Comparison of JSON and XML Data Formats

Alen Šimec, Magdalena Magličić
Department of IT technologies
Polytechnic of Zagreb
Konavoska 2, 10000 Zagreb, Croatia
{alen, mmaglicic}@tvz.hr

Abstract. *Choosing an appropriate data format for your data makes a difference in programming and performance speed of your application. In this paper, we'll compare two of the most popular data formats for web applications, JSON and XML, in terms of features, use, and performance testing results.*

Keywords. JSON, XML, data formats, comparison

1 Introduction

When it emerged as a W3C standard more than a decade ago, XML was viewed as a revolutionary markup language for data transport and storage. It was designed as a software and hardware-independent tool, which was convenient for the developer community at that time, in the midst of the "Internet boom". XML became a widely accepted language and gained acceptance into all programming fields, but it mostly affected the way the Internet we know today works. [7]

Over the years, web sites have shifted from being static to very dynamic and interactive. Interactivity increased data transfer and developers looked for a way to modify a small section of a web site without retransmitting all of the data. Ajax (Asynchronous JavaScript and XML) is a group of technologies that together account for asynchronous communication with the server without having to refresh the page. In the original specification Ajax was designed to work with the following technologies: HTML/XHTML and CSS for presentation, DOM to dynamically display and interact with data, XML for data exchange (and XSLT to manipulate the XML), XMLHttpRequest object for asynchronous communication and JavaScript as the cohesive technology.

For Ajax, the name itself suggests that it was built for XML, but there was the possibility of using other encoding. XML had it's limitations and did not prove to be the ideal transportation tool so new data formats emerged, such as JSON (JavaScript Object Notation). Although JSON is based on the JavaScript language, it's language independent,as it uses a simple key-value pair notation. Today, JSON is widely used as an alternative to XML in the field of web development.

2 Feature comparison

2.1 Code and data model [10]

```
<product>
    <id>15</id>
    <name>Widgets</name>
    <description>These widgets are the
finest widgets ever made by
anyone.</description>
    <options type="color">
         <item>Purple </item>
         <item>Green </item>
         <item>Orange </item>
    </options>
</product>

"product" : {
    "id" : 15,
    "name" : "Widgets",
    "description" : "These widgets are
the finest widgets ever made by
anyone.",
    "options" : [
         {
              "type" : "color",
              "items" : [
                   "Purple",
                   "Green",
                   "Orange"
              ]
         }
      ]
}
```

Figure 1. Code examples (XML vs. JSON)

An XML document forms a branched structure that starts with the root element. Every XML document must be "well-formed", i.e. be in accordance with strict rules, without which it is not valid. Some of the XML technologies used for manipulating and formatting XML documents are XML Namespaces, XML Schema, XSLT, and XPath. All of them are standardized and accepted by the W3C association. The greatest strength of this model is the strict

structure definition, which is why it can be implemented for data validation.

On the other hand, the XML document is, for many object-oriented languages, difficult to parse and convert into a structure or object with which they can work. Still, there are many technologies for processing XML: Document Object Model, XPath and XSLT. The use of closing tags (<id>15</id>versus JSON's"id":15) make XML too comprehensive and partially redundant, which ultimately means a longer document reading time.

JSON draws its data model from JavaScript: the data is presented as a key-value pair, with curly braces representing objects and angular braces fields. The syntax is simpler than XML's and OOP languages can easily translate the JSON string into an object. It is natively supported by JavaScript. Both languages use the Unicode standard.

JSON, because of its simple structure, is not suitable for data validation (as opposed to XML). It cannot store all data types. However there is an Internet draft called JSON Schema, which may play a role in JSON data validation. [7]

2.2 Accessing and extracting data

2.2.1 XML and XPath

XML is a text-based way to represent documents, but once an XML document has been read into memory, it's usually represented as a tree. To make developers' lives easier, several standard ways exist to represent and access that tree. [1]

The best-known data model for storing and processing XML trees is called the W3C document object model, or the DOM for short. JQuery and other similar libraries are built on top of the DOM and described in terms of the DOM. XPath 2 and 3, XQuery, and XSLT 2 all use the XQuery and XPath Data Model, or XDM, which is a slightly different data model than DOM. The XDM is more powerful than the DOM, includes support for objects with types described by W3C XML Schema, and also supports items such as floating-point numbers and sequences intermingled with the XML data.

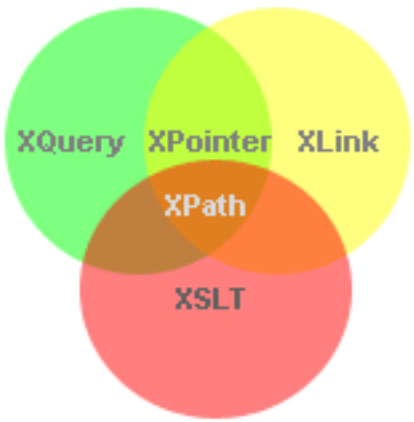

Figure 2. Connection between XPath and other XML technologies

A frequently emphasized advantage of XML is the availability of plenty tools to analyse, transform and selectively extract data out of XML documents. XPath (the XML Path Language) is one of these powerful tools. XPath is used to point into XML documents and select parts of them for later use. It was designed to be embedded and used by other languages — in particular by XSLT, XLink, and XPointer, and later by XQuery — and this means that XPath isn't a complete programming language, but can be used in host of languages (Python, PHP, Perl, C, C++, JavaScript etc.).

The most common way to use XPath is to pass an XPath expression and one or more XML documents to an XPath engine: the XPath engine evaluates the expression and gives back the result. This can be via a programming language API, using a standalone command-line program, or indirectly, as when XPath is included in another language such as XQuery or XSLT. [1] In XPath, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes. XML documents are treated as trees of nodes. The topmost element of the tree is called the root element. XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by following a path or steps. [9]

In Javascript, the main interface to using XPath is the evaluate function of the document object. This method evaluates XPath expressions against an XML based document (including HTML documents), and returns an XPathResult object, which can be a single node or a set of nodes (see Figure 3.).

```
var xpathResult = document.evaluate(
xpathExpression, contextNode,
namespaceResolver, resultType, result );
```

Figure 3. XPath and document.evaluate

An example of accessing the first item element from Figure 1. using XPath in Javascript is shown in Figure 4. The function will return an object XPathResult, on which we can perform JavaScript operations.

```
var  xpathResult =
xml.evaluate('product/options/item[5]',
'product.xml', null,
XPathResult.ANY_TYPE, null);
```

Figure 4. XPath and document.evaluate, where xml is the loaded XML document

2.2.2. JSON

JSON syntax is a subset of JavaScript syntax and originally it did not have a set of useful tools and technologies to selectively extract data, like XML

does. It only had the simplest option: of parsing the JSON file into an object and accessing those object properites the way you usually would in the selected language. Due to the fact that JSON is a natural representation of data for the C family of programming languages, the chances are high that the particular language has native syntax elements to access a JSON structure.

However, a need for the kind of technologies for JSON that would parallel the XML ones has been recognized and today we have some options for traversing/filtering JSON data, e.g.: JSPath, json:select () (inspired more by CSS selectors), JSONPath (inspired more by XPath). With these technologies data may be interactively found and extracted out of JSON structures on the client without special scripting. JSON data requested by the client can be reduced to the relevant parts on the server, such minimizing the bandwidth usage of the server response. [2]

In Figure 5. the JavaScript function JSON.parse (text) is used to convert a JSON string into a JavaScript object. A JSON parser will recognize only JSON text and will not compile scripts. In browsers that provide native JSON support, JSON parsers are also faster. [8]

```
var obj = JSON.parse(string);
```

Figure 5. JSON parsing in JavaScript

Parsing a JSON file is a bit more complicated with pure JavaScript, but if there's an option of using a JavaScript library, such as JQuery, we can use built-in functions to achieve that.

```
jQuery.getJSON( products.json );
```

Figure 6. JQuery getJSON function

2.3 Extensibility

The XML document, because of the possibility of expanding XML attributes and CDATA sections, can store all possible data types. It includes the ability to transfer the structure and formatting of the document. This makes it much more flexible, but also harder to read. So much flexibility is good for transferring documents, which can contain images, graphs, text, etc., but not suitable for simple data transmissions, because it's redundant and unnecessarily complex.

Storing data in JSON is limited to common data types (numbers - integer or floating point, string, boolean, array, object, null), which for OOP languages means simple parsing. [4] JSON is the best tool for transferring simple data because data is stored in strings and records while in XML data is stored in a treelike structure, not the most ideal structure for programming languages to understand.

3 Performance testing [6]

In this performance test we will explore the average speeds by which JSON and XML decode simple example data on the server side. Language of choice is PHP. We'll use the function microtime (), that returns the current Unix timestamp with microseconds, to calculate the time needed for the functions to process the given data.

```
<?php
$json = "[55, 'text goes here', 0.1]";
$xml = "<array><int>55</int><string>text
goes
here</string><float>0.1</float></array>"
;
$before = microtime(true);
for ($i=0 ; $i<100000; $i++) {
       simplexml_load_string($xml);
}
$after = microtime(true);
echo '<br/>XML: '.($after-$before)/$i .
" s/load-string\n";

$before = microtime(true);
for ($i=0 ; $i<100000; $i++) {
       json_decode($json);
}
$after = microtime(true);
echo '<br/>JSON: '.($after-$before)/$i .
" s/decode\n";
?>
```

Figure 2. Performance test PHP code

Table 1. Performance testing results (in s/function)

	First run	Second run	Third run
XML	2.942 E-5	3.174 E-5	3.533 E-5
JSON	7.856 E-6	7.765 E-6	7.612 E-6

Average
XML: 0.0000321626 s/load-string
JSON: 0.00000774419 s/decode
(1)

Difference
XML - JSON = 0.00002441841 s/function
(2)

This performance test uses PHP language to decode a simple array coded in the XML and JSON formats. Functions used to accomplish that are: `simplexml_load_string()` for XML and `json_decode()` for JSON.

In (1) and (2) we see that JSON outperforms XML in PHP decoding a simple array by running the function more than 4 times faster.

4 Conclusion

XML and JSON as data storage and transmission formats have their strengths and drawbacks, which determine their usefulness for certain purposes. To transfer documents with a lot of different data types

and elements, XML is the ideal choice. JSON is better suited for dynamic web applications and simple data transmissions. [5] JSON performance speed is greater than XML's because of its simple structure and ease of access to data.

JSON will not fully replace XML in the area of the Web. XML, because of its rich features, has its place in the transfer and validation of documents. JSON is better suited to data-interchange [4] and should be used instead of XML in data transmissions between a server and web application, e.g. in Ajax calls.

References

[1.] Fawcett, Joe; Ayers, Danny; Quin, Liam R. E. Beginning XML, 5th Edition, John Wiley & Sons, Inc., USA, 2012.

[2.] Gössner, Stefan. JsonPath, http://goessner.net/articles/JsonPath/, downloaded: July 11th 2014.

[3.] Graham, James. Introduction to using XPath in JavaScript, https://developer.mozilla.org/en/docs/Introduction_to_using_XPath_in_JavaScript, downloaded: July 11th 2014.

[4.] JSON. Introducing JSON, http://json.org/, downloaded: May 12th 2014.

[5.] Lindo S. XML vs. JSON - A Primer, http://www.programmableweb.com/news/xml-vs.-json-primer/how-to/2013/11/07, downloaded: May 14th 2014.

[6.] mario. phpjson_decodingvs xml parsing, http://stackoverflow.com/questions/4288849/php-json-decoding-vs-xml-parsing, downloaded: May 14th 2014.

[7.] Perkins L. Why JSON will continue to push XML out of the picture, http://blog.appfog.com/why-json-will-continue-to-push-xml-out-of-the-picture, downloaded: May 13th 2014.

[8.] w3schools. JSON Files, http://www.w3schools.com/json/json_files.asp, downloaded: July 11th 2014.

[9.] w3schools. XPath Nodes, http://www.w3schools.com/XPath/xpath_nodes.asp, downloaded: July 11th 2014.

[10.] Zazueta, R. API Data Exchange: XML vs. JSON, http://www.mashery.com/blog/api-data-exchange-xml-vs-json, downloaded: May 13th 2014