<https://octopus.com/blog/state-of-config-file-formats>

**What's the Difference Between JSON, XML, and YAML?**

Apr 28th, 2015

These text-file formats can all be used to store structured data for embedded and Web applications—but there are differences among them.

What are JSON, XML, and YAML? Text-file formats that can be used to store structured data that can be handy for embedded and Web applications.

Most developers will be familiar with XML (Extensible Markup Language) and its flexible but powerful markup capabilities (see “[XML: Flexibility Where It Counts](https://www.electronicdesign.com/embedded/xml-flexibility-where-it-counts)”). It is often used in configuration and preference files like those used for the Eclipse IDE. Most Web browsers have XML viewers, although XML is designed for structured data, making it a bit like looking at the internals of a database.

JavaScript Object Notation (JSON) is used with JavaScript, of course. It will be familiar to Web developers that use it for client/server communication.

YAML stands for YAML Ain’t Markup Language. It uses line and whitespace delimiters instead of explicitly marked blocks that could span one or more lines like XML and JSON. This approach is used in many programming languages, such as Python.

Each has its advantages and supporters. This is not a definitive description of JSON, XML, and YAML, but rather a short overview that would allow someone to actually understand how they work and possibly read existing examples. All are applicable to embedded applications where data is stored and exchanged. Each one is very popular, although there are other alternatives as well. Functions to read and write these formats are available in just about every programming language, although general data conversion works best for programming languages that support keyed collections. We will defer the discussion of schemas to another article.

**XML**

XML is defined by the [World Wide Web Consortium's (W3C')](http://www.w3.org/) XML 1.0 Specification. It was designed to be general, allowing it to be used in a wide range of applications. It is a balanced version of Hypertext Markup Language (HTML) and there is the Extensible Hypertext Markup Language (XHTML) definition. HTML was less strict leading to more compact files *(Fig. 1)*.

<code class="language-html4strict"><p>

This is a paragraph.

<p>

This is another paragraph.

</code>

*1. Initially HTML allowed unbalanced prefixes.*

Of course, this leads to some interesting problems. It turns out to be good practice to have “well-formed” definitions so, for example, paragraph definitions are bounded by

and

(*Fig. 2*).

<code class="language-html4strict"><p>

This is a paragraph.

</p>

<p>

This is another paragraph.

</p>

</code>

*2. Web browsers prefer HTML that is properly formatted like this.*

HTML has specific syntax and semantics to address presentation issues like layout, fonts, and so on.

XML is stricter as is XHTML. XML requires well-formed data *(Fig. 3)*.

<code class="language-xml"><people>

<person>

<firstName>Charles</firstName> <lastName>Schulz</lastName>

</person>

<person>

<firstName>Walt</firstName> <middleName>Elias</middleName>

<lastName>Disney</lastName>

</person>

<person>

<firstName>Gary</firstName> <lastName>Larson</lastName>

</person>

</people>

</code>

*3. This XML data is a hierarchy of people and their names.*

The indents in the example make XML easier to read, but the whitespace is irrelevant. This is highlighted by the name elements (firstName, middleName, and lastName) that appear on a single line or multiple lines. Note the matching “tags” with the trailing tag starting with a slash character, “/”.

Tags can also include attributes *(Fig. 4)*. The example also shows an alternate form where the trailing tag is eliminated as designated by the trailing slash. This is used when there is no data although information is usually provide in the form of attributes.

<code class="language-xml"><img src='image-file.jpg' alt='alternate text' />

</code>

*4. Tags can include attributes like src and alt in this example. This example also highlights the alternative syntax for a tag with no data indicated by the trailing slash.*

XML has the advantage over JSON and YAML when complex data structures come into play. It has a higher overhead and making it more work if the creator is a human rather than a program.

**JSON**

JSON is a simpler encoding method that retains the flexible entry format of XML. Two standards address JSON at this point, RFC 7159 and ECMA-404. RFC 7159 addresses some security and semantic issues whereas ECMA is primarily a syntax definition.

JSON uses name/value pairs. It also has a number of basic data types including numbers, strings, booleans, and null. It also supports arrays and objects (*Fig. 5*).

<code class="language-javascript">{

âpeopleâ: [

{ âpersonâ: {

âfirstNameâ: âCharlesâ,

âlastNameâ: âSchulzâ

}

},

{ âpersonâ: {

âfirstNameâ: âWaltâ,

âmiddleNameâ: âEliasâ,

âlastNameâ: âDisneyâ

}

},

{ âpersonâ: {

âfirstNameâ: âGaryâ,

âlastNameâ: âLarsonâ

}

}

]

}

</code>

*5. JSON’s name/value pairs are collected in a structured object bounded by curly brackets. Arrays are indicated by square brackets.*

JSON’s syntax matches JavaScript, but typically a parse function is used to convert JSON text to a JavaScript object. This adds a level of protection from malicious code since JSON data is often sent over the Internet on an unsecure channel. It also addresses bad data issues.

JSON is often used with JavaScript Ajax techniques to exchange data. This can provide a more dynamic, interactive interface for a Web page. JSON support is found in most Web browsers.

**YAML**

JSON is simpler than XML. YAML is even simpler *(Fig. 6)*. It foregoes the brackets, except for inline collections, and uses vertical alignment to indicate structure. Quotes are optional, as everything is essentially a string. Leading and trailing whitespace is ignored, so quotes can still be used especially if special characters are part of a key string.

<code class="language-javascript">people:

-

person: {firstName: Charles, lastName: Schulz }

-

person:

firstName: Walt

middleName: Elias

lastName: Disney

characters: [ Mickey, Donald, Goofy ]

-

person:

firstName: Gary

lastName: Larson

</code>

*6. YAML also uses name/value pairs. Curly brackets indicate lists of pairs. Arrays are indicated by dashes, “-”, or square brackets.*

In the example, the person: key could not be used on a line by itself because keys within a structure at a particular level must be unique. The dash indicates an array element. Arrays are just structures with numeric keys. YAML syntax options allow the same data to be represented in different ways *(Fig. 7)*.

<code class="language-javascript">people:

0:

person: {firstName: Charles, lastName: Schulz }

1:

person:

firstName: Walt

middleName: Elias

lastName: Disney

characters:

- Mickey

- Donald

- Goofy

2:

person:

firstName: Gary

lastName: Larson

</code>

*7. YAML’s different syntax options allow data to be presented in different ways.*

As with Python, it helps to have a text editor that understands the syntax—IDEs like Eclipse that have text editors that understand YAML and do things like auto indentation or column moves.

Functions are available to convert native data structures to and from XML, JSON, and YAML. These are typically used by an application and often a user will never see this data. Still, many platforms utilize this for configuration information that a programmer or user generates.

For example, [Drupal](http://drupal.org/), a content management system (CMS), is now based on [Symfony](http://symfony.com/), a PHP framework, that uses YAML for its configuration files. There is actually a configuration management system that converts YAML configuration files for system modules and stores them in a database in serialized, PHP data format.

JSON, XML and YAML have their place in many existing environments like Drupal and Symfony. It is a requirement for using those platforms. For new applications, the choice is yours.

Bottom of Form

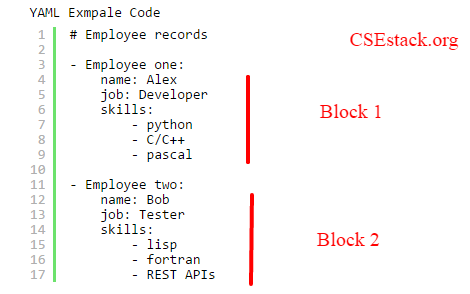
**YAML vs JSON vs XML | What is the Difference Between Them?**

YAML, XML, and JSON are the widely used data serialization language.  We will see difference Difference Between YAML and JSON and XML.

* YAML stands for “*YAML Aint Markup Language*“.
* JSON stands for “*JavaScript Object Notation*“.
* XML is “*eXtensible Markup Language”* whereas YML is not a markup language.
* XML uses a tag to define the structure just like HTML.
* YAML uses indentation to define structured data. So each block in the YAML is differentiated by the number of white spaces.
* All three mentioned serialization language has same extension as their name. (.yaml for YAML, .json for JSON, .xml for XML). So it is easy to remember.
* In fact, file extensions are arbitrary for all the three data serialization standard. It is useful for the application and users to know what files format, type of the content and their data structure.

**YAML Example**

In this YAML example, you can see that there is an equal number of white spaces in each block.



**JSON Example**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | {  "EmpRecord": {  "Employee": [  {  "-id": "emp01",  "name": "Alex",  "job": "Developer",  "skills": "python, C/C++, paskal"  },  {  "-id": "emp02",  "name": "Bob",  "job": "Tester",  "skills": "lips, forton, REST APIs"  }  ]  }  } |

**XML Example**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?xml version="1.0"?>  <EmpRecord>  <Employee id="emp01">  <name>Alex</name>  <job>Developer</job>  <skills>python, C/C++, paskal</skills>  </Employee>    <Employee id="emp02">  <name>Bob</name>  <job>Tester</job>  <skills>lips, forton, REST APIs</skills>  </Employee>    </EmpRecord> |

**Differentiating YAML vs JSON vs XML by its Applications**

Now many find it difficult which data serialization language should be used for project development.

* When you talk about Javascript, JSON is the most prominent serialization language.
* For [**Java programming**](https://www.csestack.org/java/), you must have seen XML is widely used.
* Python has the same indentation technique same as YAML. So Pythonista finds YAML more friendly than other serialization languages.

Ad on to all above points regarding the programming preference. It does not hold true always. If you want to use data serialization language, you should also consider developing techniques for your project.

**Differentiating YAML, JSON and XML according to the project need**

**Q: How should I choose the data serialization language for my next project?**

Data is represented in serialization language(YAML/JSON/XML) which is well structured. Using any of the serialization languages, consist of four major activities.

* You need to parse the content from this serialization language.
* Read the required values.
* Manipulate those values.
* Further, you may need to store manipulated values back serialization language file.

To do all these development activities, you have to write the code for parsing.

Fortunately, there are so many modules/libraries or open-source code in various core languages (like C, C++, Java…) to parse serialized data.

**Q: Why should not you waste your time writing YAML/JSON/XML parser?**

There can be some mistakes if you develop it by yourself. It’s recommended/good to use prominent existing libraries. It also saves your time.

So before using serialization language in your project; research, find parsing libraries, read documents. Find compatibility of the parser to use in your project.

If you get the parsing libraries as per your requirement, you have almost curated all data handling activities. Isn’t it easy to call simple one function to parse values rather than parsing it manually?

So for the programmer, it is easy to choose YAML vs JSON vs XML by the availability of the data parser.