JSON vs XML

Gofur Halmruatov
Elena Myazina
• Introduction XML and JSON
• Comparing JSON vs. XML
• Conclusion and references
XML

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
XML: Well-formed

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must always be quoted
XML: Validity

- Valid XML is an XML document that has an associated document type declaration and complies with the constraints expressed in it.
  - Document Type Definition (DTD)
  - XML Schema
  - Relax NG
  - ISO DSDL
XML: Processing files

- Programming languages and SAX API
- Programming Languages and DOM API
- And Others
XML: Transformation engines and filters

- **XSL-FO** is a declarative, XML-based page layout language. An XSL-FO processor can be used to convert an XSL-FO document into another non-XML format, such as PDF.

- **XSLT** is a declarative, XML-based document transformation language. An XSLT processor can use an XSLT stylesheet as a guide for the conversion of the data tree represented by one XML document into another tree that can then be serialized as XML, HTML, plain text, or any other format supported by the processor.
**XML: Transformation engines and filters**

- **XQuery** is a W3C language for querying, constructing and transforming XML data.
- **XPath** is a DOM-like node tree data model and path expression language for selecting data within XML documents. XSL-FO, XSLT and XQuery all make use of XPath. XPath also includes a useful function library.
- **Xlink** - is an XML markup language used for creating hyperlinks in XML documents.
XML: Advantages

- XML provides a basic syntax that can be used to share information between different kinds of computers, different applications, and different organizations.
- With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc)
- XML provides a gateway for communication between applications, even applications on wildly different systems. As long as applications can share data (through HTTP, file sharing, or another mechanism)
- It supports Unicode, allowing almost any information in any written human language to be communicated.
- It can represent common computer science data structures: records, lists and trees.
- Its self-documenting format describes structure and field names as well as specific values.
- It is based on international standards.
XML: DisAdvantages

- It is difficult for the end-user to understand its capabilities.
- XML syntax is redundant or large relative to binary representations of similar data, especially with tabular data.
- The redundancy may affect application efficiency through higher storage, transmission and processing costs.
- XML syntax is verbose, especially for human readers, relative to other alternative 'text-based' data transmission formats.
- The hierarchical model for representation is limited in comparison to an object oriented graph.
- Expressing overlapping (non-hierarchical) node relationships requires extra effort.
- XML namespaces are problematic to use and namespace support can be difficult to correctly implement in an XML parser.
JSON

- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format
- **JSON** code is valid JavaScript
- **JSON** is a text format that is completely language independent
- **JSON** is built on two structures:
  - A collection of name/value pairs
  - An ordered list of values
- **JSON** file extension:  *json
JSON: Object, Array, Number
JSON: String, Value

- "Any UNICODE character except " or \\ or control character"
- "quotation mark"
- "reverse solidus"
- "solidus"
- "backspace"
- "formfeed"
- "newline"
- "carriage return"
- "horizontal tab"
- "4 hexadecimal digits"

- "value"
- "string"
- "number"
- "object"
- "array"
- "true"
- "false"
- "null"
JSON: Example

```json
{
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        "212 555-1234",
        "646 555-4567"
    ]
}
```
JSON: in JavaScript

- **JSON** is a subset of the object literal notation of JavaScript
  ```javascript
  var myJSONObject = {
    "bindings": [
      {
        "ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},
      {
        "ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},
      {
        "ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}
    ]};
  ```
- Members can be retrieved using dot or subscript operators.
  ```javascript
  myJSONObject.bindings[0].method
  ```
- To convert a **JSON** text into an object, you can use the `eval()` function. `eval()` invokes the JavaScript compiler
  ```javascript
  var myObject = eval('(' + myJSONtext + ')');
  ```
- In browsers that provide native JSON support, JSON parsers are also much faster than eval
  ```javascript
  var myObject = JSON.parse(myJSONtext, reviver);
  ```
JSONRequest

- **JSONRequest** is proposed as a new browser service that allows for two-way data exchange with any JSON data server without exposing users or organization to harm.
- *It* exchanges data between scripts on pages with JSON servers in the web.
- *It* is hoped that browser makers will build this feature into their products in order to enable the next advance in web application development.
- **JSONRequest** is a global JavaScript object. It provides three methods: post, get, and cancel.
- **JSONRequest** does not send or receive cookies or passwords in HTTP headers
- **JSONRequest** works only with JSON text
- Responses will be rejected unless they contain a JSONRequest content type
JSON: in AJAX

```javascript
var the_object;
var http_request = new XMLHttpRequest();
http_request.open( "GET", url, true );
http_request.send(null);
http_request.onreadystatechange = function () {
    if ( http_request.readyState == 4 ) {
        if ( http_request.status == 200 ) {
            the_object = eval( "(" + http_request.responseText + ")" );
        } else {
            alert( "There was a problem with the URL." );
        }
    }
    http_request = null;
};
```
### Comparing Criteria: Human Readability

<table>
<thead>
<tr>
<th>XML</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;person&gt;</code></td>
<td><code>{</code></td>
</tr>
<tr>
<td><code>&lt;firstname&gt;</code>Subbu<code>&lt;/firstname&gt;</code></td>
<td><code>&quot;firstName&quot;: &quot;Subbu</code></td>
</tr>
<tr>
<td><code>&lt;lastname&gt;</code>Allamaraju<code>&lt;/lastname&gt;</code></td>
<td><code>&quot;lastName&quot;: &quot;Allamaraju&quot;</code></td>
</tr>
<tr>
<td><code>&lt;/person&gt;</code></td>
<td><code>}</code></td>
</tr>
</tbody>
</table>

Both formats Human readability are almost in same degree
Comparing Criteria: Data Creation

**XML:** JAXB or XmlBeans

```java
1. Person person = new Person();
2. person.setFirstName("Subbu");
3. person.setLastName("Allamaraju");
4. Marshaller marshaller = ... // Create a marshaller instance
5. marshaller.marshal(person, outputStream);
```

**JSON:** Json-lib ([http://json-lib.sourceforge.net](http://json-lib.sourceforge.net))

```java
1. Person person = new Person();
2. person.setFirstName("Subbu");
3. person.setLastName("Allamaraju");
4. writer.write(JSONObject.fromObject(person).toString());
```
Comparing Criteria: Extensibility

- **XML**

  ```javascript
  1. var xml = xhr.responseTextXML;
  2. var elements = xml.getElementsByTagName("firstName");
  3. var firstNameEl = elements[0];
  4. var lastNameEl = firstNameEl.nextSibling;
  
```

- **JSON**

  ```javascript
  1. alert(person.middleName);
  ```
Comparing Criteria: Speed

- Each format has its advantages and disadvantages
- JSON is smaller than XML
- JSON is part of JavaScript code, it needs less time to parse data than the time for XML
- JSON is faster than XML in AJAX
Comparing Criteria: Fetching data from Server

- **XML** uses XMLHttpRequest to fetch data from Server
- **JSON** can fetch data from Server with and without XMLHttpRequest in AJAX, it also uses **JSONRequest**
- **Speed** fetching data is almost same, but speed parsing fetched data is different
Comparing Criteria: Security

- **JavaScript eval()** function does not check json code before running it, this is place for code injection attack.

```javascript
var person = eval(xhr.responseText);
alert(person.firstName);
```

- **Solution**

```javascript
var my_JSON_object = !(/^[^,:{}\[\]0-9.-Eaeflnr-u \n\r\t]/.test(text.replace(/\n\s+[^\n]+/g, ' ')))
&& eval('(' + text + ')');
```
Comparing Criteria: Others

- JSON is thinner than Xml
- JSON is native to the client (most browsers support IE8, Firefox and others)
- JSON can do cross domain calls that XMLHttpRequest cannot
- It is easier to encode / escape in JSON than XML
Conclusion

What to use depends on the applications and your requirements. For data-oriented applications, I prefer JSON to XML due to its simplicity and ease of processing on the client side. XML may be great on the server side, but JSON is definitely easier to deal with on the client side.
References

- http://www.json.org/
- http://www.w3schools.com/XML/
- http://www.w3.org/XML/
- http://wikipedia.org/JSON
- http://www.subbu.org/blog/2006/08/json-vs-xml