Comparative study of XML and JSON using UBER platform

Hareesh

1Department of Computer Science, National Tsing Hua University, Taiwan

Abstract. Extensible Markup Language (XML) is a text format derived from Standard Generalized Markup Language (SGML). Compared to SGML, XML is simple. Hyper Text Markup Language (HTML), by comparison, is even simpler. Even so, a good reference book on HTML is an inch thick. This is because the formatting and structuring of documents is a complicated business. Most of the excitement around XML is around a new role as an interchangeable data serialization format. XML provides two enormous advantages as a data representation language when compared to HTML firstly it is text-based and It is position-independent. These together encouraged a higher level of application-independence than other data-interchange formats. The fact that XML was already a W3C standard meant that there wasn't much left to fight about (or so it seemed). Unfortunately, XML carries a lot of baggage, and it doesn't match the data model of most modern programming languages. There is another text notation that has all of the advantages of XML, but is better suited to data-interchange. That notation is JavaScript Object Notation (JSON). JSON promises the same benefits of interoperability and openness, but without the disadvantages. This paper starts with an introduction to UBER the world’s largest cab service providing company by means of Electronic Enterprise Integration, compares XML and JSON and concludes with which technology can be used in the context of UBER’s Electronic Enterprise Integration.

Keywords. XML v/s JSON, UBER data synchronization, JSON for EEI.

1. Introduction

UBER is the world’s largest cab service company that doesn't own even a single cab. UBER was founded as “UberCab” by Travis Kalanick and Garrett Camp in 2009. UBER was estimated to be worth $62.5 Billion in 2014[1]. UBER has a global presence in 60 countries and 120 cities with services increasing at a rate of 4 cities a month. UBER uses data storage architecture for its Enterprise Integration which is distributed globally because of two important reasons first being when data is physically stored in many places it reduces the risk of a single point of failure situations and second because the closer the data is physically to a user the faster it can be sent and received (higher transfer speeds).

Figure 1. Global data distribution.
UBER’s distributed EEI needs technologies to transfer data. In this paper we try to compare and contrast 2 such data transfer/synchronization technologies XML (extensible Markup Language) and JSON (Java Script Object Notation).

2. UBER

UBER is an on-demand car service that allows you to request private drivers through applications for I-Phone and Android devices. The service utilizes dispatch software to send the nearest driver to your location. Don't be fooled into thinking this is some sort of shared ride or cab service - UBER will send a private taxi-style car. The service provides a cashless solution that charges your ride directly to the credit card on file with your account.

![UBER Application](image)

**Figure 2. UBER Application.**

2.1. How UBER works

The beauty of UBER is its simplicity and convenience. You request a car, an available driver accepts your request, you receive a text message notifying you a car is on its way (along with it's estimated arrival time) and another once it has arrived. Get in, tell the driver where to take you, and relax. Once you reach your destination, there’s no need to hand any money to your driver – tip is included and your fare is automatically deducted from your credit card. All you need is an UBER account and a working cell phone[1-4].

If we represent UBER’s process in terms of producer and consumer. The cab owner and drivers will make the producer and the customer wanting to take a cab ride can be seen as the consumer. The service the consumer consumes is the commute which is the physical activity of getting from place A to B which is delivered by the producer.

![UBER Process Model](image)

**Figure 3. UBER Process Model**
2.2. Process model Conventional V/S UBER

Conventional cab companies own cabs and hire drivers. They own both cabs and cab drivers for a fixed amount of time in a day. In producer consumer representation they it can be said that they own the producer while delivering the product which is the commute.

![Figure 4. Process model of conventional cab companies.](image1)

In its process model UBER however doesn’t own any cab and doesn’t bind a cab driver to a cab. UBER owns only the commute.

![Figure 5. Process model of UBER.](image2)

3. XML

XML is a software- and hardware-independent tool for storing and transporting data. XML stands for Extensible Markup Language. It is a markup language much like HTML which was designed to store and transport data that was designed to be self-descriptive [3]. XML is a W3C Recommendation for data transfer in Enterprise scenarios.

3.1. XML syntax and semantics

XML is used in many aspects of web development. It is often used to separate data from presentation. XML does not carry any information about how information is to be displayed [4]. The same XML data can be used in many different presentation scenarios. Because of this, with XML, there is a full separation between data and presentation.

An XML document contains XML Elements. An XML element is everything from (including) the element's start tag to (including) the element's end tag. An element can contain (text, attributes, other elements or a combination)
A XML file by default has tree view when displayed on web browser. The view when working with an xml file is as below

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```

The below image represents the default tree view when an XML file is opened on a web browser.

```
<bookstore>
  ▼<book category="cooking">
    ▼<title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```
3.2. XML Example

An example of a post it note data can be represented using XML is as follows.

![Note](image)

**Figure 9.** Post it note information.

Now if this information is to represented in XML the information would look like below.

```xml
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

**Figure 10.** XML for the post it.

4. JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language[5-7].

JSON is built on two structures:
- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.
4.1. JSON syntax and semantics

JSON is simple and has only 2 fundamental entities to describe data to be stored. They are objects and arrays. The values that can be stored in these entities can be different. More details as follows.

An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).

An array is an ordered collection of values. An array begins with [ (left bracket) and ends with ] (right bracket). Values are separated by , (comma).

A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.

4.2. JSON example

JSON files contain 2 parts. The first being Meta data which is information about data. Meta data is the base template or the data structure. Meta data is defined only once. The second part is data which uses Meta data. There is no restriction on the number of data elements that can be stored in a JSON file.

```json
{
   "schema": "http://json-schema.org/draft-04/schema#",
   "title": "Product",
   "description": "A product from Acme's catalog",
   "properties": {
      "id": {
         "description": "The unique identifier for a product",
         "type": "integer"
      }
   }
}
```

Figure 14. JSON Meta-Data.
5. Comparison of XML V/S JSON technology

The below table is compilation of comparison and contrasts of main features of XML and JSON technology.

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>XML</th>
<th>JSON</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simplicity</td>
<td>XML is simpler than SGML</td>
<td>JSON is much simpler than XML. XML has a much smaller grammar and maps more directly onto the data structures.</td>
<td>Simplicity is subjective. XML can easily be used as simply as JSON syntactically. But simplicity comes at the price of ignoring many common more robust information sharing needs in an extended network – rather than just point-to-point. The mapping referenced here is for objects within a JavaScript environment only. Outside of that context this is not so. All major programming environments have robust XML support.</td>
</tr>
<tr>
<td>2</td>
<td>Extensibility</td>
<td>XML is extensible because it is a mark-up language</td>
<td>JSON is not extensible because it does not need to be.</td>
<td>JSON is not a document markup language, so it is not necessary to define new tags or attributes to represent data in it. This sometimes a naïve view. Things change constantly with new information and types of information with type of sharing needs. Particularly as more participants are added to exchanges and standards evolve. Only in limited cases such as Twitter can we see set formats.</td>
</tr>
<tr>
<td></td>
<td>Interoperability</td>
<td>XML is an interoperability standard.</td>
<td>JSON has the same interoperability potential as XML.</td>
<td>JSON clearly has significant limitations and gaps with regard to information semantics and reuse because it’s a newer technology and xml has been in the market for a long time.</td>
</tr>
<tr>
<td>---</td>
<td>------------------</td>
<td>--------------------------------------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.</td>
<td>Openness</td>
<td>XML is an open standard</td>
<td>JSON is at least as open as XML, perhaps more so because it is not in the center of corporate/political standardization struggles.</td>
<td>This is a highly subjective parameter to be analyzed. XML has proven to be universally adopted and implemented not just in software but firmware devices and communications systems. Also Java script is an intricate part of JSON working is not immune from manipulation as anything else as happened with JavaScript itself.</td>
</tr>
<tr>
<td>5.</td>
<td>Multi Media support</td>
<td>XML is strictly text only</td>
<td>JSON support multimedia</td>
<td>Since structures are not hard bond JSON is flexible and can support many types of media such as audio, video, 3D rendering etc.</td>
</tr>
<tr>
<td>6.</td>
<td>Object-Oriented</td>
<td>XML is document-oriented.</td>
<td>JSON is data-oriented.</td>
<td>JSON can be mapped more easily to object-oriented systems. The reverse is an issue however, objects do not necessarily map easily to documents. Plus not all content is objects; it actually constrains the use model. XML on the other hand is well equipped for use as object-oriented content as well as documents.</td>
</tr>
</tbody>
</table>
| 7. | Adaptation       | XML is just beginning to become known. Its simplicity and the ease of converting XML to JSON make JSON ultimately more adoptable. | The use of JSON is limited to web client-server scenarios. Within that domain it is popular. Outside of that domain XML completely dominates the data interchange and transfer technology for many years now and could continue to do so for years to come.
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8.</td>
<td><strong>Self-Describing Data</strong></td>
<td>This is a key XML design objective.</td>
<td>XML and JSON have this in common. However XML has richer semantics available than JSON to describe data.</td>
</tr>
<tr>
<td>9.</td>
<td><strong>Data Views</strong></td>
<td>XML displays many views of one data</td>
<td>JSON does not provide any display capabilities because it is not a document markup language. XML has broader applicability. Therefore you can write once, use everywhere. While JSON can be expected to be changed into XML for such extended uses.</td>
</tr>
<tr>
<td>10.</td>
<td><strong>Code Re-invention</strong></td>
<td>There is a wide range of reusable software available to programmers to handle XML so they don’t have to re-invent code.</td>
<td>JSON, being a simpler notation, needs much less specialized software. JSON is mainly available in JavaScript and not in a wide range of programming environments. Further it is not the simplicity of the syntax; it is the drastically reduced capabilities. Hence JSON only provides very limited functionality.</td>
</tr>
<tr>
<td>11.</td>
<td><strong>common exchange format</strong></td>
<td>XML is a better document exchange format. Use the right tool for the right job.</td>
<td>JSON is a better data exchange format. This is entirely subjective again and no metrics are being given here. What defines the term “better”? Clearly JSON is significantly less capable and restricted in its use cases. Therefore “mileage may vary in actual use and what area it is used” would be an appropriate caution here when trying to measure what is “better” where and how.</td>
</tr>
<tr>
<td>12.</td>
<td><strong>Internation alization</strong></td>
<td>XML and JSON both use Unicode.</td>
<td>XML and JSON both use Unicode. However JSON has some limitations in its use of encoding and exchanges.</td>
</tr>
</tbody>
</table>
| 13. | **Human Readable** | XML is human readable | JSON is much easier for human to read than XML. It is easier to write, too. It is also easier for machines to read and write. Again this is an entirely subjective statement. Markup is markup there is no “easier” here. Machines have no notion of “easier”. The notion of “easier to read” and presumably comprehend the meaning of is notoriously hard to
6. Using XML and JSON in UBER application

This section represents sample UBER data.

- UBER represented in XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<UBER>
  <OperatingCity country="India" lang="en">Delhi</OperatingCity>
  <User id="1000001" DisplayName="Kunal" PhoneNumber="09953459848">Kunal Kapoor</User>
  <CurrentLocation>
    <Latitude>28.436944</Latitude>
    <Longitude>77.098810</Longitude>
  </CurrentLocation>
  <Destination>
    <Latitude>28.436944</Latitude>
    <Longitude>77.098810</Longitude>
  </Destination>
  <UberMoney CurrencyType="IndianRupee">1034</UberMoney>
</UBER>
```

Figure 16. UBER XML

- UBER information in JSON

```json
{
    "name": "http://www.uber.com/uber-api/1.0/uber",
    "title": "UBER-India",
    "description": "string describing information about Uber user",
    "type": "User",
    "properties": {
        "id": {
            "description": "The unique identifier for Uber user",
            "type": "string"
        },
        "DisplayName": {
            "description": "User display name",
            "type": "string"
        },
        "PhoneNumber": {
            "description": "User phone number",
            "type": "integer"
        }
    }
}
```

Figure 18. JSON schema definition
7. Conclusions

From the comparative analysis above we can see that XML’s biggest advantage is it is time tested in Enterprise integration context and JSON biggest advantage is it is young and takes into consideration to modern programming architectures along with technological evolution changes like things such as multimedia etc.

For UBER’s scenario my personal view is that a healthy combination of XML and JSON can be used to achieve EEI. This combination model is based on the law of requisite variety which says “only variety can absorb variety”. The combination approach where in some parts of UBER use XML and some parts use JSON would help to leverage on the maturity of XML and the modernity of JSON. Such combinations can help achieve both reliability and speed in today’s ever changing dynamic business needs.

8. Acknowledgement

This research is for the final project of Electronic Enterprise Integration course of Industrial engineering and engineering management course of NTHU, Taiwan. I thank Professor. Amy Trappey for patient guidance.

References