2014-06-19

Kandidatavhandling i Programvaruteknik
06 2014

En jämförande prestandastudie mellan JSON och XML

Joel Einarsson, Johannes Winger-Lang

Kontaktinformation:
Författare:
Johannes Winger-Lang
E-mail: wingerlang.johannes@gmail.com

Joel Einarsson
E-mail: joel.einarsson@gmail.com

Handledare:
Kari Rönkkö

Sektionen för datavetenskap och kommunikation
Blekinge Tekniska Högskola
371 79 Karlskrona
Sweden

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

Abstrakt

När man utvecklar en ny produkt eller tjänst står man ofta inför valet av dataformat. De mest använda idag är JSON och XML. Formaten ser väldigt olika ut, erbjuder olika funktioner, men används inte sällan till samma sak. Vilket som egentligen är snabbast finns det mycket åsikter om, men inte lika mycket testresultat. Den luckan skall detta arbete täcka.

Programmeringsspråken som används är Python och JavaScript, vilka båda är populära på webben.

Genom experiment testas hur snabbt JSON och XML kan kodas och avkodas. Testerna går ut på att XML och JSON konverteras till en lämplig intern datastruktur. I JavaScript är det ett Object, för Python är det en dictionary. Det testas även att konvertera från datastrukturen, till XML och JSON. Både stora och små datamängder testas.

Resultaten visar enhälligt att JSON är betydligt snabbare än XML, upp mot en faktor 100. För JavaScript gäller detta i de tre stora webbläsarna som testas; Google Chrome, Mozilla Firefox samt Internet Explorer. Det stämmer även för Python. Resultatet är detsamma för liten datamängd som för en stor datamängd.

Innehållsförteckning

1 Introduktion

Webb-API:er används mer och mer, därför att samma data kan placeras i flera olika sammanhang. Detta kan sänka utvecklingskostnader för företag eftersom samma API kan användas för flera olika tjänster. En motivationsfaktor för företag att tillhandahålla öppna API:er är att andra företag kan utveckla tjänster åt dem, med hjälp av datan som levereras i API:et. Det finns alltså en ekonomiskt drivande faktor för att tillhandahålla API:er.

Det finns dels öppna API:er (fria att använda) och stängda (kräver någon form av autentisering). Oavsett vilken typ av API det är så används något dataformat. De två vanligaste är XML och JSON. Det är dessa två format som kommer testas i studien.

Denna studie undersöker dataöverföringsformaten XML och JSON. Vi testar vilket format som har bäst prestanda i olika programmeringsspråk. Studien fokuserar på prestandan i de två programmeringsspråken Python och JavaScript.

I webben används mer och mer JavaScript, och de flesta webbutvecklare behöver kunna språket. JSON själv kommer ursprungligen ifrån JavaScript, vilket gör det till ett intressant språk att jämföra med.

Det finns även ett programmeringsspråk som heter Python, vilket är känt för att vara väldigt enkelt att förstå och komma igång med. Till webben finns även många populära ramverk, bland annat Django[1] och CherryPy[2]. Till studien valdes Python som ett av språken, eftersom det under litteraturstudien ej hittades några tidigare studier gjorda för detta programmeringsspråk, trots att Python används till flertalet stora webbsidor.[3]

Python är ett serverspråk, vilket betyder att koden exekveras, och levererar data (ofta i form av HTML, JSON eller XML). Python är alltså ett språk som kan tillhandahålla API:er. JavaScript däremot är ett klientbaserat språk, vilket betyder att koden körs på den dator som anropar en webbsida. Det betyder även att JavaScript kan *anropa* ett API. Detta i sin tur leder till att man kan skriva ett API med hjälp av Python, och ifrån sin JavaScriptkod anropa till detta. För att datan man hämtar skall kunna användas i språket, behöver det konverteras ifrån det format det var (exempelvis JSON eller XML), till någon sorts datastruktur. Den konverteringen kommer vi testa i denna studie.

Programmableweb indexerar API:er, och visar att XML och JSON har ungefär lika stor marknadsandel idag[4] [5]. Eftersom JSON ökar i popularitet (delvis på grund av JavaScripts

---

[1] *Meet Django*, hämtad 2014-05-22, <https://www.djangoproject.com/>

[2] *A Minimalist Python Web Framework*, hämtad 2014-05-22, <http://www.cherrypy.org/>

[3] Codecondo, *10 Popular Sites Powered by Django Web Framework*, 2013, hämtad 2014-06-08, <http://codecondo.com/popular-websites-django/>

[4] Programmableweb, *Web Services Directory*, hämtad 2014-03-25, <http://www.programmableweb.com/apis/directory/1?format=XML>

framfart), känns det ett bra tillfälle att jämföra det med XML, när de används lika mycket..

Litteraturstudien vi gjort visar på att det finns ett tomrum inom vårt område, alltså prestandaanalyser för JSON och XML. Dessutom hittas ingen studie med programmeringsspråken Python och JavaScript. Vår målsättning är att fylla det tomrum med vår undersökning. Eftersom det är två vanliga format i kombination med två populära programmeringsspråk vi testar, ser vi möjlighet till en stor målgrupp som har intresse av studien.

Målet med denna studie var att göra en prestandaanalys som webbprogrammerare kan ha nytta av när de ska välja format mellan XML och JSON. Resultatet av studien ger svar på vilket av överföringsformaten som har bäst prestanda i form av tidsåtgång i de två programmeringsspråken Python och JavaScript.

Vår hypotes var att JSON skulle att ha bättre prestanda i majoriteten, om inte samtliga, av våra tester. Detta trodde vi eftersom strukturen på JSON-data är väldigt lik strukturerna för dictionary i Python och objekt i JavaScript, vilket vi trodde skulle snabba upp konverteringen. Vi visste dock inte om skillnaden var marginell, eller väldigt påtaglig. Uppsatsen riktar sig främst till webbprogrammerare som står inför valet av överföringsformat till sina projekt. De kan använda sig av våra resultat för att ta ett informerat beslut i vilket format de ska välja.

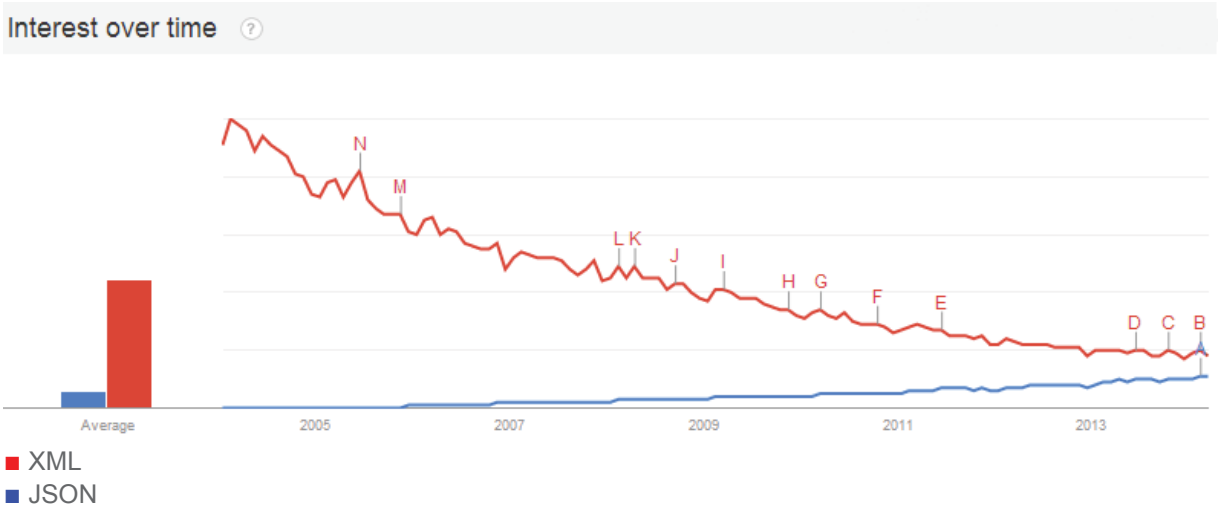
Studien visar att om hastighet viktas högt, bör man välja JSON över XML. Detta gäller oavsett om man programmerar i Python eller JavaScript och oavsett vilken av de webbläsare vi testat man använder sig av.

2 Bakgrund

JSON[6] och XML[7] är två olika format för att skicka, lagra, samt hantera data. Dessa två format utgör en stor majoritet av formaten som används till dessa ändamål idag. ProgrammableWeb rapporterar att av de 11 200 API:er de har indexerade, använder 6 093 XML[8] och 5 305 JSON[9]. Google trends visar att sökningar på XML kraftigt minskar, medan JSON svagt ökar (se Figur 2.1).

---

[5] Programmableweb, *Web Services Directory*, hämtad 2014-03-25, <http://www.programmableweb.com/apis/directory/1?format=JSON>
[6] D Crockford, *Introducing JSON*, hämtad 2014-03-31, <http://www.json.org>
[7] W3Schools, *XML Tutorial*, hämtad 2014-03-31, < http://www.w3schools.com/xml/>
[8] Programmableweb, *Web Services Directory*, hämtad 2014-03-25, <http://www.programmableweb.com/apis/directory/1?format=XML>
[9] Programmableweb, *Web Services Directory*, hämtad 2014-03-25, <http://www.programmableweb.com/apis/directory/1?format=JSON>



Figur 2.1: Google Trends rapporterar hur mycket de båda nyckelorden XML och JSON sökts på Google i intervallet 2004 - 2014. Den röda linjen avser XML, den blå JSON.

2.1 API och webb-API

API står för Application Programming Interface och är en välanvänd term inom programmering. Termen används för att beskriva funktionalitet i ett program, som är till för att andra program skall kunna kommunicera med det.[10]

API:erna som refereras till i detta arbete är mer specifikt webb-API:er. Dessa API:er kommunicerar ofta via HTTP-anrop, det vill säga en förfrågan att hämta eller ändra en resurs på en viss webbadress.

2.2 XML

XML står för Extensible Markup Language. Den första versionen - en så kallad working draft - publicerades av W3C i November 1996[11]. Det främsta målet var att kunna använda generisk Standard Generalized Markup Language (SGML), vilket är en standard som definieras enligt ISO 8879:1986[12], på samma sätt som HTML. Andra mål de hade var; enkelhet, kompatibilitet och läsbarhet. XML ersatte SGML.

XML används främst för två olika syfte; Representera låg-nivå data, samt lägga till metadata (data om data) för dokument. XML finns bland annat som literaler i Visual Basic[13] och inkluderat

[10] Encyclopaedia Britannica, *API (computer programming), 2014, hämtad 2014-06-07, <http://www.britannica.com.miman.bib.bth.se/EBchecked/topic/1472947/API>*
[11] W3C, *Extensible Markup Language (XML)*, 1996, hämtad 2014-03-25, <http://www.w3.org/TR/WD-xml-961114>
[12] ISO, *ISO 8879:1986*, hämtad 2014-03-25, <http://www.iso.org/iso/catalogue_detail.htm?csnumber=16387>
[13] Microsoft, *XML Literals Overview (Visual Basic)*, 2014, hämtad 2014-03-25, <http://msdn.microsoft.com/en-us/library/bb384629.aspx>

i programmeringsspråket Scalas standard-bibliotek[14].

Nedan följer ett kort exempel på hur XML ser ut. Exemplet beskriver hur ett brev kan kodas med hjälp av XML.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <note>
3     <to>Tove</to>
4     <from>Jani</from>
5     <heading>Reminder</heading>
6     <body>Don't forget me this weekend!</body>
7 </note>
```

Figur 2.2: XML-exempelkod, i detta fall ett brev.

Det måste alltid finnas en rot-nod, i det här fallet är det *note*. Där efter följer taggar med tillhörande data. Det finns även attribut i XML, vilket ser ut så här:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <person id="100">
3     <name first="Jane" last="Doe"></name>
4 </person>
```

Figur 2.3: Exempel på attribut i XML. "id", "first" och "last" är attribut.

2.3 JavaScript och Ajax

Den ursprungliga versionen av JavaScript utvecklades av Brendan Eich 1995, på endast tio dagar. Utvecklingsnamnet var Mocha, vilket senare byttes till LiveScript. I och med Javas uppgång ändrades det sedan till JavaScript. 1996-1997 ville man standardisera språket, varav man tog det till European Computer Manufacturers Association (ECMA).

Nästa genombrott skedde den 18e februari 2005 då Jesse James Garrett skrev den berömde artikeln Ajax: A New Approach to Web Applications[15]. Garrett myntar här begreppet *Ajax*, vilket är en kortform för Asynchronous JavaScript + XML. Kortfattat är det en rad tekniker för att i bakgrunden ladda in information, istället för att kräva en hel sidomladdning. Garrett påpepar också att han inte uppfunnit det (Till exempel säger han att Google redan använder det på flertalet nya sidor), utan att han definierar tekniken. Trots att XML är en del av namnet i Ajax, så är det oberoende av dataformatet, även JSON går utmärkt att använda.

[14] *Scala.xml API*, hämtad 2014-03-25, <http://www.scala-lang.org/api/2.10.3/index.html#scala.xml.package>
[15] Jesse James Garrett, Ajax: A New Approach to Web Applications, hämtad 2014-04-13, <http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/>

2.4 JSON

JavaScript Object Notation (JSON) är ett språkoberoende, lättviktigt dataöverföringsformat baserat på JavaScripts objektlitteralnotation. Det kan användas för att överföra data mellan program skrivna i alla moderna programmeringsspråk. JSON är ett textbaserat format som är läsbart både av människor och datorer.[16]

JSON är definerat både i form av en informativ RFC, RFC 4627[17], och som en ECMA-standard, ECMA-404[18].

JSON växte fram när Douglas Crockfords företag *Veil Networks* behövde ett effektivt sätt att hämta och skicka data mellan en webbläsare och en server utan att behöva ladda om en hel sida. De tittade på XML, men tyckte inte alls det var vad de ville ha. Crockford kom då på att JavaScripts objektlitteralnotation borde vara perfekt för ändamålet då JavaScript förstår det och eftersom det representerar datan exakt som den är. Efter ett fåtal justeringar, så som att nycklarna skulle vara omslutna av citationstecken, hade JSON fötts.[19]

Enligt Crockford blev JSON populärt som en effekt av att AJAX blev populärt inom webbutveckling. Det fick till följd att JSON plockade marknadsandelar från XML. De tidiga användarna av AJAX tyckte att XML var för krångligt som överföringsformat och såg potentialen i JSON eftersom det representerade data på nästan exakt samma sätt som JavaScript.[20]

XML-exemplet på ett brev ovan kan i JSON-format representeras så här:

```
{
    "to": "Tove",
    "from": "Jani",
    "heading": "Reminder",
    "body": "Don’t forget me this weekend!"
}
```

Figur 2.4: JSON-exempelkod, även i detta fall ett brev.

Notera att det ingenstans explicit står vilken typ av objekt det är (note), vilket det i XML-exemplet gjorde. Det finns inte heller attribut som XML erbjuder.

---

[16] D Crockford, *JavaScript The Good Parts*, O'Reilly Media, Sebastopol, CA, 2008, p. 136
[17] D Crockford, *JSON*, IETF Trust, 2006, hämtad 2014-03-10, <https://www.ietf.org/rfc/rfc4627.txt>
[18] Ecma International, *The JSON Data Interchange Format*, 2013, hämtad 2014-03-10, <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>
[19] D Crockford, *The JSON Saga by Doug Crockford*, 2011, hämtad 2014-03-10, <http://www.youtube.com/watch?v=x92vbAN_j1k>
[20] D Crockford, *Douglas Crockford: The JSON Saga*, 2009, hämtad 2014-03-20, <https://www.youtube.com/watch?v=-C-JoyNuQJs>

2.5 Syfte och forskningsfrågor

Syftet med denna studie är att besvara dessa två frågor:

1. Vilket av dataöverföringsformaten XML och JSON är snabbast på att konvertera till och från en bearbetbar datastruktur (dict) i programmeringsspråket Python?
2. Vilket av dataöverföringsformaten XML och JSON är snabbast på att konvertera till och från en bearbetbar datastruktur (object) i programmeringsspråket JavaScript?

Frågorna är valda med hänsyn till webbprogrammerare. Konvertering mellan ett överföringsformat som JSON eller XML till datastrukturer i Python och JavaScript är något som ofta behöver utföras av personer som jobbar med webbprogrammering. Frågorna ger inte bara en inblick på vilket format man bör välja, men också om programmeringsspråken i sig är snabba på att hantera de båda formaten.

3 Metod

För att besvara de två forskningsfrågorna använde vi oss av två metoder. Dels begrundade vi tidigare skrivna artiklar som vi kunde referera till och hämta fakta från, dels använde vi oss av experiment.

När vi har undersökt tidigare gjorda prestandastudier[21] [22] [23] [24] [25] har vi märkt att samtliga prestandastudier mäter tiden det tar att utföra olika experiment. Slutsatsen vi drar från det är att exekveringstid är en av de viktigaste, om inte den mest viktiga, aspekten i en prestandastudie. Därför har vi valt exekveringstid som vårt mätvärde för prestanda.

För experimenten skrev vi simpla program, som enbart gör det som behövs för att kunna få tillförlitliga mätvärden. Detta för att eliminera så många externa påverkande faktorer som möjligt. Vi exekverade testerna 5000 gånger för att få ut rimligare, mer lättlästa siffror och för att minska effekten av avvikelser. Vi körde även våra tester på två olika datorer med olika hårdvara.

3.1 Utformning av litteraturstudie

För att få fram relevant litteratur till arbetet sökte vi igenom olika databaser. Vi hade en fördefinierad lista med söktermer som vi utgick ifrån först och främst. Hittade vi något extra

---

[21] Nurseitov, Paulsen, Reynolds, Izurieta, *Comparison of JSON and XML Data Interchange Formats*, 2012, <http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf>

[22] Guanhua Wang, *Improving Data Transmission in Web Applications via the Translation between XML and JSON*, 2011

[23] Maeda, K, *Performance Evaluation of Object Serialization Libraries in XML, JSON and Binary Formats*, 2012

[24] D Peng, L Cao, W Xu, *Using JSON for Data Exchanging in Web Service Applications*, 2011, <http://www.jofcis.com/publishedpapers/2011_7_16_5883_5890.pdf>

[25] B Lin, Y Chen, X Chen, Y Yu, *Comparision between JSON and XML in Applications on AJAX*, 2012

intressant i artiklarna vi sökt fram så utökade vi vår lista med söktermer.

Sökning genomfördes i följande databaser och webbsidor:

- Engineering Village
- IEEE
- BTH Summon
- Google Scholar
- Google
- Youtube

Söktermerna som användes var:

- JSON
- XML
- JSON, XML
- JSON vs XML
- JSON as data transfer format
- XML as data transfer format
- data exchange format
- JSON performance
- XML performance

Majoriteten av de valda artiklarna hittades på Engineering Village. I vissa fall gick det inte att hämta hem intressanta artiklar som hittats där. Då hjälpte det oftast att söka i någon av de andra databaserna och hämta hem dem därifrån istället.

För att veta vilka artiklar vi skulle välja ut bestämde vi kriterier som de var tvungna att passera. Artiklar som presenterar prestandaresultat fick inte vara gjorda innan 2004. Anledningen till detta är att datorer och servrar har utvecklats mycket sedan dess, och något som var avgörande för prestandan på den tiden kan idag vara helt irrelevant. Dessutom har XML, JavaScript och Python kommit ut i nya versioner sedan dess. Vi filterade även bort artiklar som verkade uppenbart vinklade. För de artiklar som berör JavaScript läggs större fokus på de som är senare än 2008. Motiveringen är att det är ungefär då Google Chrome släpptes med deras JavasScriptmotor V8[26]. Data tidigare än så riskerar därför att vara inaktuell.

För att avgöra om en artikel var relevant för vårt område läste vi först och främst abstrakt. Var den relevant gick vi vidare och läste inledning och resultat. Var även de relevanta så läste vi hela artikeln.

Vi hittade fem artiklar som vi ansåg tillräckligt relevanta för att benämnas som huvudlitteratur. Den gemensamma nämnaren är någon form av jämförelse av formaten med fokus på prestanda. Vi hittade inte speciellt många artiklar med den inriktningen, vilket betyder att vi inte heller filtrerade bort speciellt många. Utöver det samlade vi in åtta referenser som går under

[26] Google Inc, *A fresh take on the browser*, hämtad 2014-05-14 <http://googleblog.blogspot.se/2008/09/fresh-take-on-browser.html>

rubriken övrig litteratur. Det som presenteras där är böcker, något mindre relevanta artiklar samt blogginlägg och övriga uttalanden.

3.2 Utformning av experiment

Experimenten utfördes med hjälp av två minimala program, ett i Python och ett i JavaScript. Programmens uppgift var att läsa upp JSON/XML från en sträng och konvertera det till ett format som går att bearbeta direkt i koden (dict för Python, object för JavaScript). Programmen skulle även kunna konvertera från dict/object till JSON/XML.

Tester med varierande mängd data har körts för att täcka fler användingsfall och för att få så korrekta resultat som möjligt. Datan är syntetiskt, men efterliknar sättet riktig data är uppbyggd på.

Fokus på testerna har varit att mäta exekveringstiden, det vill säga hur lång tid det tar att utföra en viss uppgift. Tidtagning i JavaScript har utförts med hjälp av funktionen console.time[27] som startats när första iterationen påbörjats och stoppats när sista iterationen avslutats. Tidtagning i Python har skett med hjälp av pythonmodulen timeit[28]. Även för testerna i Python startades tidtagningen när första iterationen påbörjades och stoppades när sista iterationen avslutats. Tiden som mättes var totaltiden för alla 5000 körningar. Pythonversionen som användes var Python 2.7.6 32-bit.

JavaScripttesterna kördes i följande webbläsare:

- Chrome 34
- Firefox 29
- Internet Explorer 11

Testerna kördes i tre olika webbläsare för att täcka de senaste versionerna av de (enligt Statcounter) mest använda webbläsarna[29]. Anledningen till att vi ville testa i flera moderna webbläsare är att de har olika JavaScript-implementationer och vi ville se om det kunde ha någon inverkan på resultatet. Potentiella skillnader kan vara att någon webbläsare tar betydligt längre tid på sig att utföra vissa operationer än de andra.

Programmen kördes på två olika datorer. Detta gjordes för att minska risken att felkonfigureringar och externa faktorer skulle kunna påverka resultaten.

Samtliga testfall kördes 5000 gånger. Dels för att få mer lättläsliga siffror som resultat och dels för att minska påverkan av potentiella avvikelser. Vi såg ingen anledning att köra testerna fler gånger.

---

[27] Mozilla Developer Network, *console.time*, hämtad 2014-05-15, <https://developer.mozilla.org/en-US/docs/Web/API/console.time>
[28] Python documentation, *timeit*, hämtad 2014-04-25, <https://docs.python.org/2/library/timeit.html>
[29] Statcounter, *Statcounter Global Stats,* 2014, hämtad 2014-03-30, <http://gs.statcounter.com/>

Testerna som utfördes i Python var följande:

ID	Från	Till	Kodare/Avkodare	Iterationer
1	JSON-sträng	dict	json.loads[30]	5000
2	XML-sträng	dict	cElementTree[31] + etree_to_dict[32]	5000
3	dict	JSON-sträng	json.dumps[33]	5000
4	dict	XML-sträng	dict2xml[34]	5000

Figur 3.1: En förklarande tabell till hur testerna kördes, och vilka bibliotek/funktioner som användes.

I JavaScript kördes följande tester:

ID	Från	Till	Kodare/Avkodare	Iterationer
5	JSON-sträng	object	JSON.parse[35]	5000
6	XML-sträng	object	X2JS.xml_str2json[36]	5000
7	object	JSON-sträng	JSON.stringify[37]	5000
8	object	XML-sträng	json2xml[38]	5000

Figur 3.2: En förklarande tabell till hur testerna kördes, och vilka bibliotek/funktioner som användes.

XML- och JSON-strängarna som testerna utgick ifrån finns i två olika varianter. En med en liten mängd data och en med en stor mängd data. JSON-strängarna sparade på fil tar upp 133 bytes för den lilla mängden data och 15,4 KB för den stora mängden data. XML-strängarna tar upp 179 bytes och 18,3 KB för liten respektive stor mängd data. Anledningen till storleksskillnaden är att XML kräver fler tecken för att representera ett objekt än JSON. Trots storleksskillnaden beskriver datan samma objekt. Se bilagor A, B, C och D för fullständig datamängd.

---

[30] *18.2. json — JSON encoder and decoder*, Python documentation, hämtad 2014-05-15, <https://docs.python.org/2/library/json.html#json.loads>

[31] *19.7. xml.etree.ElementTree — The ElementTree XML API*, Python documentation, hämtad 2014-05-15, <https://docs.python.org/2/library/xml.etree.elementtree.html>

[32] K3--rnc, *Converting xml to dictionary using ElementTree*, stackoverflow, 2012, hämtad 2014-05-15, <http://stackoverflow.com/a/10076823>

[33] *18.2. json — JSON encoder and decoder*, Python documentation, hämtad 2014-05-15, <https://docs.python.org/2/library/json.html#json.dumps>

[34] delfick, *python-dict2xml,* GitHub, 2014, hämtad 2014-05-15, <https://github.com/delfick/python-dict2xml>

[35] evilpie, Sheppy, fscholz, ethertank, Waldo, marcello.nuccio, madarche, *JSON.parse()*, MDN, 2014, hämtad 2014-05-15, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse>

[36] *x2js*, Google code, 2014, hämtad 2014-05-15, <https://code.google.com/p/x2js/>

[37] evilpie, Sheppy, Waldo, fscholz, Nickolay, miller_augusto, Enaeseth, paul.irish, *JSON.stringify()*, MDN, 2014, hämtad 2014-05-15, <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify>

[38] S Goessner, *json2xml*, goessner.net, 2006, hämtad 2014-05-15, <http://goessner.net/download/prj/jsonxml/>

3.2.1 Testsystem

	System 1	System 2
Operativsystem	Windows 8.1 Pro 64-bit	Windows 8.1
Processor	Intel Core i7-3630QM, 2.40 GHz	Intel Core i5-3230M, 2.6 GHz
Minne	16 GB	8 GB DDR3

Figur 3.3: De båda system, det vill säga datorerna som testerna kördes på.

4 Teori

Nedan beskrivs kortfattat de mest relevanta artiklar vi hittat. Därefter kommer den sekundära litteraturen, vilket är böcker, närbesläktade artiklar, bloggartiklar och liknande.

4. 1 Huvudlitteratur

I artikeln *Comparison of JSON and XML data interchange formats: A case study*[39] jämför man JSON och XML. Det är en konferensartikel som publicerades 2009. Deras jämförelse bygger på tester som de genomfört. De använder sig av ett klient-server program som de programmerat i Java, med syftet att simulera ett verkligt scenario. Deras mätvärde är överföringshastighet, CPU-belastning och minnesanvändning. Slutsatsen är att JSON generellt sett är betydligt snabbare än XML. Minnesanvändingen är dock densamma, och i vissa fall använder XML mindre CPU än JSON, dock under en mycket längre tid.

I *Improving Data Transmission in Web Applications via the Translation between XML and JSON* [40], en konferensartikel från 2011, fokuserar man på konvertering mellan XML och JSON. Författaren, Guanhua Wang, skriver om hur man korrekt kan konvertera data mellan de två formaten. Eftersom XML bygger på hierarki och JSON på nyckel-värde (Eng. key-value pair), finns det ibland inte en helt uppenbar ett-till-ett-mappning, men oftast finns det lösning som gör att man inte tappar någon information. Wang analyserar prestandan genom en rekursiv algorithm i språket PHP. Artikeln är ingen direkt jämförelse utan beskriver bara att det är möjligt att konvertera mellan formaten. Man beskriver även komplexiteten for algoritmerna. Sammanfattningen är att man ska utgå ifrån det mest lämpliga formatet, men att vid behov är det inget större problem att konvertera förlustfritt mellan XML och JSON, inom en rimlig tidsrymd.

I en konferensartikel från 2012 med titeln *Performance Evaluation of Object Serialization Libraries in XML, JSON and Binary Formats* [41]jämför man prestanda samt effektivitet för tolv olika Java-bibliotek. Serialisering och deserialisering av XML, JSON och binärformat testas.

---

[39] Nurseitov, Paulsen, Reynolds, Izurieta, Comparison of JSON and XML Data Interchange Formats, 2012,
<http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf>

[40] Guanhua Wang, Improving Data Transmission in Web Applications via the Translation between XML and JSON, 2011

[41] Maeda, K, Performance Evaluation of Object Serialization Libraries in XML, JSON and Binary Formats, 2012

De går till väga genom att spara ner och serialisera objekten i filer, och mäter sedan storleken på dem. De mäter tiden det tar att serialisera och deserialisera objekten. Artikeln är väldigt tydlig, då de frekvent använder färglagda grafer och sorterade tabeller. De går igenom olika bibliotek, t.ex. Gson (utvecklat av Google). Slutsatsen är att biblioteken har olika fördelar, men det finns ingen *bästa* lösning.

I *Using JSON for Data Exchanging in Web Service Applications*[42] skriver författarna om hur man på ett effektivt sätt processar och serialiserar JSON-data i webbtjänster. Modellen de föreslår att man ska använda kallas för Dynamic Advanced Binding, då det effektivt mappar JSON-datan mot det programmeringsspråk man använder sig av. De analyserar även prestandan vid serialisering och deserialisering av JSON-data. Resultaten jämförs med resultat för XML-baserade webbtjänster och slutsatsen är att JSON är snabbare på serialisering, deserialisering och parsning.

Studien *Comparison between JSON and XML in applications on Ajax*[43] jämför JSON med XML genom experiment. Deras resultat visar att JSON är det bättre formatet för att hantera data. De påpekar att JSON har små brister, men de tror att de kommer bli åtgärdade i framtiden. Artikeln publicerades 2012 och är en konferensartikel.

För att sammanfatta vad litteraturen säger, verkar JSON rent allmänt vara betydligt snabbare än XML. Minnesanvändningen (under konvertering) verkar dock vara ungefär densamma. De menar även att inget format är en *silverkula* (komplett, bästa lösning), vilket betyder att båda är bra i olika sammanhang. Det är även möjligt att konvertera förlustfritt mellan XML och JSON, vilket fungerar åt båda hållen.

4.2 Övrig litteratur

Förutom prestandatester har vi även använt en bok samt populära bloggar. Motiveringen till detta är att få bättre koll på vad industrin anser. Dessa källor kompletterar huvudlitteraturen, och ger en bättre blick på vad som används på riktigt.

Beginning XML 5th Edition är en bok på 864 sidor som tar upp det mesta om XML. Vi har främst använt boken för kunskap om vad XML erbjuder för egenskaper, men också om hur, när och varför XML kom till.

I IEEEs Computer, volume 45, Issue 4 hittar man artikeln *Discovering JavaScript Object Notation*[44], som är en intervju med Douglas Crockford av Dr. Charles Severance. Artikeln behandlar historien kring hur/varför JSON kom till. Det tas även upp fördelar/nackdelar, både för JSON och XML.

---

[42] D Peng, L Cao, W XU, *Using JSON for Data Exchanging in Web Service Applications*, 2011, <http://www.jofcis.com/publishedpapers/2011_7_16_5883_5890.pdf>
[43] B Lin, Y Chen, X Chen, Y Yu, *Comparision between JSON and XML in Applications on AJAX*, 2012
[44] Charles Severance, Discovering JavaScript Object Notation, april 2012 <http://www.computer.org/csdl/mags/co/2012/04/mco2012040006.pdf>

Kasia Milkoluk på Udemy, skriver att JSON är överlägset XML när handlar om dataformat, men påpekar att XML är betydligt bättre när det handlar om dokument, eftersom XML har stöd för bilder, grafer och liknande[45].

Mashery menar att XML är lämpligast om man behöver mer formella data-definitioner (exempelvis definiera vilken typ en viss data har). I och med att REST-API växer, eftersträvar fler och fler enkla och snabba API:er. De anser därför att JSON kommer vinna i längden.[46]

Som en motpol i JSON-hyllandet kan man läsa i *XML Can Give the Same Performance as JSON*[47] att prestandan kan vara likvärdig. I studien används 0,1-1MB stora dokument i JSON respektive XML. Drygt 1200 unika tester i de mest vanliga webbläsarna och operativsystemen körs. I kommentarsfälten kan man dock läsa lite kritik angående datan som används. Vissa påpekar att strukturen är enbart hierarkisk, vilket ju skulle vara en fördel på förhand till XML. Oavsett är det ett intressant inlägg.

26 april 2013 skrev användaren *drrwebber* en bloggartikel på Oracle.com med titeln *Analysis of JSON use cases compared to XML* [48]. Några av fördelarna för XML är: Inbyggd säkerhet, bättre datavisning (de nämner att JSON kan konverteras till XML för samma funktionalitet), bättre semantik samt bättre stöd för olika dataformat (exempel som nämns är ljud). Artikeln ställer sig dock neutral huruvida det ena är bättre än det andra, deras uppgift är enbart att påvisa olika användarområde.

Justin Cormack gör en jämförelse mellan JSON och XML[49]. Han listar fördelar till XML så som attribut och en bättre datamodell. Cormack menar dock att man skall anpassa formatet till målet med användningen. Han menar även att XML skullle behöva vara mindre, och pratar om hur man skulle kunna genomföra en sådan standard.

Nicholas C. Zakas inledde i början av 2008 en artikel[50] med följande citat:

> *"Use JSON, not XML. You hear it all the time these days. JSON is so much better. It must be because everyone is saying it, right? I'm starting to think not.*
>
> *Fellow Yahoo Douglas Crockford, creator/inventor/founder/father of JSON, says that*

---

[45] Kasia Milkoluk, *JSON vs XML*, hämtad 2014-04-25, <https://www.udemy.com/blog/json-vs-xml/>

[46] Rob Zazueta, *API Data Exchange: XML vs. JSON*, hämtad 2014-04-26, <http://www.mashery.com/blog/api-data-exchange-xml-vs-json>

[47] Jan Stenberg, *XML Can Give the Same Performance as JSON*, hämtad 2014-04-26, <http://www.infoq.com/news/2013/08/xml-json-performance>

[48] drrwebber, Analysis of JSON use cases compared to XML, hämtad 2014-05-06, <https://blogs.oracle.com/xmlorb/entry/analysis_of_json_use_cases>

[49] Justin Cormack, *JSON vs XML*, hämtad 2014-05-03, <http://blog.technologyofcontent.com/2010/01/json-vs-xml/>

[50] Nicholas C. Zakas, *Is JSON better than XML?*, hämtad 2014-05-06, <http://www.nczonline.net/blog/2008/01/09/is-json-better-than-xml/>

> *JSON is better than XML. He says it whenever he can (most recently in his post Does XML have a future on the web?). It is, of course, natural that he would say that JSON is better than XML; every parent thinks their kid is adorable and quite possibly the smartest kid ever. I'm just not sure his hypothesis is based in fact."*

Zakas förklarar varför han anser D. Crockford vara partisk. Han fortsätter sedan med att, relativt neutralt, lista för och-nackdelar med de båda formaten, både på klient och backend-sidan. Fördelar till JSON är bland annat säkerhet, konvertering till andra format och *frågor* (engelska querying).

5 Resultat

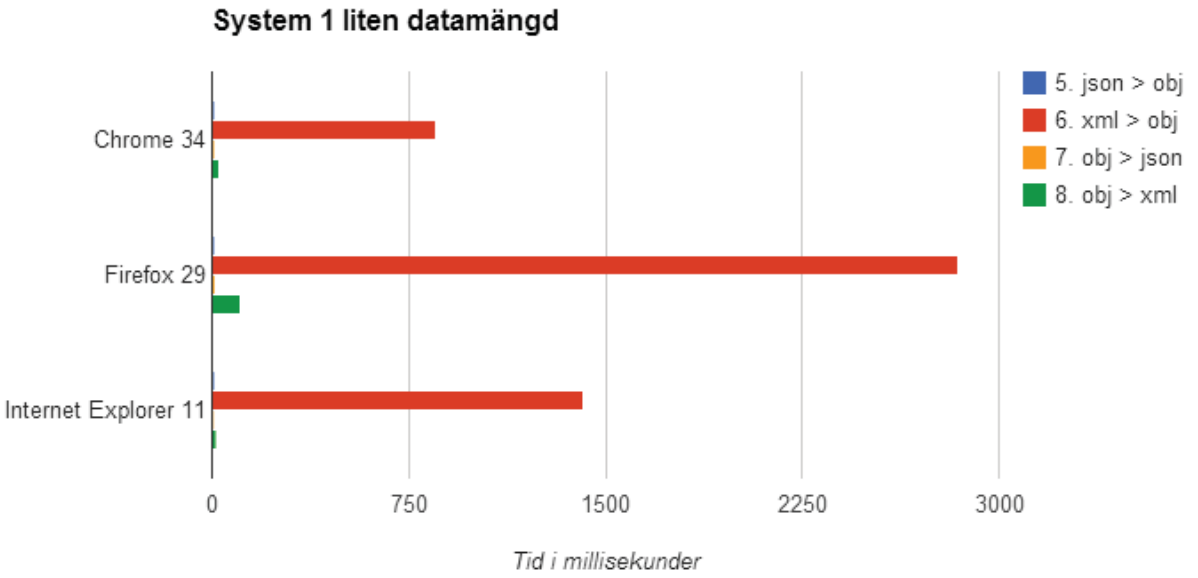
Nedan presenteras mätdatan vi samlat in genom våra experiment. Först kommer en tabell, och därefter följer ett färglagt diagram med samma data för tydligare visualisation. Vi har separata tabeller för de olika datorsystemen.

5.1 Resultat JavaScript liten datamängd

5.1.1 System 1

ID	Chrome 34	Firefox 29	Internet Explorer 11
5. json > obj	6 ms	7 ms	7 ms
6. xml > obj	851 ms	2 841 ms	1 413 ms
7. obj > json	6 ms	9 ms	5 ms
8. obj > xml	24 ms	107 ms	14 ms

Figur 5.1: Tabell över tiderna det tog att köra test 5-8 på system 1, i de tre valda webbläsarna och med liten datamängd.



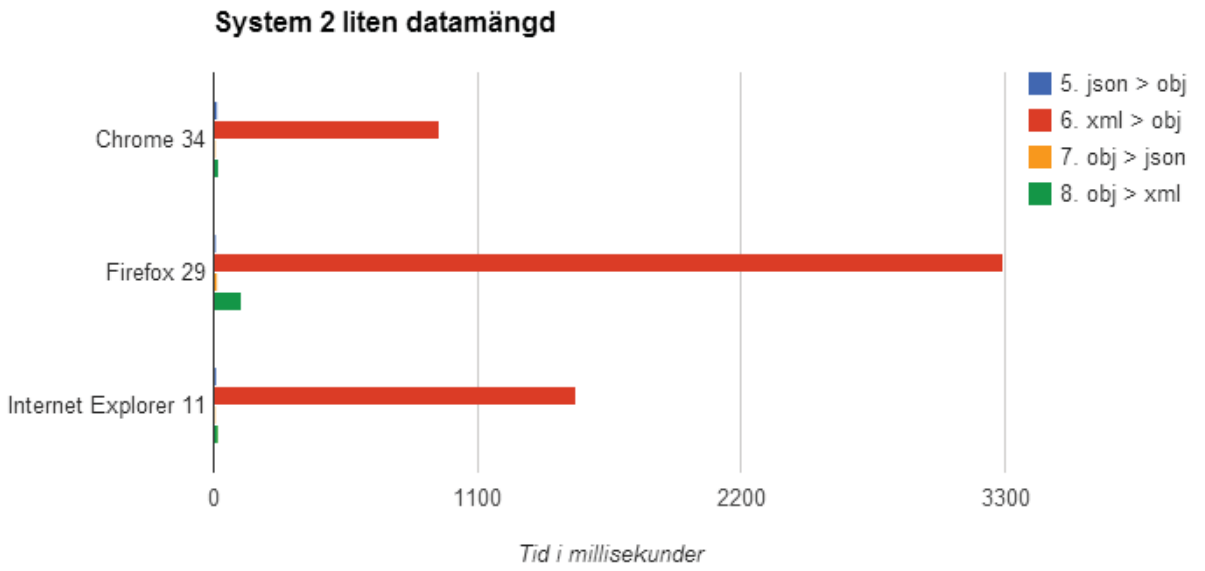
Figur 5.2: Diagram som visar de stora skillnaderna på konverteringstid i test 5-8 på system 1 för JSON och XML med liten datamängd, samt skillnaden mellan olika webbläsare.

Testerna på system 1 visar på stora skillnader i prestanda mellan JSON och XML. Konvertering från XML till objekt visar sig vara nästan 142 gånger långsammare än konvertering från JSON till objekt med den snabbaste webbläsare i testet, Chrome. Den omvända konverteringen, från objekt till JSON/XML visar på betydligt mindre skillnader. Där är XML-varianten endast 4 gånger så långsam.

5.1.2 System 2

ID	Chrome 34	Firefox 29	Internet Explorer 11
5. json > obj	12 ms	7 ms	9 ms
6. xml > obj	942 ms	3 295 ms	1 511 ms
7. obj > json	6 ms	12 ms	6 ms
8. obj > xml	24 ms	115 ms	17 ms

Figur 5.3: Tabell över tiderna det tog att köra test 5-8 på system 2, i de tre valda webbläsarna och med liten datamängd.



Figur 5.4: Diagram som visar de stora skillnaderna på konverteringstid i test 5-8 på system 2 för JSON och XML med liten datamängd, samt skillnaden mellan olika webbläsare.

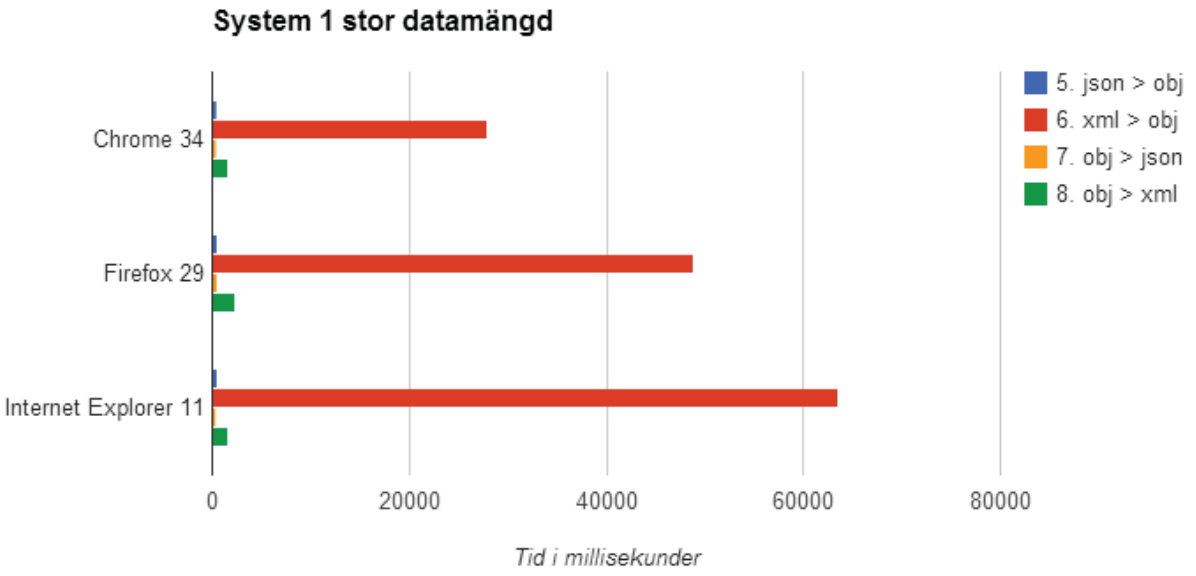
Testerna för system 2 visar liknande resultat som system 1. Den procentuella skillnaden mellan konverting från XML till objekt och JSON till objekt är dock bara ungefär hälften så stor. Detta beror främst på att konvertering från JSON till objekt går så snabbt att även en väldigt liten tidökning kan dubbla den totala tiden det tar för konverteringen.

5.2 Resultat JavaScript stor datamängd

5.2.1 System 1

ID	Chrome 34	Firefox 29	Internet Explorer 11
5. json > obj	381 ms	497 ms	483 ms
6. xml > obj	27 905 ms	48 798 ms	63 608 ms
7. obj > json	372 ms	551 ms	297 ms
8. obj > xml	1 550 ms	2 256 ms	1 627 ms

Figur 5.5: Tabell över tiderna det tog att köra test 5-8 på system 1, i de tre valda webbläsarna och med stor datamängd.



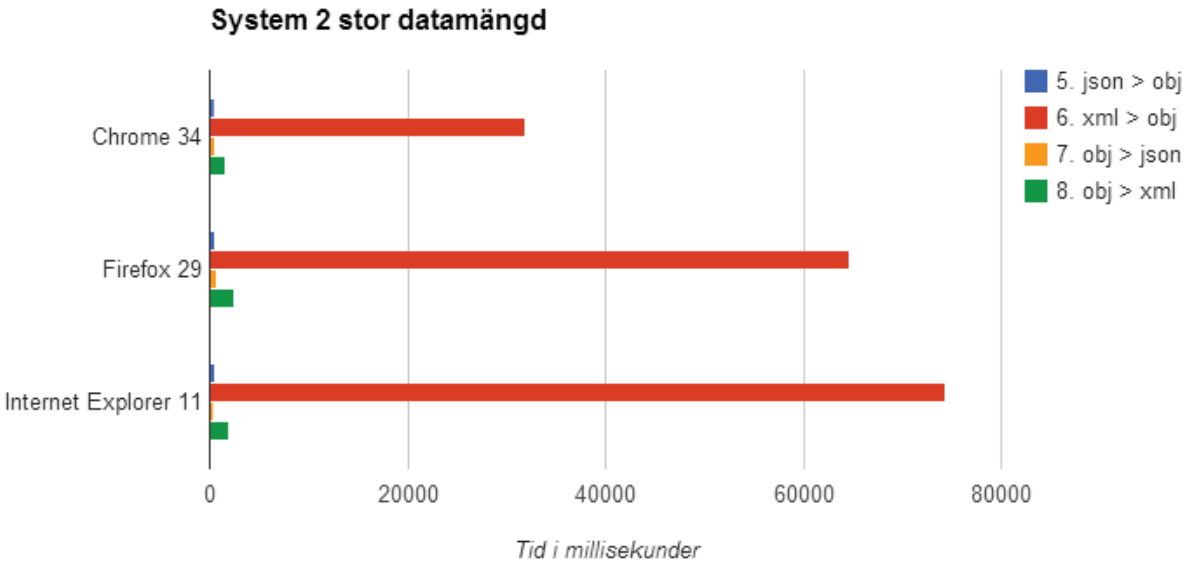
Figur 5.6: Diagram som visar de stora skillnaderna på konverteringstid i test 5-8 på system 1 för JSON och XML med stor datamängd, samt skillnaden mellan olika webbläsare.

Större datamängd ger ingen större skillnad mellan konverteringsmetoderna procentuellt sett. Bortsett från JSON/XML till objekt där JSON är ca 73 gånger snabbare än XML i webbläsaren Chrome för system 1 istället för ca 142 gånger snabbare. Denna skillnaden beror som tidigare nämnts främst på de stora svängningar som kan ske för konvertering JSON till objekt med en liten mängd data. Den omvända konverteringen är drygt 4 gånger långsammare med XML.

5.2.2 System 2

ID	Chrome 34	Firefox 29	Internet Explorer 11
5. json > obj	384 ms	544 ms	521 ms
6. xml > obj	31 957 ms	64 642 ms	74 303 ms
7. obj > json	549 ms	720 ms	320 ms
8. obj > xml	1 605 ms	2 542 ms	1 904 ms

Figur 5.7: Tabell över tiderna det tog att köra test 5-8 på system 2, i de tre valda webbläsarna och med stor datamängd.



Figur 5.8: Diagram som visar de stora skillnaderna på konverteringstid i test 5-8 på system 2 för JSON och XML med stor datamängd, samt skillnaden mellan olika webbläsare.

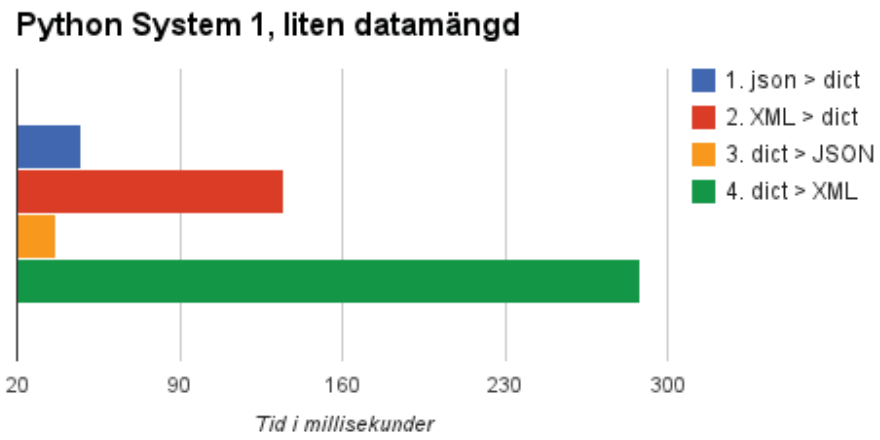
För system 2 märks en liten skillnad när det gäller större datamängder. JSON till objekt har lite större fördel mot XML till objekt än det hade med liten datamängd. Här är XML drygt 83 gånger långsammare. På den omvända konverteringen har dock fördelen för JSON krympt till att endast vara knappt 3 gånger snabbare.

5.3 Resultat Python

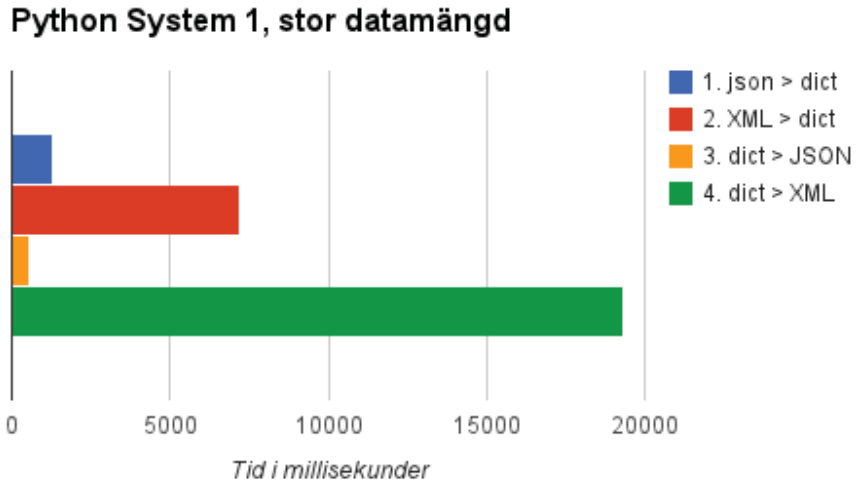
5.3.1 System 1

ID	Liten datamängd	Stor datamängd
1. json > dict	48 ms	1 270 ms
2. xml > dict	135 ms	7 179 ms
3. dict > json	37 ms	547 ms
4. dict > xml	288 ms	19 335 ms

Figur 5.9: Tabell över tiderna det tog att köra test 1-4 på system 1 med både liten och stor datamängd.



Figur 5.10: Diagram som visar skillnaderna på konverteringstid i test 1-4 på system 1 för JSON och XML med liten datamängd.



Figur 5.11: Diagram som visar skillnaderna på konverteringstid i test 1-4 på system 1 för JSON och XML med stor datamängd.

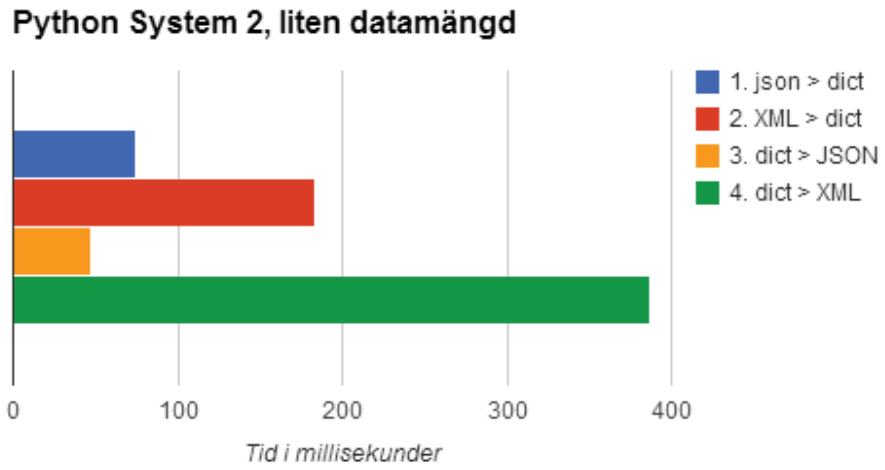
I programmeringsspråket Python är skillnaderna betydligt mindre än de som uppvisades i JavaScript-testerna. I system 1 är konvertering från XML till dictionary drygt 2,8 gånger långsammare än JSON till dictionary med en liten mängd data och knappt 5,7 gånger långsammare med en stor mängd data. Den omvända konverteringen görs knappt 7,8 gånger långsammare med en liten mängd XML-data jämfört med JSON och drygt 35,3 gånger långsammare med en stor mängd XML-data.

5.3.2 System 2

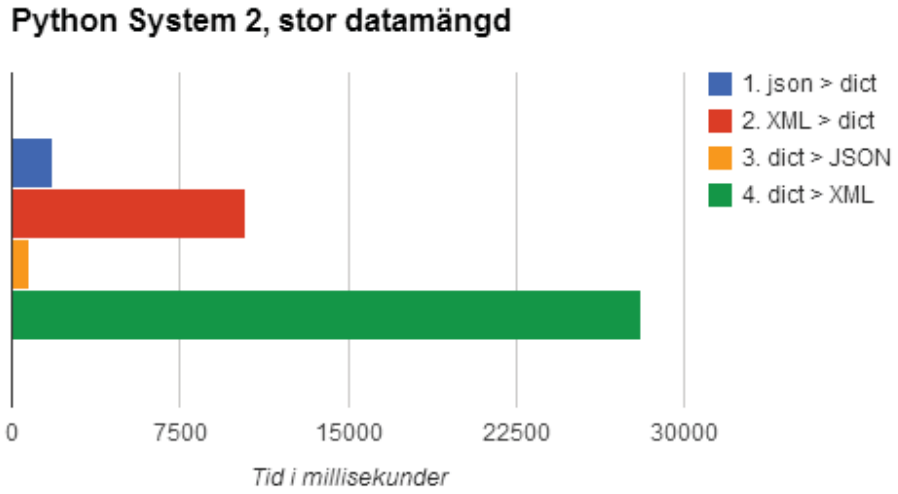
ID	Liten datamängd	Stor datamängd
1. json > dict	74 ms	1809 ms
2. xml > dict	183 ms	10 443 ms
3. dict > json	47 ms	779 ms

4. dict > xml	387 ms	28 065 ms

Figur 5.12: Tabell över tiderna det tog att köra test 1-4 på system 2 med både liten och stor datamängd.



Figur 5.13: Diagram som visar skillnaderna på konverteringstid i test 1-4 på system 2 för JSON och XML med liten datamängd.



Figur 5.14: Diagram som visar skillnaderna på konverteringstid i test 1-4 på system 2 för JSON och XML med stor datamängd.

System 2 visar på liknande resultat som system 1 både när det gäller stor och liten datamängd och konvertering till och från XML/JSON.

6 Analys

Forskningsfrågorna som vi utgått ifrån är följande:

1. Vilket av dataöverföringsformaten XML och JSON är snabbast på att konvertera till och från en bearbetbar datastruktur (dict) i programmeringsspråket Python?
2. Vilket av dataöverföringsformaten XML och JSON är snabbast på att konvertera till och från en bearbetbar datastruktur (object) i programmeringsspråket JavaScript?

Svaret på fråga 1 är att JSON är betydligt snabbare vid konvertering i programmeringsspråket Python. Vid en liten datamängd är skillnaderna inte speciellt stora. XML tog 2,47 till 2,8 gånger så lång tid att konvertera till dictionary i jämförelse med motsvarande JSON. Tidsskillnaden ökar linjärt med datastorleken. XML med en stor mängd data tog ca 5,7 gånger så lång tid att konvertera jämfört med en stor mängd JSON-data. På den omvända konvertering (dictionary till JSON/XML) visar testerna att JSON har ännu större fördel.

Även på fråga två är svaret att JSON är snabbare. Här är dock skillnaderna betydligt större än vad de var med programmeringsspråket Python. Google Chrome var övergripande den webbläsare som utförde konverteringarna snabbast i våra tester.

Figur 5.2, 5.4, 5.6 och 5.8 domineras kraftigt av de röda staplarna. Anledningen är att konverteringen från XML till objekt i JavaScript är väldigt tidskrävande. Där var skillnaden mellan konvertering av JSON och XML med en liten mängd data, en faktor av 78 till 142 till JSONs fördel. Med en stor datamängd förändrades inte faktorn speciellt mycket. XML tog 73 till 83 gånger så lång tid att konvertera jämfört med JSON. Vi testade flera olika bibliotek för att konvertera XML och valde den snabbaste vi kunde hitta. Trots detta är skillnaderna väldigt stora, och JSON är det snabbaste.

Vi tror att JSONs övertag till största delen beror på dess likheter med de datastrukturer vi konverterar till och från. Både Pythons dictionary och JavaScripts objekt är mycket lika JSON i både struktur och uppbyggnad. Skillnaderna är större i JavaScript än Python, vilket stödjer vår teori eftersom JSON är ännu mer likt JavaScript objekt än Python Dictionary.

Jämför man de olika testsystemen som används, ser vi en konstant skillnad i tidsåtgång. Det vill säga, system 2 är ungefär lika mycket långsammare än system 1 på samtliga tester. Baserat på det tror vi inte det skiljer sig nämnvärt på annan hårdvara heller. Serverarkitektur är dock så pass anorlunda och mer avancerad att det rimligtvis skiljer sig, kanske också markant.

De artiklar och bloggar vi läst (redovisade i kapitel 4) har nästan uteslutande kommit fram till samma resultat. Att JSON är det snabbare alternativet. Den tydliga skillnad vi mätt har vi dock inte tidigare kunnat utläsa.

Resultatet från *XML Can Give the Same Performance as JSON*[51] visade att hastigheten var

[51] Jan Stenberg, *XML Can Give the Same Performance as JSON*, hämtad 2014-04-26, <http://www.infoq.com/news/2013/08/xml-json-performance>

detsamma, ett resultat vi inte kunnat replikera. Men då vårt resultat stämmer med den klara majoriteten är vi inte oroliga. Det vore intressant att se hur mätvärden skulle sett ut om vi använde data mer lik den de använde i studien. Det är dock viktigt att inte glömma att studien är ensam, inga andra artiklar visar på samma resultat. Det kan tyda på att den är dåligt genomförd, eller ett verkligt specialfall.

7 Slutsats

Fokuserar man enbart på prestandan är valet enkelt; JSON presterar betydligt bättre än XML. Det spelar ingen roll om man har små dataobjekt eller stora, resultatet står sig väldigt väl. Även om JSON är 70 gånger snabbare på avkodningen för små objekt i JavaScript, bör man inte glömma att ett objekt i XML inte behöver ta mer än 130 millisekunder med en vanlig bärbar dator att avkodas 5000 gånger.

Skall man bygga en ny webbapplikation är det inte säkert det är hastigheten som är flaskhalsen. Det finns andra aspekter som kan vara minst viktiga beroende på det specifika behovet; läsbarhet, biblioteksstöd, minnesanvändning och många fler. Man bör därför undersöka vidare (se kapitel 9) innan man tar ett beslut. Enligt vår subjektiva bedömning är dock hastighet och minnesanvändning det viktigaste. Vi tror inte att minnesanvändningen skiljer sig nämnvärt. Om inte mer tid för undersökning finns, rekommenderar vi att använda JSON.

Våra experiment visar att om hastighet viktas högt, bör man välja JSON över XML. Det gäller både för JavaScript och Python.

Våra resultat visar att JSON även är snabbare i Python med en faktor 2,8 till 5,8. Även i Javascript är JSON det snabbare alternativet, i samtliga av de webbläsare vi testat. Vi tror inte resultatet kommer påverkas nämnvärt i framtida versioner heller. Eftersom JSON blir mer och mer populärt (samtidigt som XML sjunker i popularitet, se figur 2.1) tror vi att mer fokus kommer ligga på att underhålla och optimera för JSON. Vi tror därför det är det mest framtidssäkra formatet.

8. Validitetshot

Eftersom vi använder färdigskrivna bibliotek för att koda/avkoda, finns en given risk att något är felimplementerat, eller i alla fall ineffektivt skrivet. För att hantera den risken använder vi bibliotek som är väl använda och dessutom är öppen källkod. Risken minimeras även genom att vi använder flera olika bibliotek för att testa olika saker, vilket ger en bra genomsnittlig bild över hastigheten. Vi använder dessutom en liten datamängd samt en större, för att undvika scenariot där uppstarten är långsam men själva avkodningen blixtsnabb. En risk är att olika webbläsare har olika effektiva implementationer. Vi använder därför tre olika webbläsare (Chrome, Internet Explorer, Firefox), med deras senaste stabila releaser.

De simpla program vi skrivit i JavaScript och Python är gjorda så att de ska utföra så lite onödigt

arbete som möjligt. Trots att de skrivits med detta som mål är det möjligt att koden inte är helt optimerad och att det på något sätt kan påverka resultaten. Påverkan bör dock vara den samma för samtliga tester vilket minskar allvarlighetsgraden av denna potentiella brist.

9 Framtida arbete

Arbetet försöker svara på frågan vilket format som kan ge högst prestanda. För ett ta reda på vilket format som är *bäst*, krävs det att man evaluerar flera område. Läsbarhet, minnesanvändning, biblioteksstöd och mycket mer. De aspekterna tas inte upp i detta arbete, men vore ett bra komplement.

De båda testsystemen som använts har varit vanliga arbetsstationer. Python används inom webbprogrammering ofta för backend, vilket oftast kommer köras på en dedikerad server. Det vore därför intressant att se om det fanns någon relativ skillnad i hastighet på en mer avancerad hårdvara.

Datan har vi modellerat så realistisk som möjligt, efter vår bästa förmåga. Strukturen är semantisk och liknar det vi tidigare använt på högskola och i industrin. Vår data har medvetet varit vad vi anser standard (se bilagor). Det vore intressant att se hur hastigheten skiljer sig beroende på hur datan ser ut. Ett exempel vore att ha väldigt djupa hierarkier, som på förhand borde gynna XML, eftersom XML i grunden är hierarkisk.

10 Bilagor

A JSON liten datamängd


```
{
        "book": {
                "id": 100,
                "title": "Hitchhiker's guide to the galaxy",
                "author": "Douglas Adams",
                "published": 1979
        }
}
```


B XML liten datamängd


```
<?xml version="1.0" ?>
<book>
   <id>100</id>
   <title>Hitchhiker's guide to the galaxy</title>
   <author>Douglas Adams</author>
   <published>1979</published>
</book>
```


C JSON stor datamängd


```
[
   {
      "id": 0,
      "guid": "951ca0d6-c177-4810-8383-966bb0405c32",
      "isActive": false,
      "balance": "$2,481.00",
      "picture": "http://placehold.it/32x32",
      "age": 32,
      "name": "Robin Barr",
      "gender": "female",
      "company": "CENTREE",
      "email": "robinbarr@centree.com",
      "phone": "+1 (860) 583-3115",
      "address": "132 Bedell Lane, Clarence, Maine, 4454",
      "about": "Ex duis ullamco non anim commodo pariatur esse pariatur tempor irure est excepteur fugiat nostrud. Consequat ullamco dolor incididunt duis labore incididunt commodo aute sit reprehenderit. Velit ullamco commodo ea tempor labore aliquip ad culpa cupidatat magna est nostrud. Enim irure voluptate cupidatat pariatur nulla consequat. Culpa anim veniam sunt consequat duis
```

```
proident velit. In nulla adipisicing anim deserunt irure.",
      "registered": "2014-04-21T16:47:50 -02:00",
      "latitude": 41,
      "longitude": 33,
      "tags": [
         "consectetur",
         "magna",
         "reprehenderit",
         "adipisicing",
         "ullamco",
         "amet",
         "sit"
      ],
      "friends": [
         {
            "id": 0,
            "name": "Rosanna Griffin"
         },
         {
            "id": 1,
            "name": "Dina Macdonald"
         },
         {
            "id": 2,
            "name": "Mccarty Sellers"
         }
      ],
      "greeting": "Hello, Robin Barr! You have 3 unread messages.",
      "favoriteFruit": "apple"
   },
   {
      "id": 1,
      "guid": "1c408f47-fac0-44c5-86e1-dcc911ec3044",
      "isActive": true,
      "balance": "$3,994.00",
      "picture": "http://placehold.it/32x32",
      "age": 21,
      "name": "Alice Pittman",
      "gender": "female",
      "company": "DYMI",
      "email": "alicepittman@dymi.com",
      "phone": "+1 (941) 564-2156",
```

```
    "address": "869 Beacon Court, Glenville, Maryland, 9941",
    "about": "Adipisicing mollit magna labore velit incididunt ad pariatur. Ipsum laboris ea sint nisi officia mollit in. Cupidatat aute laborum nostrud adipisicing pariatur. Aliquip irure nostrud pariatur aliquip ipsum eiusmod sunt elit ex. Quis ex nisi irure quis quis officia velit laboris qui occaecat cupidatat tempor cillum. Reprehenderit ea quis elit consequat. Anim eu proident ut dolor in mollit exercitation do non nostrud dolore.",
    "registered": "2014-04-10T13:44:32 -02:00",
    "latitude": 30,
    "longitude": -133,
    "tags": [
      "ex",
      "ipsum",
      "est",
      "occaecat",
      "tempor",
      "nostrud",
      "est"
    ],
    "friends": [
      {
        "id": 0,
        "name": "Leila Terrell"
      },
      {
        "id": 1,
        "name": "Davis Merritt"
      },
      {
        "id": 2,
        "name": "Rodriquez Goodwin"
      }
    ],
    "greeting": "Hello, Alice Pittman! You have 3 unread messages.",
    "favoriteFruit": "strawberry"
  },
  {
    "id": 2,
    "guid": "f3dcbe03-2b54-4f1b-82cf-f9916d649bd0",
    "isActive": false,
    "balance": "$2,691.00",
    "picture": "http://placehold.it/32x32",
    "age": 35,
```

```
    "name": "Sofia Rice",
    "gender": "female",
    "company": "CIRCUM",
    "email": "sofiarice@circum.com",
    "phone": "+1 (990) 598-3273",
    "address": "117 Garden Street, Salunga, Tennessee, 9199",
    "about": "Fugiat eiusmod ad enim ad ea duis occaecat et cillum aliquip magna duis ad pariatur. Nulla esse minim cupidatat ut do consectetur excepteur sunt reprehenderit in. Qui et dolor sint ut dolore sint aliqua fugiat laboris enim nulla incididunt aliqua ipsum. Velit laborum occaecat tempor elit magna minim occaecat aliquip duis occaecat incididunt eu. Nostrud dolore et aute fugiat esse pariatur Lorem elit id.",
    "registered": "2014-03-20T06:17:06 -01:00",
    "latitude": 84,
    "longitude": 37,
    "tags": [
      "adipisicing",
      "commodo",
      "magna",
      "esse",
      "commodo",
      "tempor",
      "dolor"
    ],
    "friends": [
      {
        "id": 0,
        "name": "Crosby Ryan"
      },
      {
        "id": 1,
        "name": "Frieda Kelly"
      },
      {
        "id": 2,
        "name": "Mooney Ewing"
      }
    ],
    "greeting": "Hello, Sofia Rice! You have 3 unread messages.",
    "favoriteFruit": "banana"
  },
  {
    "id": 3,
```

```
    "guid": "fb8ad549-2090-4f41-8e35-1fa89485118f",
    "isActive": false,
    "balance": "$1,762.00",
    "picture": "http://placehold.it/32x32",
    "age": 25,
    "name": "Alyson Beard",
    "gender": "female",
    "company": "SLAX",
    "email": "alysonbeard@slax.com",
    "phone": "+1 (864) 421-3790",
    "address": "291 Banner Avenue, Garberville, Missouri, 4828",
    "about": "Id anim laboris laboris qui ut aute elit duis id nisi sint minim culpa fugiat. Minim anim adipisicing incididunt cillum do cillum et commodo quis eiusmod nulla. Consequat amet veniam dolor est ut minim proident consequat voluptate tempor enim non laborum.",
    "registered": "2014-01-08T01:42:16 -01:00",
    "latitude": -43,
    "longitude": -60,
    "tags": [
      "consectetur",
      "esse",
      "nisi",
      "veniam",
      "pariatur",
      "sit",
      "cillum"
    ],
    "friends": [
      {
        "id": 0,
        "name": "Kari Everett"
      },
      {
        "id": 1,
        "name": "Petra Hopkins"
      },
      {
        "id": 2,
        "name": "Newton Hart"
      }
    ],
    "greeting": "Hello, Alyson Beard! You have 1 unread messages.",
    "favoriteFruit": "apple"
```

```
  },
  {
    "id": 4,
    "guid": "29293ae8-c79c-49f7-8190-a9701a59f464",
    "isActive": true,
    "balance": "$2,138.00",
    "picture": "http://placehold.it/32x32",
    "age": 35,
    "name": "Vasquez Rivera",
    "gender": "male",
    "company": "STUCCO",
    "email": "vasquezrivera@stucco.com",
    "phone": "+1 (807) 555-2743",
    "address": "384 Prospect Place, Alden, Minnesota, 167",
    "about": "Irure consequat laboris consectetur laboris Lorem. Magna quis veniam deserunt nostrud non amet adipisicing velit sint mollit sint eu voluptate. Nostrud laborum ipsum ut et aute qui velit.",
    "registered": "2014-04-22T09:49:05 -02:00",
    "latitude": 55,
    "longitude": 171,
    "tags": [
      "voluptate",
      "quis",
      "velit",
      "do",
      "culpa",
      "laboris",
      "fugiat"
    ],
    "friends": [
      {
        "id": 0,
        "name": "Osborne Craft"
      },
      {
        "id": 1,
        "name": "Benton Hunt"
      },
      {
        "id": 2,
        "name": "Marisa Hayden"
      }
```

```
    ],
    "greeting": "Hello, Vasquez Rivera! You have 9 unread messages.",
    "favoriteFruit": "strawberry"
  },
  {
    "id": 5,
    "guid": "f7c33396-d90b-4b7e-bee8-addddf265c7b",
    "isActive": false,
    "balance": "$1,672.00",
    "picture": "http://placehold.it/32x32",
    "age": 25,
    "name": "Ilene Fernandez",
    "gender": "female",
    "company": "SEALOUD",
    "email": "ilenefernandez@sealoud.com",
    "phone": "+1 (953) 432-3230",
    "address": "430 Schenck Place, Marshall, New York, 1723",
    "about": "Amet reprehenderit enim mollit sit pariatur proident elit excepteur anim aliqua ut deserunt ut officia. Ullamco est consectetur do sunt eiusmod consequat laboris enim voluptate. Tempor incididunt mollit ea aliqua et ex laborum labore. Eiusmod quis eu ad anim in eu consectetur Lorem non. Dolor ipsum consectetur occaecat amet velit cillum duis magna nisi pariatur fugiat id ipsum.",
    "registered": "2014-04-03T21:37:58 -02:00",
    "latitude": 59,
    "longitude": 47,
    "tags": [
      "reprehenderit",
      "deserunt",
      "sit",
      "enim",
      "labore",
      "eu",
      "cillum"
    ],
    "friends": [
      {
        "id": 0,
        "name": "Reyes Rose"
      },
      {
        "id": 1,
        "name": "Paige Salas"
```

```
      },
      {
        "id": 2,
        "name": "Rebecca Grimes"
      }
    ],
    "greeting": "Hello, Ilene Fernandez! You have 3 unread messages.",
    "favoriteFruit": "apple"
  },
  {
    "id": 6,
    "guid": "514e0f69-9ad3-4cd2-ad35-3e29ebb116b1",
    "isActive": true,
    "balance": "$2,978.00",
    "picture": "http://placehold.it/32x32",
    "age": 39,
    "name": "Roxie Knapp",
    "gender": "female",
    "company": "IMPERIUM",
    "email": "roxieknapp@imperium.com",
    "phone": "+1 (921) 430-3073",
    "address": "894 Farragut Road, Edenburg, Rhode Island, 8324",
    "about": "Proident reprehenderit do qui enim voluptate sit est fugiat ipsum in sit. Amet pariatur duis enim duis reprehenderit enim. Quis occaecat ullamco id deserunt nisi do ullamco qui. Veniam sunt laborum tempor esse veniam commodo elit magna qui excepteur. Reprehenderit consectetur ex pariatur anim quis amet ut eu pariatur nisi aute commodo.",
    "registered": "2014-03-10T08:42:33 -01:00",
    "latitude": 90,
    "longitude": -90,
    "tags": [
      "culpa",
      "minim",
      "labore",
      "culpa",
      "voluptate",
      "eiusmod",
      "do"
    ],
    "friends": [
      {
        "id": 0,
        "name": "Frances Rowland"
```

```
      },
      {
        "id": 1,
        "name": "Keri Clayton"
      },
      {
        "id": 2,
        "name": "Sexton Stokes"
      }
    ],
    "greeting": "Hello, Roxie Knapp! You have 6 unread messages.",
    "favoriteFruit": "apple"
  },
  {
    "id": 7,
    "guid": "9cb9b5f7-e2cc-4937-ba90-9ad8d588ebea",
    "isActive": false,
    "balance": "$1,515.00",
    "picture": "http://placehold.it/32x32",
    "age": 39,
    "name": "Glass Riley",
    "gender": "male",
    "company": "GLUKGLUK",
    "email": "glassriley@glukgluk.com",
    "phone": "+1 (800) 451-2307",
    "address": "888 Hendrix Street, Konterra, Virginia, 2315",
    "about": "Quis dolore occaecat elit excepteur reprehenderit magna adipisicing consectetur occaecat non non. Elit irure consequat et voluptate excepteur qui cillum laboris irure amet non quis. Pariatur nostrud deserunt veniam exercitation laboris ut irure.",
    "registered": "2014-04-07T23:11:43 -02:00",
    "latitude": 59,
    "longitude": 81,
    "tags": [
      "ea",
      "dolore",
      "ipsum",
      "incididunt",
      "ex",
      "do",
      "pariatur"
    ],
    "friends": [
```

```
      {
        "id": 0,
        "name": "Diaz Nguyen"
      },
      {
        "id": 1,
        "name": "Mona Schwartz"
      },
      {
        "id": 2,
        "name": "Frank Gamble"
      }
    ],
    "greeting": "Hello, Glass Riley! You have 6 unread messages.",
    "favoriteFruit": "strawberry"
  },
  {
    "id": 8,
    "guid": "8fefc1a2-5470-40d0-a6f5-c6868a39519d",
    "isActive": false,
    "balance": "$2,882.00",
    "picture": "http://placehold.it/32x32",
    "age": 36,
    "name": "Wyatt Carey",
    "gender": "male",
    "company": "XELEGYL",
    "email": "wyattcarey@xelegyl.com",
    "phone": "+1 (995) 600-3982",
    "address": "630 Schenck Avenue, Sunriver, North Dakota, 3447",
    "about": "Voluptate nisi occaecat sit nulla minim labore qui dolor in exercitation. Ex aliquip cillum pariatur non enim ullamco non anim deserunt excepteur voluptate consequat. Cupidatat irure consequat enim nostrud aliquip occaecat culpa. Nostrud dolor ut eiusmod in ut aute ut proident deserunt culpa ut enim adipisicing. Ipsum officia et veniam eiusmod. In reprehenderit proident non eiusmod quis enim eu anim eiusmod aliqua pariatur aute fugiat. Duis labore nulla dolore nulla occaecat nulla ad magna tempor consectetur occaecat labore.",
    "registered": "2014-04-04T19:43:21 -02:00",
    "latitude": 88,
    "longitude": -137,
    "tags": [
      "veniam",
      "cupidatat",
      "anim",
```

```
            "et",
            "veniam",
            "exercitation",
            "incididunt"
        ],
        "friends": [
            {
                "id": 0,
                "name": "Alvarado Bird"
            },
            {
                "id": 1,
                "name": "Harriett Richmond"
            },
            {
                "id": 2,
                "name": "Little Sims"
            }
        ],
        "greeting": "Hello, Wyatt Carey! You have 7 unread messages.",
        "favoriteFruit": "banana"
    },
    {
        "id": 9,
        "guid": "49bc4254-cbab-4fdb-af17-3666ad891074",
        "isActive": false,
        "balance": "$2,848.00",
        "picture": "http://placehold.it/32x32",
        "age": 20,
        "name": "Fern Valencia",
        "gender": "female",
        "company": "ZAJ",
        "email": "fernvalencia@zaj.com",
        "phone": "+1 (860) 427-3384",
        "address": "211 Middagh Street, Shaft, Texas, 509",
        "about": "Irure officia sit irure exercitation commodo nulla reprehenderit velit nisi. Sint consequat duis duis et non quis esse reprehenderit nisi ipsum ad deserunt. Commodo laborum id occaecat exercitation cillum incididunt excepteur aute commodo veniam. Lorem magna in id minim dolor. Qui officia eu qui et deserunt aliqua consectetur laborum aliquip mollit deserunt ea aliqua. Dolor esse culpa commodo id esse eu et esse cupidatat elit.",
        "registered": "2014-01-17T09:10:36 -01:00",
        "latitude": -83,
```

```
    "longitude": -170,
    "tags": [
      "fugiat",
      "aliqua",
      "consequat",
      "velit",
      "labore",
      "eiusmod",
      "ad"
    ],
    "friends": [
      {
        "id": 0,
        "name": "Lorrie Lott"
      },
      {
        "id": 1,
        "name": "Greene Mullins"
      },
      {
        "id": 2,
        "name": "Webb Compton"
      }
    ],
    "greeting": "Hello, Fern Valencia! You have 6 unread messages.",
    "favoriteFruit": "strawberry"
  }
]
```


D XML stor datamängd


```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <element>
    <about>Ex duis ullamco non anim commodo pariatur esse pariatur tempor irure est excepteur fugiat nostrud. Consequat ullamco dolor incididunt duis labore incididunt commodo aute sit reprehenderit. Velit ullamco commodo ea tempor labore aliquip ad culpa cupidatat magna est nostrud. Enim irure voluptate cupidatat pariatur nulla consequat. Culpa anim veniam sunt consequat duis proident velit. In nulla adipisicing anim deserunt irure.</about>
    <address>132 Bedell Lane, Clarence, Maine, 4454</address>
    <age>32</age>
    <balance>$2,481.00</balance>
```

```
    <company>CENTREE</company>
    <email>robinbarr@centree.com</email>
    <favoriteFruit>apple</favoriteFruit>
    <friends>
      <element>
        <id>0</id>
        <name>Rosanna Griffin</name>
      </element>
      <element>
        <id>1</id>
        <name>Dina Macdonald</name>
      </element>
      <element>
        <id>2</id>
        <name>Mccarty Sellers</name>
      </element>
    </friends>
    <gender>female</gender>
    <greeting>Hello, Robin Barr! You have 3 unread messages.</greeting>
    <guid>951ca0d6-c177-4810-8383-966bb0405c32</guid>
    <id>0</id>
    <isActive>false</isActive>
    <latitude>41</latitude>
    <longitude>33</longitude>
    <name>Robin Barr</name>
    <phone>+1 (860) 583-3115</phone>
    <picture>http://placehold.it/32x32</picture>
    <registered>2014-04-21T16:47:50 -02:00</registered>
    <tags>
      <element>consectetur</element>
      <element>magna</element>
      <element>reprehenderit</element>
      <element>adipisicing</element>
      <element>ullamco</element>
      <element>amet</element>
      <element>sit</element>
    </tags>
  </element>
  <element>
    <about>Adipisicing mollit magna labore velit incididunt ad pariatur. Ipsum laboris ea sint nisi officia mollit in. Cupidatat aute laborum nostrud adipisicing pariatur. Aliquip irure nostrud pariatur aliquip ipsum eiusmod sunt elit ex. Quis ex nisi irure quis quis officia velit laboris qui occaecat
```

cupidatat tempor cillum. Reprehenderit ea quis elit consequat. Anim eu proident ut dolor in mollit exercitation do non nostrud dolore.</about>
<address>869 Beacon Court, Glenville, Maryland, 9941</address>
<age>21</age>
<balance>$3,994.00</balance>
<company>DYMI</company>
<email>alicepittman@dymi.com</email>
<favoriteFruit>strawberry</favoriteFruit>
<friends>
<element>
<id>0</id>
<name>Leila Terrell</name>
</element>
<element>
<id>1</id>
<name>Davis Merritt</name>
</element>
<element>
<id>2</id>
<name>Rodriquez Goodwin</name>
</element>
</friends>
<gender>female</gender>
<greeting>Hello, Alice Pittman! You have 3 unread messages.</greeting>
<guid>1c408f47-fac0-44c5-86e1-dcc911ec3044</guid>
<id>1</id>
<isActive>true</isActive>
<latitude>30</latitude>
<longitude>-133</longitude>
<name>Alice Pittman</name>
<phone>+1 (941) 564-2156</phone>
<picture>http://placehold.it/32x32</picture>
<registered>2014-04-10T13:44:32 -02:00</registered>
<tags>
<element>ex</element>
<element>ipsum</element>
<element>est</element>
<element>occaecat</element>
<element>tempor</element>
<element>nostrud</element>
<element>est</element>
</tags>

```
  </element>
  <element>
    <about>Fugiat eiusmod ad enim ad ea duis occaecat et cillum aliquip magna duis ad pariatur. Nulla esse minim cupidatat ut do consectetur excepteur sunt reprehenderit in. Qui et dolor sint ut dolore sint aliqua fugiat laboris enim nulla incididunt aliqua ipsum. Velit laborum occaecat tempor elit magna minim occaecat aliquip duis occaecat incididunt eu. Nostrud dolore et aute fugiat esse pariatur Lorem elit id.</about>
    <address>117 Garden Street, Salunga, Tennessee, 9199</address>
    <age>35</age>
    <balance>$2,691.00</balance>
    <company>CIRCUM</company>
    <email>sofiarice@circum.com</email>
    <favoriteFruit>banana</favoriteFruit>
    <friends>
      <element>
        <id>0</id>
        <name>Crosby Ryan</name>
      </element>
      <element>
        <id>1</id>
        <name>Frieda Kelly</name>
      </element>
      <element>
        <id>2</id>
        <name>Mooney Ewing</name>
      </element>
    </friends>
    <gender>female</gender>
    <greeting>Hello, Sofia Rice! You have 3 unread messages.</greeting>
    <guid>f3dcbe03-2b54-4f1b-82cf-f9916d649bd0</guid>
    <id>2</id>
    <isActive>false</isActive>
    <latitude>84</latitude>
    <longitude>37</longitude>
    <name>Sofia Rice</name>
    <phone>+1 (990) 598-3273</phone>
    <picture>http://placehold.it/32x32</picture>
    <registered>2014-03-20T06:17:06 -01:00</registered>
    <tags>
      <element>adipisicing</element>
      <element>commodo</element>
      <element>magna</element>
```

```
        <element>esse</element>
        <element>commodo</element>
        <element>tempor</element>
        <element>dolor</element>
      </tags>
    </element>
    <element>
      <about>Id anim laboris laboris qui ut aute elit duis id nisi sint minim culpa fugiat. Minim anim adipisicing incididunt cillum do cillum et commodo quis eiusmod nulla. Consequat amet veniam dolor est ut minim proident consequat voluptate tempor enim non laborum.</about>
      <address>291 Banner Avenue, Garberville, Missouri, 4828</address>
      <age>25</age>
      <balance>$1,762.00</balance>
      <company>SLAX</company>
      <email>alysonbeard@slax.com</email>
      <favoriteFruit>apple</favoriteFruit>
      <friends>
        <element>
          <id>0</id>
          <name>Kari Everett</name>
        </element>
        <element>
          <id>1</id>
          <name>Petra Hopkins</name>
        </element>
        <element>
          <id>2</id>
          <name>Newton Hart</name>
        </element>
      </friends>
      <gender>female</gender>
      <greeting>Hello, Alyson Beard! You have 1 unread messages.</greeting>
      <guid>fb8ad549-2090-4f41-8e35-1fa89485118f</guid>
      <id>3</id>
      <isActive>false</isActive>
      <latitude>-43</latitude>
      <longitude>-60</longitude>
      <name>Alyson Beard</name>
      <phone>+1 (864) 421-3790</phone>
      <picture>http://placehold.it/32x32</picture>
      <registered>2014-01-08T01:42:16 -01:00</registered>
      <tags>
```

```
        <element>consectetur</element>
        <element>esse</element>
        <element>nisi</element>
        <element>veniam</element>
        <element>pariatur</element>
        <element>sit</element>
        <element>cillum</element>
      </tags>
    </element>
    <element>
      <about>Irure consequat laboris consectetur laboris Lorem. Magna quis veniam deserunt nostrud non amet adipisicing velit sint mollit sint eu voluptate. Nostrud laborum ipsum ut et aute qui velit.</about>
      <address>384 Prospect Place, Alden, Minnesota, 167</address>
      <age>35</age>
      <balance>$2,138.00</balance>
      <company>STUCCO</company>
      <email>vasquezrivera@stucco.com</email>
      <favoriteFruit>strawberry</favoriteFruit>
      <friends>
        <element>
          <id>0</id>
          <name>Osborne Craft</name>
        </element>
        <element>
          <id>1</id>
          <name>Benton Hunt</name>
        </element>
        <element>
          <id>2</id>
          <name>Marisa Hayden</name>
        </element>
      </friends>
      <gender>male</gender>
      <greeting>Hello, Vasquez Rivera! You have 9 unread messages.</greeting>
      <guid>29293ae8-c79c-49f7-8190-a9701a59f464</guid>
      <id>4</id>
      <isActive>true</isActive>
      <latitude>55</latitude>
      <longitude>171</longitude>
      <name>Vasquez Rivera</name>
      <phone>+1 (807) 555-2743</phone>
```

```
    <picture>http://placehold.it/32x32</picture>
    <registered>2014-04-22T09:49:05 -02:00</registered>
    <tags>
      <element>voluptate</element>
      <element>quis</element>
      <element>velit</element>
      <element>do</element>
      <element>culpa</element>
      <element>laboris</element>
      <element>fugiat</element>
    </tags>
  </element>
  <element>
    <about>Amet reprehenderit enim mollit sit pariatur proident elit excepteur anim aliqua ut deserunt ut officia. Ullamco est consectetur do sunt eiusmod consequat laboris enim voluptate. Tempor incididunt mollit ea aliqua et ex laborum labore. Eiusmod quis eu ad anim in eu consectetur Lorem non. Dolor ipsum consectetur occaecat amet velit cillum duis magna nisi pariatur fugiat id ipsum.</about>
    <address>430 Schenck Place, Marshall, New York, 1723</address>
    <age>25</age>
    <balance>$1,672.00</balance>
    <company>SEALOUD</company>
    <email>ilenefernandez@sealoud.com</email>
    <favoriteFruit>apple</favoriteFruit>
    <friends>
      <element>
        <id>0</id>
        <name>Reyes Rose</name>
      </element>
      <element>
        <id>1</id>
        <name>Paige Salas</name>
      </element>
      <element>
        <id>2</id>
        <name>Rebecca Grimes</name>
      </element>
    </friends>
    <gender>female</gender>
    <greeting>Hello, Ilene Fernandez! You have 3 unread messages.</greeting>
    <guid>f7c33396-d90b-4b7e-bee8-addddf265c7b</guid>
    <id>5</id>
```

```
    <isActive>false</isActive>
    <latitude>59</latitude>
    <longitude>47</longitude>
    <name>Ilene Fernandez</name>
    <phone>+1 (953) 432-3230</phone>
    <picture>http://placehold.it/32x32</picture>
    <registered>2014-04-03T21:37:58 -02:00</registered>
    <tags>
      <element>reprehenderit</element>
      <element>deserunt</element>
      <element>sit</element>
      <element>enim</element>
      <element>labore</element>
      <element>eu</element>
      <element>cillum</element>
    </tags>
  </element>
  <element>
    <about>Proident reprehenderit do qui enim voluptate sit est fugiat ipsum in sit. Amet pariatur duis enim duis reprehenderit enim. Quis occaecat ullamco id deserunt nisi do ullamco qui. Veniam sunt laborum tempor esse veniam commodo elit magna qui excepteur. Reprehenderit consectetur ex pariatur anim quis amet ut eu pariatur nisi aute commodo.</about>
    <address>894 Farragut Road, Edenburg, Rhode Island, 8324</address>
    <age>39</age>
    <balance>$2,978.00</balance>
    <company>IMPERIUM</company>
    <email>roxieknapp@imperium.com</email>
    <favoriteFruit>apple</favoriteFruit>
    <friends>
      <element>
        <id>0</id>
        <name>Frances Rowland</name>
      </element>
      <element>
        <id>1</id>
        <name>Keri Clayton</name>
      </element>
      <element>
        <id>2</id>
        <name>Sexton Stokes</name>
      </element>
    </friends>
```

```
    <gender>female</gender>
    <greeting>Hello, Roxie Knapp! You have 6 unread messages.</greeting>
    <guid>514e0f69-9ad3-4cd2-ad35-3e29ebb116b1</guid>
    <id>6</id>
    <isActive>true</isActive>
    <latitude>90</latitude>
    <longitude>-90</longitude>
    <name>Roxie Knapp</name>
    <phone>+1 (921) 430-3073</phone>
    <picture>http://placehold.it/32x32</picture>
    <registered>2014-03-10T08:42:33 -01:00</registered>
    <tags>
      <element>culpa</element>
      <element>minim</element>
      <element>labore</element>
      <element>culpa</element>
      <element>voluptate</element>
      <element>eiusmod</element>
      <element>do</element>
    </tags>
  </element>
  <element>
    <about>Quis dolore occaecat elit excepteur reprehenderit magna adipisicing consectetur occaecat non non. Elit irure consequat et voluptate excepteur qui cillum laboris irure amet non quis. Pariatur nostrud deserunt veniam exercitation laboris ut irure.</about>
    <address>888 Hendrix Street, Konterra, Virginia, 2315</address>
    <age>39</age>
    <balance>$1,515.00</balance>
    <company>GLUKGLUK</company>
    <email>glassriley@glukgluk.com</email>
    <favoriteFruit>strawberry</favoriteFruit>
    <friends>
      <element>
        <id>0</id>
        <name>Diaz Nguyen</name>
      </element>
      <element>
        <id>1</id>
        <name>Mona Schwartz</name>
      </element>
      <element>
        <id>2</id>
```

```
            <name>Frank Gamble</name>
          </element>
        </friends>
        <gender>male</gender>
        <greeting>Hello, Glass Riley! You have 6 unread messages.</greeting>
        <guid>9cb9b5f7-e2cc-4937-ba90-9ad8d588ebea</guid>
        <id>7</id>
        <isActive>false</isActive>
        <latitude>59</latitude>
        <longitude>81</longitude>
        <name>Glass Riley</name>
        <phone>+1 (800) 451-2307</phone>
        <picture>http://placehold.it/32x32</picture>
        <registered>2014-04-07T23:11:43 -02:00</registered>
        <tags>
          <element>ea</element>
          <element>dolore</element>
          <element>ipsum</element>
          <element>incididunt</element>
          <element>ex</element>
          <element>do</element>
          <element>pariatur</element>
        </tags>
      </element>
      <element>
        <about>Voluptate nisi occaecat sit nulla minim labore qui dolor in exercitation. Ex aliquip cillum pariatur non enim ullamco non anim deserunt excepteur voluptate consequat. Cupidatat irure consequat enim nostrud aliquip occaecat culpa. Nostrud dolor ut eiusmod in ut aute ut proident deserunt culpa ut enim adipisicing. Ipsum officia et veniam eiusmod. In reprehenderit proident non eiusmod quis enim eu anim eiusmod aliqua pariatur aute fugiat. Duis labore nulla dolore nulla occaecat nulla ad magna tempor consectetur occaecat labore.</about>
        <address>630 Schenck Avenue, Sunriver, North Dakota, 3447</address>
        <age>36</age>
        <balance>$2,882.00</balance>
        <company>XELEGYL</company>
        <email>wyattcarey@xelegyl.com</email>
        <favoriteFruit>banana</favoriteFruit>
        <friends>
          <element>
            <id>0</id>
            <name>Alvarado Bird</name>
          </element>
```

```
      <element>
        <id>1</id>
        <name>Harriett Richmond</name>
      </element>
      <element>
        <id>2</id>
        <name>Little Sims</name>
      </element>
    </friends>
    <gender>male</gender>
    <greeting>Hello, Wyatt Carey! You have 7 unread messages.</greeting>
    <guid>8fefc1a2-5470-40d0-a6f5-c6868a39519d</guid>
    <id>8</id>
    <isActive>false</isActive>
    <latitude>88</latitude>
    <longitude>-137</longitude>
    <name>Wyatt Carey</name>
    <phone>+1 (995) 600-3982</phone>
    <picture>http://placehold.it/32x32</picture>
    <registered>2014-04-04T19:43:21 -02:00</registered>
    <tags>
      <element>veniam</element>
      <element>cupidatat</element>
      <element>anim</element>
      <element>et</element>
      <element>veniam</element>
      <element>exercitation</element>
      <element>incididunt</element>
    </tags>
  </element>
  <element>
    <about>Irure officia sit irure exercitation commodo nulla reprehenderit velit nisi. Sint consequat duis duis et non quis esse reprehenderit nisi ipsum ad deserunt. Commodo laborum id occaecat exercitation cillum incididunt excepteur aute commodo veniam. Lorem magna in id minim dolor. Qui officia eu qui et deserunt aliqua consectetur laborum aliquip mollit deserunt ea aliqua. Dolor esse culpa commodo id esse eu et esse cupidatat elit.</about>
    <address>211 Middagh Street, Shaft, Texas, 509</address>
    <age>20</age>
    <balance>$2,848.00</balance>
    <company>ZAJ</company>
    <email>fernvalencia@zaj.com</email>
    <favoriteFruit>strawberry</favoriteFruit>
```

```
      <friends>
        <element>
          <id>0</id>
          <name>Lorrie Lott</name>
        </element>
        <element>
          <id>1</id>
          <name>Greene Mullins</name>
        </element>
        <element>
          <id>2</id>
          <name>Webb Compton</name>
        </element>
      </friends>
      <gender>female</gender>
      <greeting>Hello, Fern Valencia! You have 6 unread messages.</greeting>
      <guid>49bc4254-cbab-4fdb-af17-3666ad891074</guid>
      <id>9</id>
      <isActive>false</isActive>
      <latitude>-83</latitude>
      <longitude>-170</longitude>
      <name>Fern Valencia</name>
      <phone>+1 (860) 427-3384</phone>
      <picture>http://placehold.it/32x32</picture>
      <registered>2014-01-17T09:10:36 -01:00</registered>
      <tags>
        <element>fugiat</element>
        <element>aliqua</element>
        <element>consequat</element>
        <element>velit</element>
        <element>labore</element>
        <element>eiusmod</element>
        <element>ad</element>
      </tags>
    </element>
</root>
```

E Python

Notera rad 12-16. Vi har ersatt textsträngarna med en referens till tidigare bilagor.

```
1.  import timeit
```

```
2. import dict2xml
3. import xmldict
4. import time
5. import json
6. import xml.etree.cElementTree as ElementTree
7. import CTree
8. import etree_to_dict
9.
10. NUMBER_OF_TEST_RUNS = 5000
11.
12. XML = <Bilaga B>
13. JSON = <Bilaga A>
14.
15. JSON_LARGE = <Bilaga C>
16. XML_LARGE = <Bilaga D>
17.
18. def load_xmldict_str(s):
19.     return xmldict.xml_to_dict(s)
20.
21.
22. def load_json_file(path):
23.     return json.load(open(path))
24.
25.
26. def load_json_str(s):
27.     return json.loads(s)
28.
29.
30. def dump_json_str(d):
31.     return json.dumps(d)
32.
33.
34. def load_xmlctree_file(path):
```

```
35.     tree = ElementTree.parse(path)
36.     root = tree.getroot()
37.     return CTree.XmlDictConfig(root)
38. 
39. 
40. def load_xmlctree_str(s):
41.     tree = ElementTree.ElementTree(ElementTree.fromstring(s))
42.     root = tree.getroot()
43.     return dict(CTree.XmlDictConfig(root))
44. 
45. 
46. def xml_to_dict(xml_string):
47.     root = ElementTree.XML(xml_string)
48.     return dict(etree_to_dict.etree_to_dict(root))
49. 
50. 
51. def dict_to_xml(d):
52.     return dict2xml.dict2xml(d)
53. 
54. 
55. def test_timeit(function_name, argument, text, number):
56.     statement = function_name + '(' + argument + ')'
57.     print text, timeit.timeit(statement, 'from __main__ import ' + function_name + ', '
    + argument, number=number)
58. 
59. 
60. json_dict = load_json_str(JSON)
61. json_dict_large = load_json_str(JSON_LARGE)
62. xml_dict = xml_to_dict(XML)
63. xml_dict_large = xml_to_dict(XML_LARGE)
64. 
65. print '----- SMALL DATA -----'
66. test_timeit('load_json_str', 'JSON', 'json to dict', NUMBER_OF_TEST_RUNS)
```

```
67. test_timeit('xml_to_dict', 'XML', 'xml to dict', NUMBER_OF_TEST_RUNS)
68. test_timeit('dump_json_str', 'json_dict', 'dict to json', NUMBER_OF_TEST_RUNS)
69. test_timeit('dict_to_xml', 'xml_dict', 'dict to XML', NUMBER_OF_TEST_RUNS)
70.
71.
72. print '----- LARGE DATA -----'
73. test_timeit('load_json_str', 'JSON_LARGE', 'json to dict', NUMBER_OF_TEST_RUNS)
74. test_timeit('xml_to_dict', 'XML_LARGE', 'xml to dict', NUMBER_OF_TEST_RUNS)
75. test_timeit('dump_json_str', 'json_dict_large', 'dict to json', NUMBER_OF_TEST_RUNS)
76. test_timeit('dict_to_xml', 'xml_dict_large', 'dict to XML', NUMBER_OF_TEST_RUNS)
```

F Javascript

Notera rad 15-21. Vi har ersatt textsträngarna med en referens till tidigare bilagor.

```
1.  <!-- http://www.thomasfrank.se/xml_to_json.html very slow-->
2.  <script src="xml2json.js"></script>
3.
4.  <!-- http://goessner.net/download/prj/jsonxml/ -->
5.  <script src="json2xml.js"></script>
6.
7.  <!-- https://code.google.com/p/x2js/ -->
8.  <script src="xml2json.min.js"></script>
9.  <script>
10. (function () {
11. 'use strict';
12.
13. var NUMBER_OF_TEST_RUNS = 5000;
14.
15. var JSON_STRING = <Bilaga A>;
16.
17. var XML_STRING = <Bilaga B>;
18.
19. var JSON_STRING_VERY_LARGE = <Bilaga C>;
20.
```

```
21. var XML_STRING_VERY_LARGE = <Bilaga D>;
22.
23. var x2js = new X2JS();
24.
25. function convertJsonToObj(jsonString) {
26.     return JSON.parse(jsonString);
27. }
28.
29. function convertXMLToObj(xmlString) {
30.     return xml2json.parser(xmlString);
31. }
32.
33. function convertXMLToObjX2JS(xmlString) {
34.     return x2js.xml_str2json(xmlString);
35. }
36.
37. function convertObjToJson(obj) {
38.     return JSON.stringify(obj);
39. }
40.
41. function convertObjToXML(obj) {
42.     return json2xml(obj, '');
43. }
44.
45. function convertObjToXMLX2JS(obj) {
46.     return x2js.json2xml_str(obj);
47. }
48.
49. function test(func, argument, timerName) {
50.     var i = 0;
51.     console.time(timerName);
52.     for(i = 0; i < NUMBER_OF_TEST_RUNS; i++) {
53.         func(argument);
```

```
54.   }
55.   console.timeEnd(timerName);
56. }
57.
58. var jsonObject = convertJsonToObj(JSON_STRING);
59. var jsonObjectLarge = convertJsonToObj(JSON_STRING_VERY_LARGE);
60. var xmlObject = convertXMLToObjX2JS(XML_STRING);
61. var xmlObjectLarge = convertXMLToObjX2JS(XML_STRING_VERY_LARGE);
62.
63. console.log('-----SMALL DATA-----');
64. test(convertJsonToObj, JSON_STRING, 'json to obj');
65. test(convertXMLToObjX2JS, XML_STRING, 'xml to obj x2js');
66. test(convertObjToJson, jsonObject, 'obj to json');
67. test(convertObjToXML, xmlObject, 'obj to xml');
68.
69. console.log('-----LARGE DATA-----');
70. test(convertJsonToObj, JSON_STRING_VERY_LARGE, 'json to obj');
71. test(convertXMLToObjX2JS, XML_STRING_VERY_LARGE, 'xml to obj x2js');
72. test(convertObjToJson, jsonObjectLarge, 'obj to json');
73. test(convertObjToXML, xmlObjectLarge, 'obj to xml');
74.
75. console.log('-----SMALL OUTPUTS-----');
76. console.log('jsonToObj output', jsonObject);
77. console.log('xmlToObjX2JS output', xmlObject);
78. console.log('objToJson output', convertObjToJson(jsonObject));
79. console.log('objToXML output', convertObjToXML(xmlObject));
80.
81. console.log('-----LARGE OUTPUTS-----');
82. console.log('jsonToObj large output', jsonObjectLarge);
83. console.log('xmlToObjX2JS large output', xmlObjectLarge);
84.
85. }());
86. </script>
```