikationer. Datorer ersätts i allt större utsträckning av surfplattor och mobiltelefoner. I en snabbt förändrande värld där programmerare måste välja mellan ett flertal plattformar och operativsystem kan webben och Internet erbjuda ett bra alternativ. En enhetlig plattform där alla enheter med en modern webbläsare kan köra applikationen [4]. På webben finns applikationerna ständigt tillgänglig och behöver varken installeras eller uppdateras[3] på enheten. Företag behöver inte ha kompetens om olika plattformar, utan kan specialisera sig på utveckling i HTML5. Applikationer behöver därmet inte anpassas efter varje plattform, och behöver heller inte ta hänsyn till plattformsspecifika begränsningar. I *The Death of Binary Software* spår författarna en framtid där binära, kompilerade program är begränsade till operativsystemets kärna och där majoriteten av alla program till slutanvändaren är webbaserad. Med hjälp av HTML5 får utvecklare möjligheter att använda sig av nya verktyg som förr inte existerat, eller endast varit möjlig med hjälp av tredjeparts programvara[4]. I takt med ett växande utbud av webbaserade applikationer har även ansträngningar gjorts för att höja prestandan i JavaScriptsmotorerna i webbläsarna[13]. Det finns redan ett flertal verktyg och projekt som är byggda med HTML5/JavaScript och även operativsystem så som Google Chromium OS där applikationerna i själva verket är webbaserade tjänster[15].

När Tim Berners-Lee 1989 uppfann Internet [5] kunde ingen förutspå den enorma succé som skulle följa. Syftet var att skapa ett nätverk för att kunna dela information och kunskap mellan i främsta hand universitet. Han utvecklade ett markeringsspråk, HTML (*HyperText Markup Language*), tillsammans med ett protokoll, HTTP (*HyperText Transfer Protocol*), för att överföra dess genom nätverket.
HTML baseras på SGML (*Standard Generalized Markup Language*) som i sin tur är en standard för att definiera märkspråk [8]. Iden med ett märkspråk är att kunna ge beskrivande information till den som läser dokumenten, i detta fallen en texteditor, som är skiljt ifrån innehållet i texten.
HTTP i sin tur till ger instruktioner till servern som används för att avgöra vilket innehåll som efterfrågas. Inledningsvis var detta en ytterst primitiv funktion där klienten helt enkelt uppgav ett kommando “GET filnamn” och servern returnerade detta. Detta har senare utvecklats något genom att bland annat hantera statuskod och filtyp.

För att få bättre kontroll på utvecklingen grundades 1994 IETF (*Internet Engineering Task Force)* [5] [9]. Syfte med denna grupp var att utveckla Internet och få det att fungera bättre, bland annat med hjälp av framtagning av standarder. Detta bidrog även till en snabb utveckling av HTML specifikationen. Senare samma år släpptes specifikationen för HTML 2 och redan nästa år presenterades skissen för HTML 3. 1997 släpps HTML 3.2 och arbetet med *Cougar* påbörjas, som senare byter namn till HTML 4.
Då HTML ursprungligen är tänkt att användas för att visa information, är det därför endast utvecklat för att presentera statiskt innehåll och är långt ifrån anpassat för den enorma utveckling som Internet har haft.
Det första stora steget mot en mer dynamisk webb började 1995. Då satte sig en ensam för Netscape vid namn Brendan Eich [18] under några dagar i maj och kodade ihop ett scriptspråk som senare skulle komma att kallas JavaScript. Till skillnad från tidigare kunde man nu skapa effekter och funktioner i ett annars helt statiskt innehåll. Syftet var bland annat att validera formulär och andra användarrelaterade uppgifter [11].
Sedan introduktionen av JavaScript har man sett nya möjligheter att utnyttja Internet och en rad olika tekniker har utvecklats [2] för att täcka dessa behov. Med hjälp av divers tilläggsprogram, som Flash, Quicktime och Shockwave, kunde man skapa interaktiva webbsidor med animeringar och ljud.
Det andra stora genombrottet kom 2005 då uttrycket Ajax (Asynchronous JavaScript and XML) myntades [18]. Detta är ett sätt att utnyttja JavaScript för att göra asynkrona anrop till servern och uppdatera delar av innehåller på en sida istället för hela sidan.

På senare tid har en ny marknad introducerats, den mobila. Detta innebär fler plattformar och fler operativsystem att ta hänsyn till. Det har gjort att många utvecklare känt sig överväldigad att behöva utveckla en produkt för varje

plattform och en del företag väljer istället att utveckla så kallade RIA som en crossplatform lösning [6]. RIA (*Rich Internet Application*) innefattar program som på många sätt efterlikar de som används i skrivbordmiljö, men som kan köras i en webbläsare på ett eller annat sätt, antingen med hjälp av till exempel plugin program eller med JavaScript. De tillgängliga tekniker för detta har fram till nyligen dock varit beroende av plugin program och har därmed inte varit en helt universal lösning för crossplatform, främst på grund av de mobila plattformerna.

I takt med att HTML5 närmar sig en standard och behovet av tilläggsprogram minskar har även intresset bland utvecklare att använda webben som en universal plattform ökat [14]. Även om HTML5 i sig inte kan räknas till en RIA kan man tillsammans med bland annat JavaScript och Ajax skapa en motsvarande miljö. Vi har redan börjat se smakprov på mer avancerade program och spel som inte längre är i behov av att installeras på den lokala datorn, utan kan köras direkt i webbläsaren [10] utan några tilläggsprogram. Fördelen med dessa applikationer är att de inte längre är bundna till en specifik hårdvara, utan användaren kan enkelt byta såväl plattform som plats och fortfarande ha tillgång till sitt material.

I den här rapporten tittar vi på möjligheterna att använda HTML5 med relaterande tekniker för att producera webbaserad programvara med motsvarande funktionalitet som nativ programvara, så kallad RIA (Rich Internet Application). Vi undersöker om dessa tekniker är tillräckligt utvecklade för att kunna konkurrera med applikationer specifikt utvecklade för en plattform. Med hjälp av nya tekniker så som WebGL [31], som tillåter hårdvaruacceleration direkt i webbläsaren, och Emscripten, som möjliggör konvertering av C/C++ kod till JavaScript, tillsammans med redan existerande tekniker redogör vi om webben har blivit en mogen plattform för mjukvarutuveckling. Forskningen är främst koncentrerad till de tekniker som moderna webbläsare har inbyggt stöd för (HTML, CSS, JavaScript) och grundar sig på litterära studier samt intervjuer med utvecklare och deras syn på webbaserad utveckling, så väl som plattformsspecifik.
Två intervjuer utfördes på två lokala företag, varav det ena utvecklar webbaserade applikationer, medans det andra företaget fokuserar utvecklingen på mer traditionell mjukvara. Intervjuernas huvudsyfte är att ta del av utvecklarnas erfarenhet om användingen av HTML5.

Resultatet visar att webbaserade RIA kan konkurrera med plattformsspecifika applikationer. Detta styrks av vår studie då det går att utföra bra testning av kod, vid utveckling av webbaserade applikationer. Vi påvisar även några av de problem som finns med webbaserade RIA , så som dålig prestanda.

Forskningsfrågorna som den här studien fokuser på är följade:

- *Är det möjligt att utveckla applikationer baserade på HTML5 som kan jämföra sig med motsvarande programvara utvecklad i nativ miljö?*
  Avhandlingen reder ut om det finns några tekniska begränsningar som är till hinder för utveckingen av programvara baserad på HTML5.
- *Kan HTML5 applikationer uppnå liknande prestanda som plattformsspecifika applikationer?*
  Avhandlingen undersöker JavaScripts prestanda i literaturstudien och i den empiriska studien. Därefter klargöra prestanda skillnaderna och analysera deras betydande för HTML5 baserad RIA utveckling.
- *Kan HTML5 baserade applikationer erbjuda liknande möjligheter för kodtestning som applikationer vars utveckling bedrivs i nativ kod?*
  Avhandlingen undersöker vilka testningsmöjligheter det finns med JavaScript och jämföra det med den testningen som utförs med traditionella språk.

Litteraturdesign

Vår huvudsakliga källa för artiklar är http://www.engineeringvillage.com som använder sig av databaserna Compendex, Inspec, GeoRef, GEOBASE, EncompassLIT, EncompassPAT, US Patents och EP Patents. Vi valde denna källa då den riktar sig mot teknisk litteratur. För att ytterligare få bort irrelevanta resultat valde vi att endast söka efter litteratur som var utgiven efter 2010. Motivationen till detta var att webben är i ständig rörelse och många av de tekniker som berörs i detta arbete var inte i kommersiellt bruk innan 2010. Litteraturundersökningen utgör teori kapitlet i denna avhandling.

För att svara på frågorna måste vi först reda ta reda på vilka tekniker som finns tillgängliga. Med utgångspunkt från HTML5 börjar vi sökningen. För att begränsa sökresultaten specificerar vi söksträngen i olika kombinationer för att få med all relevanta avhandlingar. I kombination med *HTML5* använde vi oss av nyckelord andra nyckelord som *web development, RIA, JavaScript, web-based system, Web programming*. Söksträngen som tillslut gav bäst resultat var:

((HTML5) OR (“HTML 5”)) AND (“web application”) AND (technolog*)

Denna sökningen gav oss mycket relevant information. Med det visar också att det är ett relativt nytt område och inte så mycket forskning har hunnit bedrivas. Efter ha granskat igenom papprena kom vi fram till ett antal fler ord att söka på, *ECMAscript, JavaScript 2.0, WebGL, Lively Kernel* som gav oss ytterligare en del intressant information om exprementiella projekt och tekniker under utveckling.

För att ytterligare erhålla information deltog vi dels i Webbdagarna, en konferens i Stockholm, samt följde Google I/O via Internet. Syftet med dessa var främst att försöka snappa upp nya trender och tekniker som vi sedan kunde undersöka noggrannare En av dessa tekniker som dök upp var ams.js. En sökning på detta gav dock inget resultat bland de erkända sökmotorerna. Då detta är en så pass intressant och ny teknik valde vi att ändå ta upp det och istället vända oss till mindre granskade källor, och utvärdera dessa mot varandra, för om inte annat påtala styrkor och brister i detta. Det ställde krav på att det fanns flera källor av oberoende aktörer för att resultatet skulle bli objektivt som möjligt. Detta spår ledde även in på andra intressante områden och vi hittade nya sökord som Compiled JavaScript, Emscripten, mandreel, llvm to JavaScript.

Intervjudesign

Intervjuer valdes som studiemetod eftersom de tillfrågade med större sannolikhet medverkar i denna typ av studie. Det kan i många fall vara omotiverade med enkätundersökningar, vilket kan göra att de tillfrågade låter bli att delta, eller undviker att svara utförligt. Intervjuer ger däremot möjlighet för djupare diskussion.

För att få ett objektivt resultat av studien valdes två programmerare inom det studerade området. Programmerarna är verksamma inom två skilda företag. Det ena företaget jobbade främst med webbutveckling och hade anammat HTML5 som de använde i större utsträckning i sin produktutveckling. Det andra företaget jobbade med mjukvara och programvarututveckling i stort och hade inte bara webben som främsta plattform Deras utbredning av HTML5 var därför inte lika stor som hos det första.

Intervjun hade bland annat i avsikt att tydligöra de skillnader som fanns företagen emellan, angående deras erfarenhet av att använda HTML5. Därav ställdes samma grundfrågor under båda intervjuerna Dessa grundfrågor ledde till vidare diskussion. Grundfrågorna var förbedda och nerskrivna på papprena. Grundfrågorna som ställdes till de intervjuade grundade sig i två kategoriter. Första kategorin relaterade till företagets erfarenheter inom HTML5

och dess behov av de möjligheter som HTML5 ger. Andra kategorin gällde programvaruutveckling generellt, och de tillfrågades erfarenheter inom detta. Frågorna var utformade för att få en bild av traditionell utveckling i förhållande till webbutveckling och var i stor utsträckning relaterade till den erfarenhet de tillfrågade fått av att arbeta med programvaruutveckling, samt deras åsikter om konkurenskraften HTML5 har gentemot andra metoder.

Intervjuerna utfördes på de tillfrågades arbetsplats. Detta gav oss möjlighet att även observera hur de arbetade och gav oss inspiration till flera frågor, så som frågan *"hur utbredd är kompetensen om HTML5 på företaget"*. Intervjuernas upplägg var inte strikt förutbestämt. Grundfrågorna styrde riktning för projektet. Intervjuerna spelades in för att i efterhand ha möjlighet att gå tillbaka och granska dessa. Intervjuernas längd var ca 30 minuter långa och följdes av en liten diskussion på ca 15 minuter. Detta var tillräcklig för att utföra en välplanerad intervju och hinna få svar på samtliga frågor. Diskussionen som följde resulterade även i att fler frågor och information dök upp som man kanske kunnat missa annars.

Dessvärre har även den här typen av forskning vissa svagheter. En nackdel är att denna typ av forskning är väldigt tidskrävande och lämpas inte för en större skala människor, vilket kan ge ett väldigt spridda resultat beroende på vilka som tillfrågas. En annan möjlig nackdel är att de frågor som uppkommer under intervjun kan vara partiska och därför försämra tillförlitligheten hos intervjuerna. Det utfördes endast två intervjuer, vilket även kan leda till missvisande resultat.

Teori

Utbredningen och utvecklingen av Internet har gjort att man sett nya möjligheter och användningsområden som från början inte var tänkt. Mikkonen kategoriserar i sin forskning in utvecklingen av Internet till tre faser [2]. Den dokument orienterade fasen beskriver Internets ursprungliga syfte med delning och distribuering av statisk information. Användaren kunde navigera runt på olika webbsidor, som var just sidor, med data. Varje ny efterfrågan resulterade i att en ny sida hämtade och presenterades.
Den andra fasen beskriver hur Internet utvecklas till en bas för applikationer. Tack vare JavaScripts introduktion, som möjliggjorde manipulering av de tidigare statiska sidorna, började intresset för alternativa användningsområden i webbläsaren komma igång.
Begränsningarna i webbläsarna gjorde sig nu påmind. Olika webbläsare tolkade innehållet något annorlunda och de grafiska egenskaperna var mycket begränsade. Detta har resulterat i utveckling av en rad olika program för att kringå dessa problem. Olika pluginprogram som Flash, Silverlight och Realtime har tidigare varit mer eller mindre nödvändiga för att uppnå önskad funktionalitet.
I och med introduktionen av Ajax som möjliggjorde asynkron kommunikation till servern ökade intresset för webben som en applikationsplatform. Stora ansträngningar lades på att optimera JavaScript för att få det att bli snabbare. Nya JavaScriptsmotorer, så som Apples SquirrelFish, Googles V8 och Microsofts Chakra, såg sitt ljus [3].
I den tredje fasen förutspår Mikkonen [2] att webben i framtiden kommer utgöra huvudplattform för utveckling av mjukvara. Fördelarna med en centraliserad plattform är många. Ett argument som Mikkonen tar upp som ett av de avgörande är att Internet erbjuder direkt distribuering världen över. Inga installationer behövs och uppdateringar görs tillgängliga direkt. I princip vem som helst kan utveckla och sprida en applikation från sin egna sida. Det underlättar samarbete mellan användare och även vid installation och underhåll [3].

Den snabba utvecklingen från statiska dokument till interaktiva applikationer har gjort att tekniken inte riktigt hunnit med. I forskningen Are web applications more defect-prone than desktop applications [16] tar författarna upp några av dessa problem. Deras forskning visar på att webbapplikationer är mer defektbenägna. Forskning går ut på att jämföra likvärdiga applikationer som uppfyller samma syfte och finns motsvarande för webb och skrivbordsmiljö. Även om de jämförda applikationeras syfte och funktionalitet är väldigt lik så har de dessvärre skapats av olika företag, företag som kan ha olika kvalitetsmål på sina produkter. Detta är något som kan påverka studiens resultat avsevärt. De drar dock slutsatsen att det beror på att testning av JavaScript är väldigt svårt och att utvecklingstiden är mycket kortare hos webbapplikationer i jämförelse med PC applikationer. De hävdar även att den ständiga ström med nya webbtekniker som etableras i rask takt försvårar inlärningen till fullo av nya tekniker. Även den Empiriska studien visar att testning är något som inte har prioriterats vid javascripts utveckling. De både intervjuade nänmer att de inte har någon kunskap om hur testning utförs med javascript. Däremot utför båda företagen testning av annan kod. De nämner att de inte har någon kompetens på företaget angående testning av javascript. Peter Svensson nämner att det har gått trögt att gå över till HTML5, även om det finns kompetens angående HTML5 på företaget. Däremot visade den andra intervjun som utfördes med Andreas Björklund att utvecklingen har gått över till att använda HTML5, så mycket det går. Vilket resulterar i att det finns skillnader i hurvida stor utsträckning HTML5 används vid utveckling, beroende på vilken typ av företag utvecklingen bedrivs på.
Liknande slutsatser kommer Anttonen och Salminen fram till i deras forskning [19]. De påvisar att det finns vissa prestanda problem med JavaScript virtual machine, men poängterar att det har förbättrats avsevärt i takt med att webbapplikationer blir allt vanligare. De skriver att JavaScript är väldigt dynamiskt och blir därav väldigt svårt att testa, och därav mer defektbenäget.

Kraven på prestandan i JavaScript har tidigare varit relativt låga och eventuella problemen har inte märkts av särskilt påtagligt. Men i takt med att applikationerna blivit mer avancerade har dessa problem visat sig allt tydligar. De JavaScriptsmotorer som tagits fram av företag som Google, Apple och Microsoft har visserligen ökat prestandan avsevärt, men är fortfarande påtaglig i takt med att applikationerna och komplexiteten ökar.

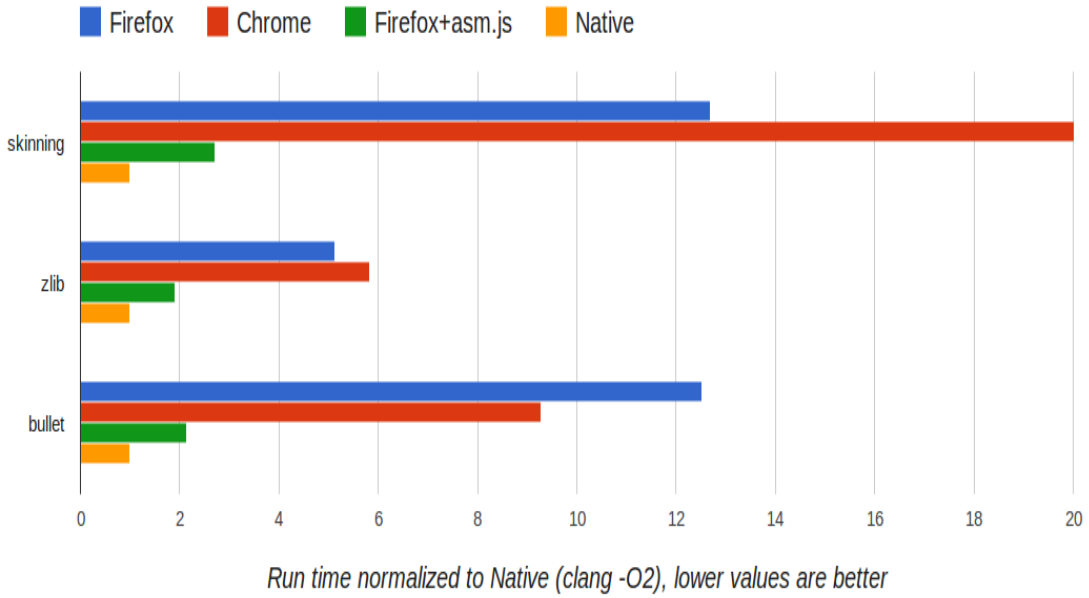
Liknande resultat har Jazayeri och Ahmadi kommit fram till i sin forskning [15]. De använder en casestudy med hjälp av AgentWeb[15]. som är en grafisk editor i HTML5. De skapar webbapplikationer och PC applikationer och jämför prestandan. De drar slutsatsen att PC applikationer har överlägsen prestanda men att optimeringar av JavaScriptsmotorerna gör att skillnaderna minskar allt mer.
Dock så påvisar Martinsen och Grahn i sin forskning [11] ett problem med dagens sätt att mäta prestandan i JavaScript. De tester som används idag är samma tester kom körs för programmeringsspråk avsett specifikt för PC datorer och att dessa mätningar inte ger ett sanningsenligt resultat. De försöker därför presentera en metodik för att karakterisera JavaScript och därmed skapa bättre prestandatester för JavaScript. Även den emperiska studien visade att javascript används mer än innan. De både intervjuade nämner att de har märkt att javascriptsmotorerna har blivit snabbare. Men Andreas Björkblom som var den tillfrågade från Milou nämde att javascriptsmotorerna fortfarande inte verkar vara så snabba i mobila enheter.

Fram till nyligen har en av de största brister med webbläsaren varit dess oförmåga att hantera 3D grafik. Tidigare har man tvingats förlita sig på tredjepartsprogram för detta. Problemen med dessa är att de inte fungerar för alla plattformer och webbläsare. Därför är WebGL en teknik som kanske är den mest intressanta. WebGL är en del av OpenGL som drar nytta av nya canvas elementet i HTML5 för att rendera 3D grafik med hårdvaruacceleration direkt i webbläsaren. Taivalsaari och Mikkonen tror att detta kommer ha ett betydande inflytande på utvecklingen då detta eliminerar vad de menar är "det sista säkra fästet för konventionella binära applikationer" [2]. De får även medhåll från andra som menar att WebGL är det största som har hänt webben på länge [19]. En annan nyhet med HTML5 är en cache funktion som gör att man kan lagra data lokalt på enheten och därigenom ha tillgång till detta även offline. För att kunna bygga bra RIA applikationer tas denna funktion upp som en nödvändighet [12][4].

Nyligen presenterade grundaren av jQuery, John Resig, ett mycket intressant inlägg på sin blog där han berättar om asm.js [20]. Asm.js är ett forskningsprojekt av Mozilla och har som mål att definiera den delmängd JavaScript som kompilatorer som Emscripten och Mandreel genererar. Emscripten är en kompilator som tar LLVM (Low Level Virtual Machine) kod och konverterar detta till JavaScript [23][24]. Emscripten har tidigare endast genererat vanlig JavaScripts kod [23]. Men nu kan även Emscripten kompilera LLVM kod till asm.js kod. Kompilering av JavaScript är i sig inget nyhet i sig utan används i flera sammanhang. Det kanske vanligaste exemplet är Googles *Closure Compiler* som helt enkelt minimerar och optimerar JavaScript koden för att få den effektivare och mindre resurskrävande [21]. Tack vare kompilatorer som kan konvertera från LLVM till JavaScript är det möjligt att utveckla program i andra miljöer så som C/C++ och sedan konvertera detta till LLVM kod, för att senare kompilera till JavaScript.
Problematiken med JavaScript uppstår då enklare funktioner börjar övergå till avancerade beräkningar. JavaScripts dynamiska struktur gör att datatyper måste listas ut allt eftersom. Ju fler funktioner desto fler variabler, vilket i sin tur påverkar prestandan. Asm.js har som mål att eliminera dessa problem. Ams.js sätter tydliga begränsningar i vad den får göra och hur det kan fungera. Genom en flagga, "use asm" får webbläsaren information om att den ska tolka koden som asm.js. Webbläsare med implementerat stöd försöker då kompilera koden AOT (Ahead Of Time) utifrån dessa strikta regler genom att i förhand förvänta sig rätt datatyp och även förhindra att dessa senare ändras. På så vis undviker man att identifiera varje datatyp vid körning, vilket är enormt krävande process. Om kompileringen skulle misslyckas, eller om webbläsaren inte har implementerat stöd för asm.js tolkas helt enkelt koden som vanlig JavaScript med kontroll av datatyp som följd.
Tester som utförts visar att i komplexa program som till exempel 3D-spel, kan prestandan i en webbläsare med rätt stöd komma ner i siffror under faktor 2, i jämförelse mot motsvarande program kompilerad i C++. Dessa siffror är jämförbara med kod skriven i Java eller C# [20]. Då ska man ha i åtanke att asm.js fortfarande är i ett tidigt utvecklingsstadie. Även om konverteringen från LLVM till asm.js sker i Emscripten kan prestandavinster uppnås genom förbättrat stöd för asm.js i webbläsare. Med hjälp av asm.js kan man utveckla i en strikt miljö så som C/C++ och sedan konvertera till JavaScript med gott resultat.

Nackdelarna med ams.js såhär lång är att webbläsaren måste ha stöd för det för att fungera tillfredsställande. Även om ams.js i grunden är JavaScript och kommer fungera även i webbläsare utan implementerat stöd, visar tester att det i vissa fall rent av kan bli sämre än att använda vanlig JavaScript [25].
Ams.js kräver även en minnes array storlek med exponenten 2, vilket gör att det kan bli väldigt minneskrävande vid avancerade program. Detta beror på att JavaScripts egna minneshantering använder för mycket resurser, vilket betyder att minnet i en asm.js applikation inte allokeras dynamiskt. En annan nackdel med asm.js ligger i hur JavaScript hanterar 64-bitars heltal. JavaScript kan endast hantera 32-bitars heltal, vid större tal blir dessa istället flyttal[23]. Detta medför risker som att avrundningar som görs i en PC miljö inte blir den samma när avrundningar görs i den konvertade asm.js koden.
En annan begränsning är att JavaScript i sig inte har stöd för trådning. Även om det i teorin finns metoder för att simulera multitrådning med hjälp av Web Workers [26], som möjliggör att script kan köras i bakgrunden, är dessa i dagsläget väldigt begränsade och skulle bli väldigt långsamma. Emscripten kan därför bara hantera enkeltrådiga program och är således även det testerna baseras på. Dock förklarar studien att det är möjligt att använda ett tillägg till JavaScript vars kodnamn är River Trail [27] för att skapa trådning. River Trail är ett Open Source projekt som drivs av Mozilla och har nyligen blivit integrerad i JavaScriptsmotorn för Firefox Nightly[28]. Denna tillägg är således ännu inte implementerad i webbläsare för för kommersiellt bruk.

Realalistic/Large Benchmarks



Då testet utfördes hade endast Firefox stöd för asm.js
(Källa: http://kripken.github.io/mloc_emscripten_talk/gindex.html)

Analys

Diskussion

Forskningen visar att tekniker som krävs för utveckling av RIA har utvecklats kraftigt. Nya optimeringar och användningssätt av JavaScript gör att behoven av tredjepartens programvara inte längre är nödvändiga. Vissa begränsningar kvarstår fortfarande, främst i mobila enheter., så som sämre presterande JavaScripts motorer.

Studien visar att testning i JavaScript är näst intill obefintlig. Den visar även att testningen av JavaScript i sig har varit väldigt svårt att utföra, dels pågrund av dess dynamiska struktur och uppförande, och en av de grundläggande anledningar att testverktyg inte funnits att tillgå. Asm.js ger möjlighet att skriva kod i C++ och utnyttja de testverktyg som redan finns utvecklade i C/C++ och därefter konvertera koden till asm.js JavaScripts kod. Detta kan i många fall lösa problem med testing relaterad till JavaScript, då studien visade att de utvecklarna vi var i kontakt med ofta besatt god kunskap inom testning av statiska språk, så som Java och .NET, men inte inom JavaScript och därför helt enkelt lät bli att testa denna. Det betyder att testkvaliten möjligen kan öka om utvecklingen sker i en statisk miljö, som sedan kompileras till JavaScript. Med andra ord, företag som har investerat tid och pengar i testverktyg kan dra stor nytta av Emscriptens förmåga att konvertera LLVM kod till asm.js kod. När asm.js kombineras med det hårdvaruaccelererande WebGL kan utvecklare, tack vare asm.js prestanda fördelar, skapa avancerade 3D applikationer.

En viktig förutsättning för att asm.js ska vara användbart, är att den asm.js generade koden uppför sig på samma sätt i den webbaserade miljön, som den gör i den ursprungliga miljön. Som tidigare nämnt är det endast kompilatorer som läser asm.js kod och om fel uppstår i asm.js kod är det omöjligt att felsöka problemet. Fel kan tillexempel vara avrundningar som inte hanteras på samma sätt, som får den asm.js konverterade applikationen att generera andra resultat, än vad den ursprungliga applikationen genererar. Därför är det oerhört viktigt att Emscripten inte genererar falerande asm.js kod.

Vi fann dock att ett av problemen med asm.js är att JavaScript är seriellt och därav inte implementerar trådning. De tester vi tog del av i vår studie fann att flertrådade applikationer som konverterats till asm.js tog stor skada av JavaScripts oförmåga att hantera trådning. Vi hittade att trådning skulle gå att emulera på ett tvivelaktigt sätt med hjälp av Web Workers, som är en del av HTML5 API. En annan potentiell lösning som är under utveckling är River Trail. River Trail är ett open source projekt av Mozilla som ska möjliggöra trådning i JavaScript. En kombination mellan asm.js och River Trail skulle i så fall innebära att de största bristerna med JavaScript skulle kringås.

Fördelen med att använda asm.js är att det endast är JavaScript. Då det nu är möjligt att konvertera LLVM kod till asm.js JavaScript har utvecklare möjlighet att skapa avancerade crossplatform applikationer, något som innan varit svårt. Asm.js ger utvecklarna möjlighet att skapa applikationer i andra språk för att sedan konvertera dem till optimerad JavaScript, som kan läsas av webbläsaren. Nackdelen är det inte längre är möjligt för en människa att läsa och förstå asm.js kod, men även inte består av någon semantik. I brist på ordentliga testverkyg för JavaScript blir det därmed praktiskt omöjligt att debugga och felsöka koden vid eventuella fel och buggar. Detta gör det oerhört viktigt att asm.js koden exekveras på samma vis som i ursprunglig miljö. Det finns även andra tekniker med liknande syfte som har både fördelar och nackdelar gentemot asm.js. Till exempel Googles open source projekt Native Client, som även lovar höga prestandavinster. Även denna teknik tar hjälp av HTML5s WebGl API. Däremot är Native Client, till skilnad från asm.js en plugin som för närvarande måste installeras och endast är stöds av Google.

Att använda kompilationrer som exempelvis Emscripten kan förkorta utvecklingstiden för projekt där syftet är att konvertera redan befintlig kod till JavaScript. Men det är viktigt att komma ihåg att en webbapplikation skiljer sig från nativ programvara då dess syfte ofta inte är att kopiera en traditionella applikation mer än funktionsmässigt,

utan istället dra nytta av de möjligheter en webbapplikation ger. En webbapplikation som använder sig av HTML5 och dess elementen kan nyttja den möjlighet som CSS ger i form av responsiv design för att effektivt skapa crossplattform applikationer.

Traditionellt sett har mycket av beräkningarna gjorts på servern och sedan har datan visualiserats med hjälp av HTML och CSS. JavaScripts uppgift har främst bestått i att sköta assynkron kommunikation mellan klient och server, samt att förstärka användarupplevelsen i form av funktionallitet och effekter. Asm.js har än så länge fokuserat på att låta klienten utföra beräkningarna. Då HTML5 innefattar websockets är det i teorin möjligt att låta serverar utföra en del av beräkningarna som klienten annars hade behövt göra själv. Dock har vi inte hittat information som stödjer detta för just asm.js. En alternativ teknik som är under utvecking av Leaningtech skulle kunna vara en lösning. Duetto [29] är tänkt att kunna konvertera C/C++ kod till en klient - server lösning. Den utnyttjar JavaScript som det ser ut idag, snarare än att begränsa funktionaliteten som asm.js gör. Det innebär att lösningen skulle fungera väl även i webbläsare utan stöd för asm.js. Hur väl de lyckas med detta återstår dock att se, då de fortfarande är väldigt tidigt i utvecklingen.

Reslutatet av forskningen talar för att det är möjligt att skapa RIA webbapplikationer med HTML5, men även att det är mer fördelaktigt att utnyttja HTML5 i stället för att använda sig av tredjeparts tekniker som Adobe Flash. Då Flash inte har lika utbrett stöd bland webbläsare. Dock har fortfarande tredjeparts tekniker som Microsoft Silverlight och Adobe Flash möjlighet att kryptera information som skickas från servern med hjälp av DRM. HTML5 har för närvarande inget stöd för det. Men det finns planer på att införa Web Cryptography API [30] som en del av HTML5.

De mobila JavaScriptsmotorerna inte är lika snabba som de en traditionell webbläsare har. Då en mobil har begränsad batteritid och processorkraft är det ytterst viktigt att applikationer är så effektiva som möjligt. Vi tror att kombinationen av asm.js och HTML5 gör möjligt att att skapa ännu bättre interaktiva applikationer för mobiler. Det är dock viktigt att poängtera att det även utan asm.js går att skapa RIA med hjälp av HTML5.

Studien talar för att man med hjälp av HTML5 och de nya möjligheter som följer, tillsammans med nya tekniker att optimera JavaScript, går att utveckla avancerade applikationer som går snabbt att köra. Från att ha varit en lekplats för interaktiva spel och applikationer kan webben nu på allvar börja konkurrera om platsen som fullblodsplattform på samma nivå som Java och C#. Vi kommer bara i år se lansering av ett flertal nya tekniker för att möjliggöra detta, där bland Googles Dart och Mozillas asm.js. Även mindre aktörer som Leaningtechs Duetto verkar intressant och lovande. Hur dessa står sig i förhållande till varandra återstår att se.

Dart är ett helt nytt programmeringsspråk speciellt framtaget för utveckling av avancerade webbapplikationer. Men en struktur som liknar redan etablerade språk, tillsammans med en kompilator från C/C++ till Dart, hoppas Google på att få genomslagskraft på marknaden för större och avancerade webbapplikationer och att fler webbläseutvecklare ska implementera stöd för det.

Vidare forskning

Vidare forskning om hurvida det är möjligt att optimera asm.js kod och optimera JavaScripts motorerna behövs. Det behövs även vidare forskning inom användingen av asm.js kod tillsammans med flera HTML5 APIer, så som websockets. Forskning för hur prestandan i JavaScript skulle förändras med parallel JavaScript, det vill säga trådning, behövs. Det krävs även vidare forskning i hurvida Navite Client, asm.js och Duetto skiljer sig i prestanda och vilken teknik som är att föredra i olika sammanhang. Det krävs även vidare forskning om hur asm.js kan dra nytta av trådning.

Slutsats

Genom att kombinera det HTML5 relaterade WebGL APIet med asm.js går det att skapa webbaserade applikationer som kan konkurera med plattformsspecifika applikationer. Prestandan hos de webbbaserade applikationerna är inte lika bra som hos de plattformsspecifika. Men möjligheten att nå applikationen på flera olika plattformar utan installation kan i många fall väga upp den något sämre prestandan. Testning är ett viktigt kriteria för att säkerhetställa kvalié. Testning i javascript har tidigare varit svårt. Men med hjälp av Emscriptens förmåga att konvertera LLVM kod till asm.js går det ta det ta del av de testningsmöjligheter som till exempel språket C++ ger och därmed undvika utförandet av testning i javascript.

Även om asm.js ger bättre prestanda än vanlig JavaScript så erbjuder erbjuder ofta plattformsspecifika programmeringsspråk parallelism, vilket javascript inte erbjuder. Men vår studie har visat att det Mozilla drivna projektet River Trail har ambitioner att tillföra parallelism till javascript, vilket stärker vår slutsatts att Webbaserade applikationer kan konkurera med platformspecifika applikationer.

Referenser

[1] A.Balasubramanian,B.Levine, A.Venkataramani, “Enhancing Interactive Web Applications in Hybrid Networks” , i *MobiCom'08 - Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, p 70-80, 2008

[2] A.Taivalsaari, T.Mikkonen, “The Web as an Application platform: The Saga Continues”, i *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications,* p 170-4, 2011

[3] A.Taivalsaari,T.Mikkonen, M.Anttonen, A.Salminen, “The Death of Binary Software: End User Software Moves to the Web”, i *2011 Ninth International Conference on Creating, Connecting and Collaborating through Computing (C5)*, p 17-23, 2011

[4] C.Li-li, L.Zheng-long, “Design of Rich Client Web Architecture Based on HTML5”, i *2012 Fourth International Conference on Computational and Information Sciences (ICCIS)*, p 1009-12, 2012

[5] D. Raggett, J. Lam, I. Alexander, M. Kmiec, *Raggett on HTML 4*, 1998, ISBN 0-201-17805-2 http://www.w3.org/People/Raggett/book4/ch01.html [June 11, 2013]

[6] D.Pavlić, M.Pavlić, V.Jovanović, “Future of Internet Technologies”, i *2012 35th International Convention on Information and Communication Technology, Electronics and Microelectronics*, p 1366-71, 2012

[7] G.Ghiani, L.Isoni, F.Paterno(2011), “Security in Migratory Interactive Web Applications”, i *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia, MUM 2012*, 2012

[8] W3C, “HTML 4.01 Specification”, 24 Dec. 1999, http://www.w3.org/TR/html4/cover.html [Accessed 11 June 2013]

[9] The Internet Engineering Task Force, *IETF*, http://www.ietf.org/ [Accessed 11 June 2013]

[10] I. Jacobs, J. Jaffe, and P. L. Hégaret, “How the Open Web plattform Is Transforming Industry”, i *IEEE Internet Computing*, v 16 n 6 p 82 -86, 2012
Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6355548 [Accessed 11 Feburary 2013].

[11] J. K. Martinsen, H. Grahn, “A Methodology for Evaluating JavaScript Execution Behavior in Interactive Web Applications”, *2011 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, p 241-8, 2011

[12] J.Harjono, G.Ng, D.Kong, J.Lo (2010), “Building Smarter Web Applications with HTML5”, i *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research, CASCON'10*, p 402-403, 2010

[13] J.Martinsen, H.Grahn (2011), “A Methodology for Evaluating JavaScript Execution Behavior in Interactive Web Applications”, i *2011 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, p 241-8, 2011

[14] M. B. Hoy, “HTML5: A New Standard for the Web”, *Medical Reference Services Quarterly*, v 30 n 1 p 50-55, 2011
Available at: http://www.tandfonline.com/doi/abs/10.1080/02763869.2011.540212 [Accessed 10 Feburary 2013].

[15] M.Jazayeri, N.Ahmadi, “End-User Programming of Web-Native Interactive Applications”, i *ACM International Conference Proceeding Series*, v 578, p 11-16, 2011

[16] M.Torchiano, F.Ricca, A.Marchetto, “Are web applications more defect-prone than desktop applications?”, i *International Journal on Software Tools for Technology Transfer*, v 13, n 2, p 151-66, April 2011

[17] R.Glotzbach, L. Kocur, “Work in Progress - Evaluation and Feedback for Web Programming Curriculum”, i *41st ASEE/IEEE Frontiers in Education Conference (FIE 2011)*, p F4E (2 pp.), 2011

[18] W3C, *Short History of JavaScript*, http://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript, 27 June 2012 [Accessed 11 June 2013]

[19]M.Anttonen, A.Salminen, T.Mikkonen (2011), Transforming the Web into a Real Application plattform: New Technologies, Emerging Trends and Missing Pieces, i *Proceedings of the ACM Symposium on Applied Computing*, p 800-807, 2011, *26th Annual ACM Symposium on Applied Computing, SAC 2011*

[20] J. Riseg, "Asm.js: The JavaScript Compile Target", 3 April 2013, http://ejohn.org/blog/asmjs-JavaScript-compile-target/ [Accessed 11 June 2013]

[21] Google, "Closure Tool", 2 July 2012, https://developers.google.com/closure/compiler/ [Accessed 11 June 2013]

[22] ams.js, "asm.js Working Draft", 17 March 2013, http://asmjs.org/spec/latest/ [Accessed 11 June 2013]

[23] A. Zakai, "Emscripten: An LLVM-to-JavaScript Compiler", i *SPLASH'11 Compilation - Proceedings of OOPSLA'11, Onward! 2011, GPCE'11, DLS'11, and SPLASH'11 Companion*, p 301-312, 2011

[24] Emscripten, "Emscripten", 10 June 2013, https://github.com/kripken/emscripten/wiki [Accessed 11 June 2013]

[25] P. Bright, "Surprise! Mozilla *can* produce near-native performance on the Web", 23 May 2013, http://arstechnica.com/information-technology/2013/05/native-level-performance- on-the-web-a-brief-examination-of-asm-js/ [Accessed 11 June 2013]

[26] Web workers, "Web workers - HTML standard", 10 June 2013, http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html [Accessed 11 June 2013]

[27] J. Sreeram, S.Herhut, R.Hudson, T.Shpeisman, "Teaching Parallelism with River Trail", i *SPLASH 2012: DCP 2012 - Proceedings of the 2012 ACM Workshop on Developing Competency in Parallelism: Techniques for Education and Training*, p 1-8, 2012

[28] L. Wagner, "asm.js in Firefox Nightly", 21 March 2013, http://blog.mozilla.org/luke/2013/03/21/asm-js-in-firefox-nightly/ [Accessed 11 June 2013]

[29] Duetto, "What is duetto?", 11 June 2013, http://leaningtech.com/duetto/ [Accessed 11 June 2013]

[30] Web Cryptography, 4 June 2013, http://www.w3.org/2012/webcrypto/ [Accessed 11 June 2013]

[31] Khronos Group, "WebGL Specification version 1.0.2", 1 March 2013, https://www.khronos.org/registry/webgl/specs/1.0/ [Accessed 13 June 2013]

Appendix A

Appendix A innehåller de två intervjuer som utfördes under empiriska studien. Den förstnämnda Intervjun utfördes på Milou som är en webbyrå i Karlskrona. Den andra intervjun utfördes på Fujitu i karlskrona, som är ett mjukvaruföretag. Den första intervjun gjordes tillsammans med Andreas Björkblom, medans den senare utfördes tillsammans med Peter Svensson.

Intervju Milou

Hur ser din bakgrund ut inom programvaruutveckling?
Den tillfrågade svarar att han har gått från att utveckla applikationer i Flash till att utveckla webbapplikationer.

Vilka typer av applikationer har ni valt att göra i HTML5 i dagsläget?
Andreas har varit med och utvecklat applikationer i HTML5 genom webbyrån, med hjälp av ramverket Phonegap. Applikationerna som Andreas har varit med och utvecklat har i största del varit mobilapplikationer.

Har applikationerna varit helt byggda på HTML5, eller använt sig av andra tekniker samtidigt?
Andreas svarar att att applikationer dels har varit byggda helt i HTML5, men det har även förekommit HTML5 applikationer som har varit inbäddade i plattforms specifika applikationer. Majoriteten av applikationerna har använt sig av ett flertal tekniker som ligger utanför HTML5s omfattning.

Varför skapar ni HTML5 applikationer, isället för nativ applikationer.
Andreas svarar att HTML5 applikationer är billigare att utveckla, då de ska nås från flera olika plattformar. Andreas förklarar att samma kod går att använda på alla enheter, vilket sänker utvecklingskostnaderna.. Andreas tillläger att det är en starkt bidragande faktor för vilken typ av applikation kunden bestämmer sig för att välja.

Finns det nackdelar med HTML5 applikationer?
Andreas svarar att responsen från att användaren trycker på en knapp till att den önskade händelsen inträffar ofta är för lång, men betonar att det kan vara väldigt personligt och samma intryck inte fås av alla användare. Andreas nämner att det även skiljer mellan olika webbapplikationer och ger Facebooks mobilapplikation som ett exempel med bra responstid. Andreas nämner även att de webbläsare som många mobila enheter använder sig av inte har tillräckligt snabba JavaScriptsmotorer. Men betonar att detta är under ständig förbättring.

Vilken är den främsta egenskapen hos HTML5?
Den tillfrågade svarar att HTML5 är väldigt stort, och att en specifik favorit inte finns i dagsläget.

Finns det tillfällen då HTML5 inte är att föredra i jämförelse med tredje parts lösningar
Den tillfrågade finner inga sådana tillfällen.

Hur testar ni HTML5relaterad kod
Den tillfrågade svarar att företaget använder sig av kodgranskning, men att annan testning inte utförs.

Hur ser efterfrågan ut på crossplattform applikationer
Andreas svarar att efterfrågan på crossplattform applikationer ökar, eftersom priset är så pass mycket lägre. Samtidigt minskar efterfrågan applikationer för just en specifik plattform förklarar den medverkande.

Ser ni något som hindrar HTML5s utveckling
Den medverkande svarar att Internet Explorer är en tråkig flaskhals.

Tror ni HTML5 är framtiden för webbapplikationer
Andreas svarar att det är han tror att det blir så. Han ser inga direkta konkurrenter.

Kan RIA utvecklade applikationer med HTML5 ge motsvarande användarupplevelse hos slutanvändaren som skrivbordsapplikationer?
Andreas svarar att de har funnit att HTML5 applikationer kan ge motsvarande användarupplevelse hos slutanvändaren, dock att applikataioner utvecklade i HTML5 har en benägenhet att ha en sämre responstid vid interaktion.

Vad finns det för fördelar/hinder i att utveckla applikationer med HTML5?
Andreas svarar att ett av de störta hoten mot HTML5 är Internet explorer och andra webbläsare med sämre stöd för HTML5. Han nämner även testning som något svårt att utföra i applikationer skapade med stora mängder HTML5. Det finns även en viss rädsla att överge mer säkra tredjeparts alternativ för streaming av media så som Silverlight, då Silverligh har stöd för DRM.

Kan HTML5 applikationer uppnå liknande prestanda som skrivbordsapplikationer?
Inte i dagsläget, men läget förändras till det bättre hela tiden.

Intervju Fujitsu

Hur ser din bakgrund ut inom programvaruutveckling?
Peter svar att han har arbetat med XHTML och Java Server Faces, samt arbetat en del med php. Han svarar även att han arbetat mycket med Flash och Java generellt.

Ser du att det finns begränsningar med användandet av Flash?
Peter svarar att det svåra med Flash är att få till en bra kommunikation mellan klienten och server. Det blir ofta bara en engångsportal. Peter svarar även att Flash inte vet vad som finns utanför den ruta det körs i. Han förtydligar med att nämna att HTML element som ligger runt om Flash inte går att nå genom flash , något som han ofta saknar.

Hur är Flash Jämfört med Canvas?
Peter svarar att han inte har använt Canvas ännu, men nämner att det verkar väldigt intressant.

Hur skiljer sig utvecklingskostnaden mellan en crossplattform applikation och flera applikationer som är specifikt utvecklade för en viss plattform?
Peter svarar att det till stor del beror på vad applikationer skall utföra. Han betonar att en webbapplikation i många fall behöver en server som den kommunicerar med. Något som kan vara en nackdel lägger han till. Peter säger även att Java har sin virtuella maskin som gör det möjligt att köra applikationer på flera olika operativsystem.

Hur ser efterfrågan ut på cross-plattform applikationer.
Peter svarar att han tror att det ökar. Han har inga konkreta siffror men han vet att det är många kunder som inte vill att deras applikation skall vara budet till ett visst operativsystem.

Hur viktigt är cross plattform? Hur mycket kan man kompromissa med kvaliten?
Om man kör det via nätet blir det samma upplevelse för alla.

Peter svarar att kvalité är viktigt men även att det är ett väldigt brett område som innefattar väldigt mycket. Peter säger att det viktigaste är att funktionalitet finns, att saker tar lite längre tid och att det visuella inte alltid är så vackert som de kunde vara inte har lika stor betydelse.

Hur utför ni testning av era webbapplikationer
Peter svarar att den enda testning de utför är på server sidan, där de har tillgång till automatiserade enhetstestning, kodanalys , code coverage och liknande kvalitetsmätande instrument. Dock betonar han att han inte vet om det finns något sådant till klientbaserad kod, något som han tycker är synd. Peter nämner även att en applikation skiljer sig ofta mycket från en annan och då är det även naturligt att testningen ser helt olika ut beroende på vilken applikation som testas. Peter avslutar frågan med att säga att de har försökt använda sig av Celenium för testning men att det inte riktigt har fungerat någon vidare. Det har inte gett bra resultat och har även varit väldigt tidsödande.

Vilket är ert val när ni väljer göra interaktiva visuella element.
Peter svarar att de använder Flash och Java eftersom det är något de har erfarenhet om och fungerar bra.

Använder ni er av HTML5 i er programvaruutveckling?
Peter svarar att de använder sig av HTML5 i fyra till fem projekt som de just nu har igång.

Hur ser kompetensen ut angående HTML5 på företaget?
Peter svarar att det ser väldigt varierande ut. Vissa har väldigt bred kunskap om HTML5, men att den inte är vidare djup. Medan vissa inte har någon kompetens alls. Han nämner dock att det finns de på företaget som är väldigt duktiga inom HTML5.

Hur ser din syn på kvalité ut
Peter svarar att kvalité för honom är att de utlovade funktionella kraven är uppfylda..

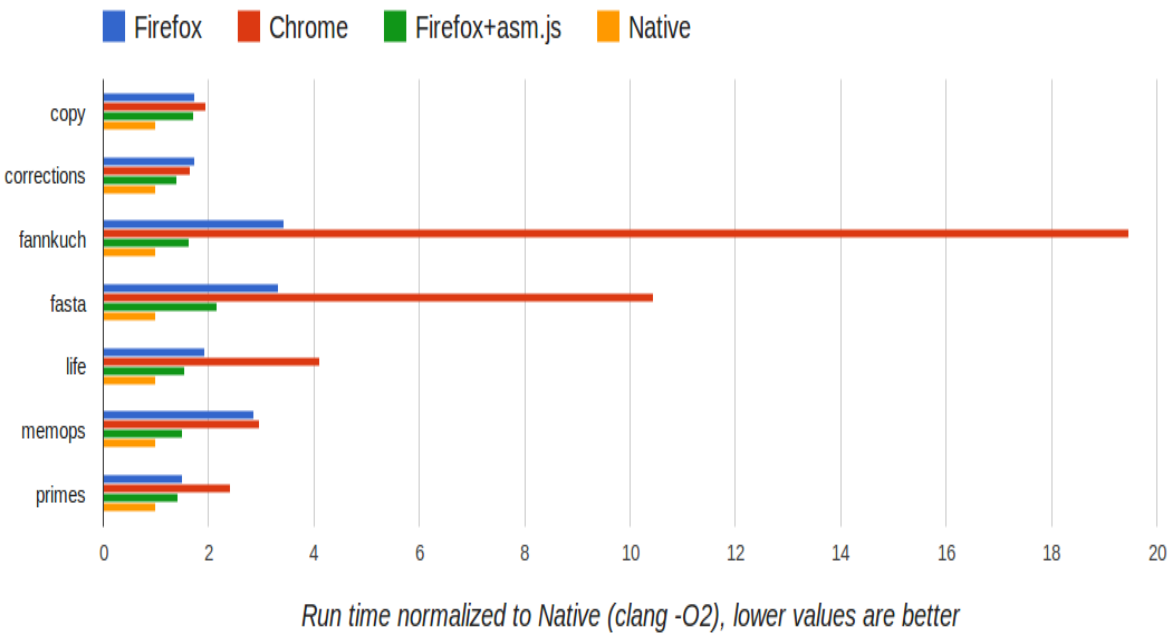
Vad är en okej responstid på en webbaapplikation?
Peter svarar att det beror på vad som skall utvecklas. Men han nämner att det maximalt får dröja en sekund från då man trycker till att funktionalliten har utförts. Peter betonar dock att om det är en massa beräkningar så får det ta längre tid , så länge användaren får feedback om detta.

Hur ser framtiden ut?
Peter svarar att han tror att det går mot att det blir mer webbaserat.

Appendix B

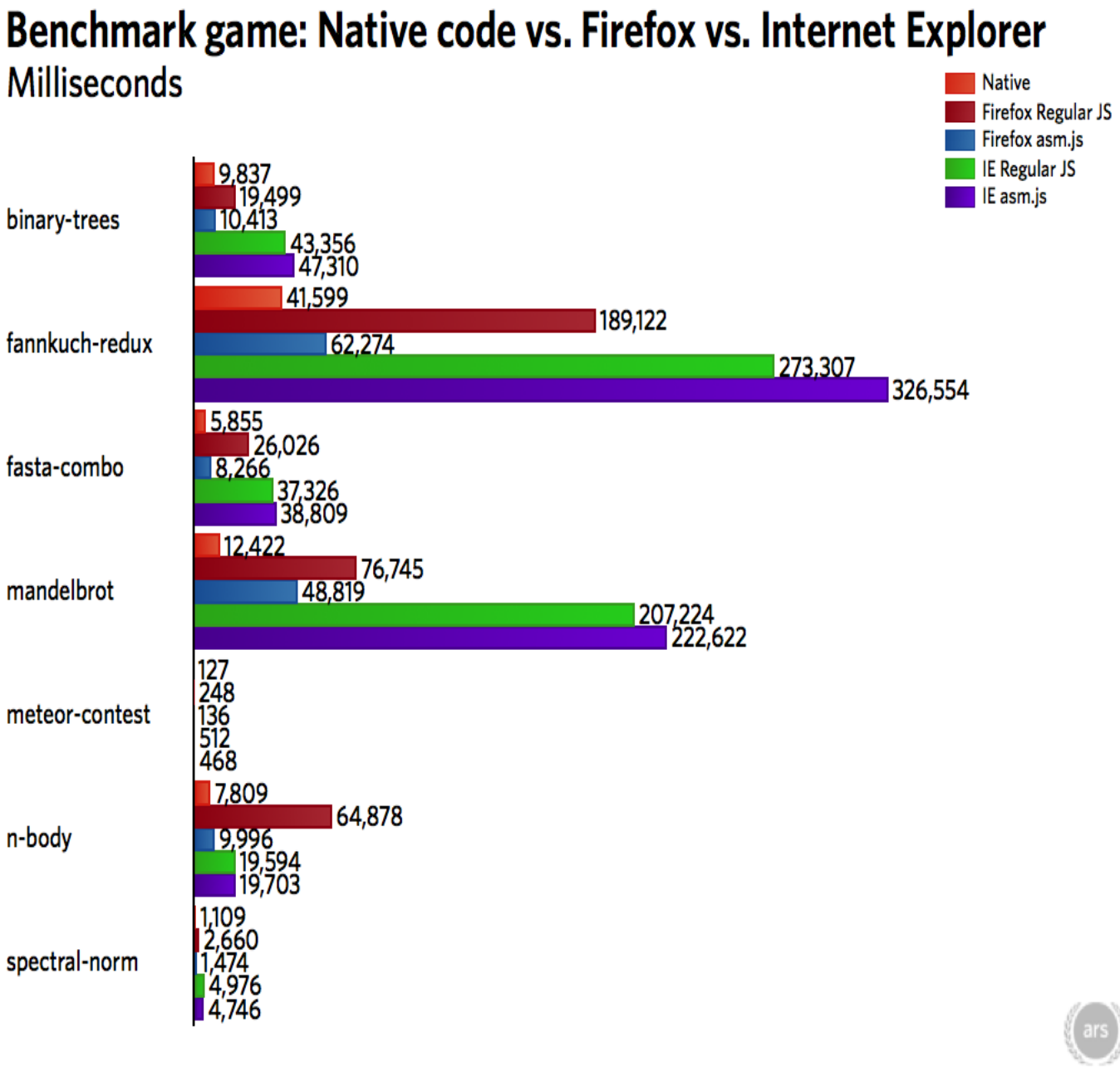
(Källa: http://kripken.github.io/mloc_emscripten_talk/gindex.html)

Microbenchmarks



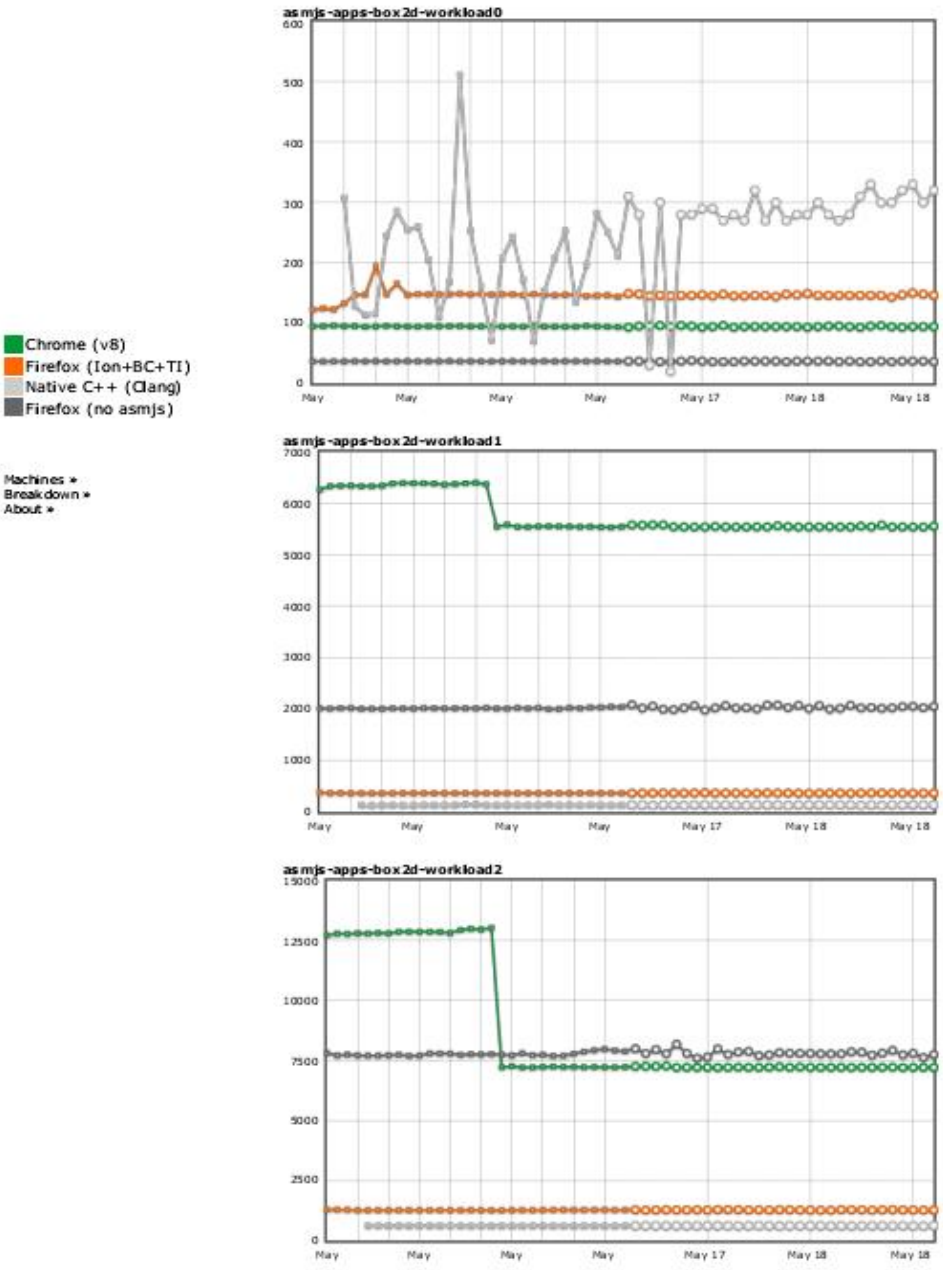
Appendix C

(Källa: http://arstechnica.com/information-technology/2013/05/native-level-performance-on-the-web-a-brief-examination-of-asm-js/)



Appendix D

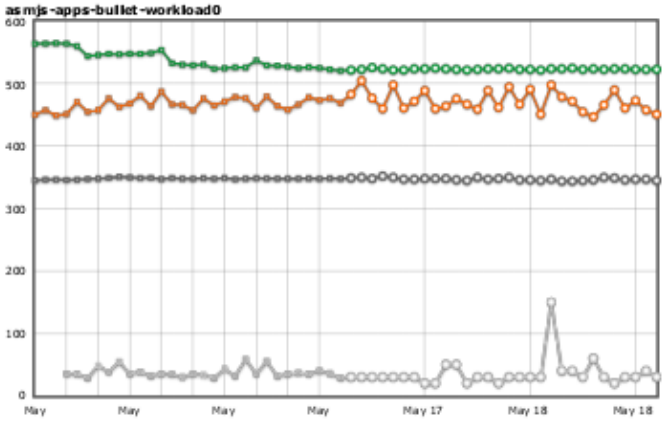
(Källa: http://arewefastyet.com/#machine=11&view=breakdown&suite=asmjs-apps)



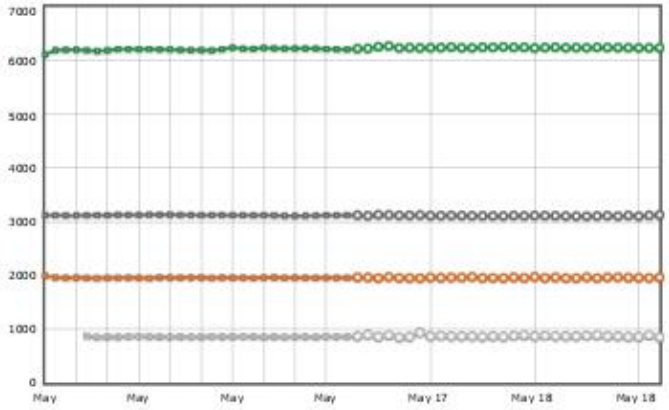
asmjs-apps-box2d-workload3

asmjs-apps-box2d-workload4

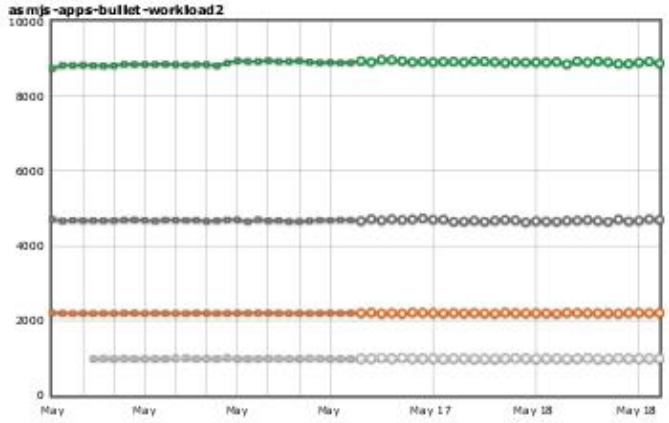
asmjs-apps-bullet-workload0



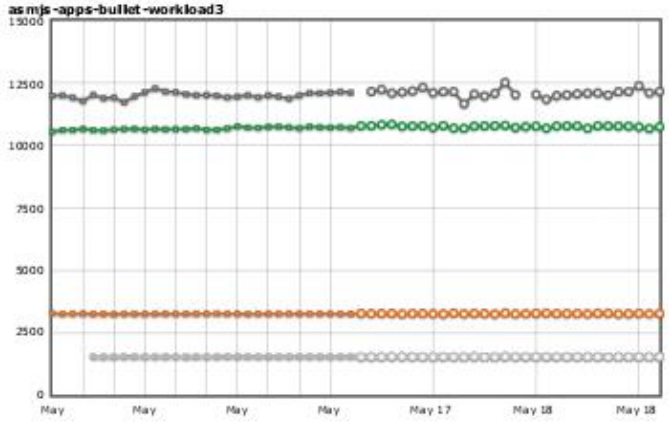
asmjs-apps-bullet-workload1



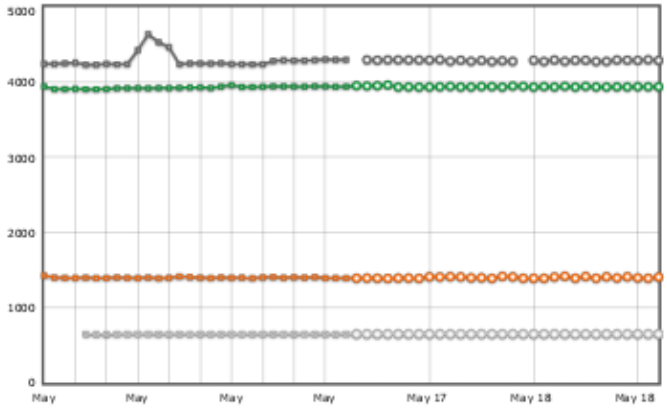
asmjs-apps-bullet-workload2



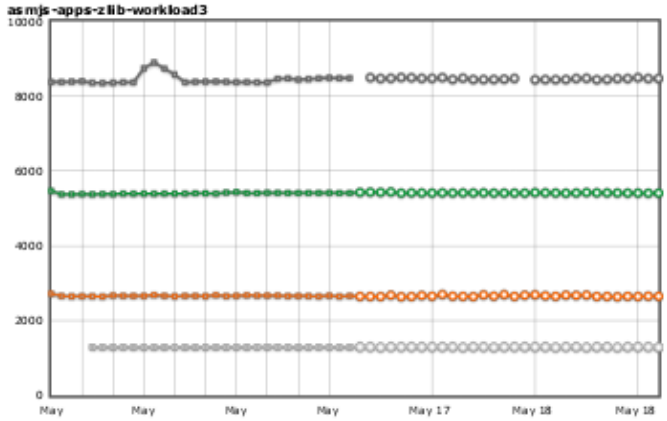
asmjs-apps-bullet-workload3



asmjs-apps-bullet-workload4



asmjs-apps-zlib-workload3



asmjs-apps-zlib-workload4