YAML is one of the most popular languages for writing configuration files.

In this article, you will learn how YAML compares to XML and JSON - two languages also used for creating configuration files.

You will also learn some of the rules and features of the language, along with its basic syntax.

Here is what we will cover:

1. Scalars
2. Collections

# What is YAML?

YAML stands for **YAML A**in't **M**arkup **L**anguage, but it originally stood for **Y**et **A**nother **M**arkup **L**anguage.

YAML is a human-readable data serialization language, just like XML and JSON.

Serialization is a process where one application or service that has different data structures and is written in a different set of technologies can transfer data to another application using a standard format.

In other words, serialization is about translating, converting, and wrapping up a data structure in another format.

The data in the new format can be stored in a file or transmitted to another application or service over a network.

YAML is a widely used format for writing configuration files for different DevOps tools, programs, and applications because of its human-readable and intuitive syntax.

# XML VS JSON VS YAML - What's The Difference?

XML, JSON, and YAML are all used for creating configuration files and transferring data between applications.

Each language has its advantages and disadvantages.

Now, let's see some of the characteristics of the three languages. You will also see an example of how the same code is written in each language to demonstrate the high-level differences in their syntax.

## XML

XML, which stands for Extensible Markup Language, was first introduced in 1996 and was designed for general-purpose use.

XML is a generalized markup language. It offers a structured yet flexible syntax and a defined document schema. This makes it a good choice when working with complex configurations that require a structured format and finer control over schema validation to ensure configurations always have the correct format.

With that said, XML's syntax can be verbose, redundant, and harder to read in comparison with other serialization languages.

```
<Employees>
    <Employee>
        <name> John Doe </name>
        <department> Engineering </department>
        <country> USA </country>
    </Employee>
     <Employee>
        <name> Kate Kateson </name>
        <department> IT Support </department>
        <country> United Kingdom </country>
    </Employee>
</Employees>
```

## JSON

JSON stands for JavaScript Object Notation and has been around since the early 2000s.

JSON was initially inspired by the JavaScript programming language, but it is not tied to only one language. Instead, it is a language-independent format.

Most modern programming languages have libraries for parsing and generating JSON data.

JSON offers a much more readable, human-friendly, compact, and simple syntax compared to XML. It makes for a great format for storing and transferring information between web applications and servers over a network.

With that said, it may not offer the best support for complex configurations.

```
{
    "Employees": [
    {
            "name": "John Doe",
            "department": "Engineering",
            "country": "USA"
        },

        {
            "name": "Kate Kateson",
            "department": "IT support",
            "country": "United Kingdom"
        }
    ]
}
```

## YAML

YAML, originally known as Yet Another Markup Language, was created in 2001 but now stands for YAML Ain't Markup Language.

YAML is an official strict superset of JSON despite looking very different from JSON.

YAML can do everything that JSON can and more. A valid YAML file can contain JSON, and JSON can transform into YAML.

YAML has the most human-readable, intuitive, and compact syntax for defining configurations compared to XML and JSON.

YAML uses indentation to define structure in the file, which is helpful if you are used to writing Python code and are familiar with the indentation style the language uses.

With that said, if you don't get the indentation and format right, it can lead to validation errors, making it not the friendliest for beginners.

```
Employees:
- name: John Doe
  department: Engineering
  country: USA
- name: Kate Kateson
  department: IT support
  country: United Kingdom
```

# Features and Basic Rules of YAML

Now, let's go over some of the basic rules and features of the language.

## How to Create a YAML File

To create a YAML file, use either the `.yaml` or `.yml` file extension.

## Multi-Document Support in YAML

Before writing any YAML code, you can add three dashes ( `---` ) at the start of the file:

```yaml
---
Employees:
- name: John Doe
  department: Engineering
  country: USA
- name: Kate Kateson
  department: IT support
  country: United Kingdom
```

YAML allows you to have multiple YAML documents in a singe YAML file, making file organization much easier.

Separate each document with three dashes ( `---` ):

```yaml
---
Employees:
- name: John Doe
  department: Engineering
  country: USA
- name: Kate Kateson
  department: IT support
  country: United Kingdom
---
Fruit:
 - Oranges
 - Pears
 - Apples
```

You can also use three dots ( . . . ) to mark the end of the document:

```
---
Employees:
- name: John Doe
  department: Engineering
  country: USA
- name: Kate Kateson
  department: IT support
  country: United Kingdom
...
```

## Indentation in YAML

In YAML, there is an emphasis on indentation and line separation to denote levels and structure in data. The indentation system is quite similar to the one Python uses.

YAML doesn't use symbols such as curly braces, square brackets, or opening or closing tags - just indentation.

## Tabs in YAML

YAML doesn't allow you to use any tabs when creating indentation - use spaces instead.

## Whitespace in YAML

Whitespace doesn't matter as long as child elements are indented inside the parent element.

## How to Write A Comment in YAML

To add a comment to comment out a line of code, use the `#` character:

```
---
# Employees in my company
Employees:
- name: John Doe
  department: Engineering
  country: USA
- name: Kate Kateson
  department: IT support
  country: United Kingdom
```

## Explicit Data Types in YAML

Although YAML auto-detects the data types in a file, you can specify the type of data you want to use.

To explicitly specify the type of data, use the `!!` symbol and the name of the data type before the value:

```
# parse this value as a string
date: !!str 2022-11-11

## parse this value as a float (it will be 1.0 instead of 1)
fave_number: !!float 1
```

# An Introduction to YAML Syntax

## Scalars

Scalars in YAML are the data on the page - strings, numbers, booleans, and nulls.

Let's see some examples of how to use each one.

In YAML, strings in some cases can be left unquoted, but you can also wrap them in single ( `' '` ) or double ( `" "` ) quotation marks:

```
A string in YAML!

'A string in YAML!'

"A string in YAML!"
```

If you want to write a string that spans across multiple lines and you want to preserve the line breaks, use the pipe symbol ( `|` ):

```
|
 I am message that spans multiple lines
 I go on and on across lines
 and lines
 and more lines
```

Make sure that the message is indented!

Alternatively, if you have a string in a YAML file that spans across multiple lines for readability, but you want the parser to interpret it as a single line string, you can use the `>` character, which will replace each line break with a space:

```
>
 I am message that spans
 multiple lines
 but I will be parsed
 on one line
```

Again, make sure you don't forget to indent the message!

Numbers express numerical data, and in YAML, these include integers (whole numbers), floats (numbers with a decimal point), exponentials, octals, and hexadecimals:

```
# integer
19

# float
8.7

# exponential
4.5e+13

# octal
0o23
```

```
# hexadecimal
0xFF
```

Booleans in YAML, and other programming languages, have one of two states and are expressed with either `true` or `false`.

Words like `true` and `false` are keywords in YAML, so don't surround them with quotation marks if you want them interpreted as booleans.

Lastly, Null values are expressed with the keyword `null` or the tilde character, `~`.

## Collections

More often than not, you will not be writing simple scalars in your YAML files - you will be using collections instead.

Collections in YAML can be:

- Sequences (lists/arrays)
- Mappings (dictionaries/hashes)

To write a sequence, use a dash ( `-` ) followed by a space:

```
- HTML
- CSS
- JavaScript
```

Each item in the sequence (list) is placed on a separate line, with a dash in front of the value.

And each item in the list is on the same level.

That said, you can create a nested sequence (remember, use spaces - not tabs - to create the levels of indentation):

```
- HTML
- CSS
- JavaScript
 - React
 - Angular
 - Vue
```

In the sequence above, `React, Angular and Vue` are sub-items of the item `JavaScript`.

Mappings allow you to list keys with values. Key/value pairs are the building blocks of YAML documents.

Use a colon ( : ) followed by a space to create key/value pairs:

```
Employees:
 name: John Doe
 age: 23
 country: USA
```

In the example above, a name gets assigned to a specific value.

The value `John Doe` gets mapped (or assigned) to the `name` key, the value `23` gets mapped to the `age` key, and the value `USA` gets mapped to the `country` key. Altogether, these create an object.

You can also use a mapping with a sequence.

For example, taking the example sequence from earlier on, here is how you would build a list of `frontend_languages`:

```
frontend_languages:
 - HTML
 - CSS
 - JavaScript
  - React
  - Angular
  - Vue
```

In the example above, I created a list of `frontend_languages`, where there are multiple values under the same key, `frontend_languages`.

Similarly, you can create a list of objects:

```
Employees:
- name: John Doe
```

```
    department: Engineering
    country: USA
  - name: Kate Kateson
    department: IT support
    country: United Kingdom
```

# Conclusion

Hopefully, this article was helpful and gave you insight into what YAML is, what the syntax of the language looks like, and how it differs from XML and JSON.

Thank you for reading, and happy coding!