**XML VS JSON**

|  |  |
| --- | --- |
| JSON object has a type | XML data is typeless |
| JSON types: string, number, array, Boolean | All XML data should be string |
| Data is readily accessible as JSON objects | XML data needs to be parsed. |
| It supports only UTF-8 encoding. | It supports various encoding. |
| It is less secured | It is more secure than JSON |

JSON

**Easy to use** – JSON API offers high-level facade, which helps you to simplify commonly used use-cases.

**Performance** – JSON is quite fast as it consumes very less memory space, which is especially suitable for large object graphs or systems.

**Free tool** – JSON library is open source and free to use.

**Clean JSON** – Creates clean, and compatible JSON result that is easy to read.

**Dependency** – JSON library does not require any other library for processing.

**JSON (JavaScript Object Notation)** is a lightweight data-interchange format and it completely language independent. It is based on the JavaScript programming language and easy to understand and generate.

**XML (Extensible markup language)** was designed to carry data, not to display data. It is a W3C recommendation. Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML focus on simplicity, generality, and usability across the Internet.

**Why XML -** XML provided a way for the industry to specify, with strict semantics, custom markup languages for any application. With the keyword being “strict semantics,” XML defined a standard that could assert the integrity of data in any XML document, of any XML sub-language. For software companies developing distributed enterprise applications that interface with disparate systems, a markup language that could assert the integrity of its data was significant. By defining structured content with XML, companies leveraged the features of this technology to interoperate with any platform, enforce data integrity of every data interchange, and to systematically reduce the software risk of their systems. For the industry, XML provided a technology to store,

communicate, and validate any kind of data, in a form that applications on any platform can easily read and process.

* Makes documents transportable across systems and applications. With the help of XML, you can exchange data quickly between different platforms.
* XML separates the data from HTML
* XML simplifies platform change process
* Allows creating user-defined tags.

**DOM Parser VS SAX Parser**

**DOM:** Dom stands for Document Object Model. The DOM API provides the classes to read and write an XML file. DOM reads an entire document. It is useful when reading small to medium size XML files. It is a tree-based parser and a little slow when compared to SAX and occupies more space when loaded into memory. We can insert and delete nodes using the DOM API.

[**SAX**](https://www.geeksforgeeks.org/java-sax-library/): (Simple API for XML), is the most widely adopted API for XML in Java and is considered the de-facto standard. Although it started as a library exclusive for Java, it is now a well-known API distributed over a variety of programming languages. It is an open-source project and has recently switched to SourceForge project infrastructure that makes it easier to track open SAX issues outside the high-volume XML-dev list

**Key Difference of DOM and SAX Parser:**

* DOM stands for Document Object Model while SAX stands for Simple API for XML parsing.
* DOM parser load full XML file in-memory and creates a tree representation of XML document, while SAX is an event based XML parser and doesn’t load whole XML document into memory.
* If you know you have sufficient amount of memory in your server you can choose DOM as this is faster because load entire XML in-memory and works as tree structure which is faster to access.
* As a thumb rule, for small and medium sized XML documents, DOM is much faster than SAX because of in memory agnostic.
* As a thumb rule, for larger XML and for frequent parsing, SAX XML parser is better because it consume less memory.

Simply Dom Parser vs XML Parser

**SAX (Simple API for XML):** Is a stream-based processor. You only have a tiny part in memory at any time and you "sniff" the XML stream by implementing callback code for events like tagStarted() etc. It uses almost no memory, but you can't do "DOM" stuff, like use xpath or traverse trees. **It is event based.** Can’t Insert and Delete node. Faster run time. Parse XML File line by line.

**DOM (Document Object Model):** You load the whole thing into memory - it's a massive memory hog. You can blow memory with even medium sized documents. But you can use xpath and traverse the tree etc. **It is tree based.** Can Insert or Delete node. Slower run time.



Ref:

1.https://stackoverflow.com/questions/6828703/what-is-the-difference-between-sax-and-dom

2. https://www.edureka.co/blog/parsing-xml-file-using-sax-parser/