YAML QUICK GUIDE

v1.0

A downloadable resource of *The Ultimate YAML Course: Learn YAML from Scratch.*

WWW.CODEWITHPRAVEEN.COM

Preface

Hi there!

I hope you are doing well.

I’ve created this guide to help you when working with YAML. You can **use this as a reference document** while using YAML for various use cases at work. It shows you usage, syntax, and examples for each of the YAML language constructs.

I personally refer to the relevant topic of this guide whenever I want to brush up the basics or look at the syntax for a certain YAML construct.

Understanding the language is the first step you need to do when working with YAML. **This downloadable resource is part of The Ultimate YAML Course - Learn YAML from Scratch**, which covers the essentials of YAML that any user **must** know. Click here to know more about the companion course.

See you in the course video!

Praveen.

Table of Contents

Getting Started with YAML

Comments

1. A **single-line comment is entered using the hash symbol '#'**.
   Example:
   ```
   # College Management System
   ```

2. **Multi-line comments aren't supported**. They are entered using multiple single-line comments.
   Example:
   ```
   # College Management System
   # This is a simple system
   # that shows how to manage
   # courses and students
   ```

3. **Recommended:** Leave a space after '#' symbol although it is not mandatory.

Key-Value Pairs

1. Mapping consists of a **key-value pair in the format a:b**
   Example:
   ```
   studentId: 130
   ```

2. **Key's can contain spaces**.
   Example:
   ```
   first Name: Ashok
   ```

3. Keys can be **quoted using both single and double-quotes**. Useful if it contains special characters that require escape sequences. A single quote can contain only a single quote inside. Double quotes can contain C-style escape characters.
   Example:
   ```
   "lastName": Kumar
   'age': 35
   ```

4. **Note:** Keys have to be unique in a given scope. Otherwise, you will get a 'duplicated mapping key' error.

Importance of Indentation

1. Indentation is very important in YAML. The addition of a space can change the entire structure of the element.
2. **YAML structure is determined by indentation**. Indentation defines the flow or a block of data.
3. How to indent?
    a. Generally, two spaces are used.
    b. If the tab key is used, it must be internally converted to spaces by the tool that you are using.

Building Blocks of YAML

Data Types

1. In YAML, the types are referred to as **Language-Independent Types**. Types that are useful across a wide range of applications.
2. Broadly two types:
    a. Scalar Types
    b. Collection Types

Scalar Types

1. Available scalar types:
    a. int
    b. float
    c. string
    d. bool
    e. timestamp
    f. binary

2. Examples

```
studentId: 130
firstName: "Ashok"
lastName: 'Kumar'
age: 35
weight: 70.5
isMale: true
dob: 1995-10-20
registeredDate: 2015-06-25T14:30:00.000Z
```

3. More on timestamp type:
    a. The values **can be date or date time**.
    b. **Supports all major formats** such as Canonical, ISO 8601, space-separated, etc.
    c. Examples
        - Canonical:
          format: YYYY-mm-ddThh:mm:ss.fffZ (time relative to UTC)
          `2020-08-20T02:59:43.1Z`
        - ISO 8601:
          `2020-08-20t17:30:00.10-05:00`
        - Space separated:
          `2020-08-20 17:30:00.10 -5`
        - No time zone (Z):
          `2020-08-20 5:30:00.10`
        - Only date (internally treats it with 00:00:00Z for time):
          `2020-08-20`

Collection Types

1. Types of collections:
    a. map - Unordered set of key: value pairs without duplicates.
    b. omap - Ordered set of key: value pairs without duplicates.
    c. pairs - Ordered sequence of key: value pairs allowing duplicates.
    d. sets - Unordered set of non-equal values.
    e. seq - Sequence of arbitrary values. Eg. arrays/lists.

2. Sequence:
    a. Sequences are analogous to lists or arrays in programming languages.
    b. **Each item is represented using the dash(-)** symbol.
    c. These value items can be scalar, mapping item, complex type, arrays.
    d. Example:
```
 enrolledCourses:
     - 5001
     - 5002
     - 5003
```
3. Mapping:
    a. Mapping **refers to a key-value pair**. Eg. key: value.
    b. Keys can contain spaces, numbers, quotes, etc.
    c. Example:
```
student:
     -
          studentId: 130
```

```
                firstName: "Ashok"
                lastName: 'Kumar'
                age: 35
                weight: 70.5
                isMale: true
                dob: 1995-10-20
                registeredDate: 2015-06-25T14:30:00.000Z
                enrolledCourses:
                      - 5001
                      - 5002
                      - 5003
          -
                studentId: 131
                firstName: James
                lastName: Smith
                age: 35
                weight: 70.5
                isMale: true
                dob: 1995-10-20
                registeredDate: 2015-06-25T14:30:00.000Z
                enrolledCourses:
                      - 5001
                      - 5002
                      - 5003
```

Styles in YAML

1. **Two broad styles/formats**
    a. Block style.
    b. Flow style.

Block Style

1. Block Styles **use indentation**.
2. **Block Scalar Styles:**
    a. The literal style is denoted by the "|" indicator.
    Example:
    ```
    address: |
          1301 Race Street,
          Philadelphia,
          US.
    ```
    b. The folded style is denoted by the ">" indicator that is subject to line folding.

Example:
```
address: >
     1301 Race Street,
     Philadelphia,
     US.
```

3. **Block Collection Styles:**
   a. A block sequence is simply a series of nodes, each denoted by a leading “-” indicator.
   Example:
   ```
   enrolledCourses:
        - 5001
        - 5002
        - 5003
   ```

   b. A Block mapping is a series of entries, each presenting a key: value pair.
   Example:
   ```
   studentId: 131
   firstName: James
   lastName: Smith
   age: 35
   weight: 70.5
   isMale: true
   dob: 1995-10-20
   ```

Flow Style

1. You **use explicit indicators** instead of using indentation to denote scope.

2. Flow Scalar Styles:
   a. Applicable only for string type as other types don't require styling in general.
   b. Strings can be represented using just string literal, single quotes, and double-quotes.
      i. Plain Style - a string without any quotes. This is the most commonly used style.
      ii. Double-quoted Style - a string with double-quotes. The double-quoted style provides escape sequences.
      iii. Single-quoted Style - a string with single quotes. The single-quoted style is useful when escaping is not needed.
      iv. Example:
      ```
      firstName: "Ashok"
      lastName: 'Kumar’
      ```

3. Flow Collection Styles:

a. Flow Sequences - It is comma-separated within square brackets.
   Example:
```
enrolledCourses: [5001, 5002, 5003]
```
b. Flow Mappings - It uses curly braces.
   Example:
```
{ studentId: 132, firstName: Robert, lastName: Smith }
```

YAML Features

Anchors

1. Repeated nodes can be anchored and referenced back throughout the YAML structure. Anchors are also called Aliases.

2. An **anchor is denoted by the "&" indicator and referred back using the '*' symbol.**
   Format:
```
key1: &anchorname value
..
key2: *anchorname
```

3. Usage 1: Duplicate Content

   Example:
```
currentCourse: &curCourse 5001
```

4. Usage 2: Represent Complex Type

   Example:
```
course:
      - 5001: &course5001
            courseId: 5001
            courseName: Computer Science
            courseDuration: 4
            courseType: Engineering

currentCourseDetails: *course5001
```

5. Usage 3: Inherit Properties using Merge Key "<<"

   The "<<" merge key is used to indicate that all the keys of one or more specified maps should be inserted into the current map. Keys in mapping nodes earlier in the sequence override keys specified in later mapping nodes.

Example:
```
hobbycourse:
      - computer hobby:
            <<: *course5001
            isWeekend: true
```

You can reference more than 1 anchor at a time.
Example 1:
```
hobbycourse:
      - computer hobby:
            <<: [*course5001, *course5002]
            isWeekend: true
```

Sets

1. A set is a collection data type that represents a sequence containing items without values.
2. There are three ways to represent a set in YAML.
    a. **Key value pairs.**
    Example:
    ```
    course:
          - 5001: &course5001
                      courseId: 5001
                      courseName: Computer Science
                      courseDuration: 4
                      courseType: Engineering
                      isWeekend: false
                      itemsRequired:
                            laptop: null
                            mobile: null
                            internet: null
    ```
    b. **Only keys.**
    Example:
    ```
    course:
        - 5002:
              courseId: 5002
              courseName: MBA
              courseDuration: 2
              courseType: Management
              itemsRequired:
                     ? laptop
                     ? mobile
                     ? internet
    ```

c. **Flow style**.
Example:
```
hobbycourse:
     - computer hobby:
          <<: *course5001
          isWeekend: true
          itemsRequired: {laptop, mobile, internet}
```

Documents

1. Document represents a group of YAML constructs.
2. Documents start with '---' and optionally end with '...’. Multiple documents are separated with '---'.
   Example:
```
---
# start of a new document
xxx
xxx
---
# start of second document
xxx
xxx
...
```

Directives & Tags

Directives in YAML

Directives are instructions to the YAML processor or tools. The specification defines two directives, “YAML” and “TAG”.
Example:
```
%YAML 1.2
---
key: value
```

YAML Directive

The “%YAML” directive specifies the version of YAML the document conforms to.
Example:
```
%YAML 1.2
---
key: value
```

Tag Directive

1. The TAG directive is used as a shortcut for URI prefixes.
2. Each “TAG” directive associates a handle with a prefix.
3. Format:
   ```
   %TAG handle prefix
   ```

Tag Handles

1. Primary Handle !
   a. The primary tag handle is a single “!” character.
   b. Example:
      ```
      %TAG ! tag:cms.com,2000:
      ---
      course5001: !course { courseId: 5001, courseName:
      Computer Science}
      ```

2. Secondary Handle !!
   a. The secondary tag handle is written as “!!”.
   b. By default, the prefix associated with this handle is “tag:yaml.org,2002:”.
   c. Possible tag values: map, seq, int, str, float, bool, null
   d. Example:
      ```
      %TAG !! tag:yaml.org,2002:
      ---
      dob: !!str 1995-10-20
      ```

3. Named Handle !name!
   a. A named tag handle surrounds a non-empty name with “!” characters.
   b. Example:
      ```
      %TAG !cms! tag:cms.com,2000:app/
      ---
      course5001: !cms!course { courseId: 5001, courseName:
      Computer Science}
      ```

Tag Prefixes

1. Local tag prefix
   a. If the **prefix begins with a “!” character**, shorthands using the handle are expanded to a local tag.
   b. Example:
      ```
      %TAG !c! !cms-
      ---
      ```

```
course5001: !c!course { courseId: 5001, courseName:
Computer Science}

# same as course5001: !cms-course { courseId: 5001,
courseName: Computer Science}
```

2. Global tag prefix
    a. If the **prefix begins with a character other than "!"**
    b. It must be a valid URI prefix.
    c. Example:

```
%TAG !! tag:yaml.org,2002:
---
version: !!str 1.2
# same as version: tag:yaml.org,2002:str 1.2
```

Full YAML Example

```
%YAML 1.2
%TAG !! tag:yaml.org,2002:
---
# College Management System
# This is a simple system
# that shows how to manage
# courses and students

course:
     - 5001: &course5001
          courseId: 5001
          courseName: Computer Science
          courseDuration: 4
          courseType: Engineering
          isFullTime: false
          itemsRequired:
          laptop: null
          mobile: null
          internet: null

     - 5002:
          courseId: 5002
          courseName: MBA
          courseDuration: 2
          courseType: Management
          itemsRequired:
          ? laptop
          ? mobile
          ? internet

hobbycourse:
     - computer hobby:
          <<: *course5001
          isWeekend: true
          itemsRequired: {laptop, mobile, internet}

student:
     -
          studentId: 130
          firstName: "Ashok"
          lastName: 'Kumar'
          age: 35
          weight: 70.5
          isMale: true
          address: |
                    1301 Race Street,
                    Philadelphia,
                    US.
          dob: !!str 1995-10-20
          registeredDate: 2015-06-25T14:30:00.000Z
          enrolledCourses:
          - 5001
          - 5002
          - 5003
          currentCourse: &curCourse 5001
          currentCourseDetails: *course5001
     -
          studentId: 131
          firstName: James
          lastName: Smith
          age: 35
          weight: 70.5
          isMale: true
          address: >
                    1301 Race Street,
                    Philadelphia,
                    US.
          dob: 1995-10-20
          registeredDate: 2015-06-25T14:30:00.000Z
          enrolledCourses: [5001, 5002, 5003]
          currentCourse: *curCourse
          currentCourseDetails:
               <<: *course5001
               isFullTime: true
     -
          { studentId: 132, firstName: Robert, lastName: Smith }
---

staff:
     - 6001:
          id: 6001
          name: James

...
```

YAML Tools & Libraries

The YAML conversion tools can be used when you want to convert from YAML format to another format such as JSON, XML, etc. You can also do the conversion the other way using these tools.

There are various libraries available to work with YAML for most of the popular programming languages.

Online YAML Conversion Tools

1. YAML Viewer (https://jsonformatter.org/yaml-viewer)
2. YAML to JSON (https://onlineyamltools.com/convert-yaml-to-json)
3. YAML to JSON (https://www.convertjson.com/yaml-to-json.htm)

C# Libraries

1. **YamlDotNet** (https://github.com/aaubry/YamlDotNet)
    a. YamlDotNet is a **YAML library for .NET Standard** and other .NET runtimes.
    b. YamlDotNet provides **low level parsing and emitting of YAML** as well as a high level object model similar to XmlDocument. A serialization library is also included that allows to read and write objects from and to YAML streams.

Python Libraries

1. **PyYAML** (https://pyyaml.org/)
2. **ruamel.yaml** (https://pypi.python.org/pypi/ruamel.yaml)

Java Libraries

1. **eo-yaml** (https://github.com/decorators-squad/eo-yaml)

Complete list

YAML Official Website (https://yaml.org)

Thank you!

I hope this resource was helpful to you.