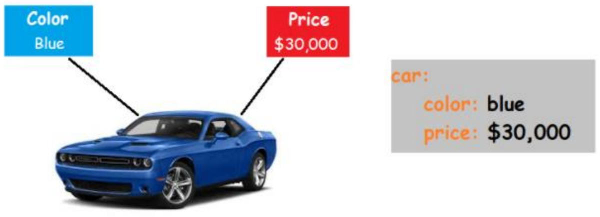
What is YAML?

YAML Ain't Markup Language (abbreviated YAML)

YAML > YAML is designed by Clark Evans, Ingy döt Net, and Oren Ben-Kiki. (2001)

YAML > YAML was meant "Yet Another Markup Language".

- Its goal was another markup language, but YAML was filling the role of a data serialization language later, so it was called YAML Ain't Markup Language, which is a recursive acronym.



What It Is: > YAML is a human-friendly data serialization standard for all programming languages.

- YAML takes concepts from popular languages like Python, Ruby, PHP, and JavaScript. There are familiar terms like scalars, arrays, lists, sequences, and mappings. It also relies on common data structures, just plain indentation, dashes, colons. It is useful to manage data and includes Unicode printable characters.

YAML Code:

```
- name: John
country: US
age: 40

- name: James
country: US
```

Q: What is YAML?
A: YAML is a data serialization format designed for human readability. YAML is a recursive acronym for "YAML Ain't Markup Language".

- Interview Q&A

Attributes of YAML

The design goals for YAML are:

YAML is human readable.
YAML is portable.
YAML works across multiple programming languages easily.
YAML matches the native date of structures, to agile languages.
YAML is consistent and is able to support generic tools.
YAML supports one-pass processing,(when a programming language looks at YAML file, it only needs to go through it once to complete its task.)
YAML is expressive easy to extend and adapt.
YAML should be easy to implement and use.

Basics of YAML

When you are creating a file in YAML, you should remember the following basic rules

- YAML is case sensitive
- The files should have .yaml(or .yml)as the extension
- YAML does not allow the use of tabs while creating YAML files; spaces are allowed instead

⚠Avoid ! :
- YAML does not allow the use of tabs. Spaces are used instead of tabs because tabs are not universally supported.

The key-value is YAML's basic building block. The key is always a string. The value is a scalar so that it can be any data type.

Key-Value Pair

```
Key: Value
```

```
Fruit: Banana
Meat: Chicken
Course: Clarusway
```

A list:

```
- 1
- false
- 'a string'
```

A dictionary:

```
key1: val1
key2: val2
key3: val3
```

A list of dictionaries:

```
Company:
    -   name: Aaron
        age: 35
    -   name: James
        age 35
```

Conventional Block Format uses `hyphen+space` to begin a new item in a specified list.

```
--- # Courses
 - Aws/DevOps
 - Web Developer
 - QA Tester
```

Inline format is delimited with `comma and space` and the items are enclosed in JSON (similar to JSON).

```
--- # Course List
   [Aws/DevOps,  Web Developer, QA Tester]
```

Folded text converts newlines to spaces and removes the leading whitespace.

```
- {name: John MC, age: 18}
- name:James Garcia
  age:19
```

YAML Basic Elements

- Comments in YAML begins with the (`#`) character.

```
# this is single line comment.
```

YAML does not support multi line comments. If you want to provide comments for multiple lines, you can do so as shown below.

```
# this
# is a multiple
# line comment
```

- Comments must be separated from other tokens by whitespaces.
- The indentation of whitespace is used to denote structure.
- Tabs are not included as indentation for YAML files.

- List members are denoted by a leading hyphen (`-`).
- List members are enclosed in square brackets and separated by commas.
- Associative arrays are represented using a colon (`:`) in the format of key-value pairs. They are enclosed in curly braces `{}` .
- Multiple documents with single streams are separated with 3 hyphens (`---`).
- Repeated nodes in each file are initially denoted by an ampersand (`&`) and by an asterisk (`*`) mark later.
- YAML always requires colons and commas used as list separators followed by space with scalar values.
- Nodes should be labeled with an exclamation mark (`!`) or double exclamation mark (`!!`), followed by a string that can be expanded into a URI or URL.

Basics

> All YAML files can optionally begin with `---` and end with `...` This is part of the YAML format and indicates the start and end of a document.

```
---
# A list of courses
- AWS
- DevOps
- Web Developer
...
```

> A dictionary is represented in a simple key: value form (the colon must be followed by a space):

```
# Courses
AWS:
    name: AWS Course
    job: Developer
    level: Advance
```

> More complicated data structures are possible, such as lists of dictionaries, dictionaries whose values are lists or a mix of both:

```
# Employee records
-  john:
     name: John Clarusway
     job: Developer
     skills:
       - python
       - js
-  james:
     name: James Garcia
     job: Developer
     skills:
       - fortran
       - erlang
```

Indentation (Optional)

Indentation and separation are the two main concepts when you are learning any programming language.

Indentation of YAML

> YAML does not include any mandatory spaces. Further, there is no need to be consistent. The valid YAML indentation is shown below

```
a:
   b:
      - c
      -  d
      - e
f:
      "ghi"
```

> There are two styles which we can write our YAML;

Block Styles : The block style is the YAML that you've probably seen before and are somewhat familiar with. It's the less compact and often indented traditional YAML that makes it easy for us humans to look at the file,

- Less Compact
- Better for Humans

```
-
   name: John MC
   hr:   65
   avg:  0.278
-
   name: James Oliver
   hr:   63
   avg:  0.288
```

Flow Styles : The flow styles are an extension of JSON that allows for YAML and JSON to sort of work together. It's less human-readable, but they are sometimes better for the computer that processing your YAML and flow styles rely on folding lines of content.

- An Extension of JSON

```
John MC {hr: 65, avg: 0.278}
Jmaes Oliver: {
    hr: 63,
    avg: 0.288
  }
```

- The flow content of YAML spans multiple lines. The beginning of flow content begins with " { " or " [".
- The block list items include same indentation as the surrounding block level because `-` is considered as a part of indentation.

> > Example:

```
--- !clarusway.com/^invoice
invoice: 20201
date   : 2-3-2020
bill-to: &id002
   given  : Aaron
   family : A
   address:
      lines: |
            112 darken Dr.
            Suite #012
      city    : Royal Oak
      state   : MI
      postal  : 121212
```

Scalars and Tags

> YAML supports some basic data types which can be used with programming languages such as;

- Scalars – strings or numbers or boolean or null.
- Sequences – arrays or lists.
- Mappings – hashes or dictionaries.

> in YAML;

- Scalars are written in block format using a literal type which is denoted as (`|`).
- Scalars are written in folded style (`>`) where each line denotes a folded space which ends with an empty line or more indented line.

```
data: |
   Each of these
   Newlines
   Will be broken up

data: >
   This text is
   wrapped and will
   be formed into
   a single paragraph
```

Strings

```
A string in YAML

'A singled-quoted string in YAML'

"A double-quoted string in YAML"
```

Number

```
# an integer
12

# an octal
014

# a hexadecimal
0xC

# afloat
13.4

# infinity
.inf
```

Others

- **Nulls** : Nulls in YAML can be expressed with `null` or `~`.
- **Booleans** : Booleans in YAML are expressed with `true` and `false` .
- **Dates** : YAML uses the ISO-8601 standard to express dates.

```
2020-03-15T21:59:43.10-05:00

# simple date
2020-03-15
```

Collections and Structures

Collections

- YAML includes block collections that use indentation for scope. Block sequences indicate each entry with a dash and space (`-`).
- Mappings use `key-value` pair representation with the usage of `colon and space (: )` .

Example of Sequence:

```
- John
- James
- Aaron
- Oliver
- Walter
```

Example of Mapping:

```
hr: 160
avg: 0.298
rbi: 153
```

Example of Mapping to Sequence:

```
AWS-DevOps:
- Linux
- AWS
- Docker

Developer:
- Html
- Css
- js
```

Example of Sequence of Mappings:

```

-
  name: Aaron
  age: 35
  job: Developer
-
  name: Oliver
  age: 36
  job: Developer
```

Example of Sequence of Sequences:

```
- [name  , age, job  ]
- [Aaron , 35  , Developer]
- [Oliver , 36  , Developer]
```

Example of Mapping of Mappings:

```
Aaron: {age: 35,  job: Developer}
Oliver: {
   age: 36,
   job: Developer
  }
```

Structures

- YAML uses three dashes (`---`) to separate directives from document content. This also serves to signal the start of a document if no directives are present. Three dots (`...`) indicate the end of a document without starting a new one, for use in communication channels.
- Repeated nodes (objects) are first identified by an anchor (marked with the ampersand - `&`) , and are then aliased (referenced with an asterisk `*`) thereafter.

```
person:
  name: &name "Oliver"
  occupation: 'programmer'
  age: 33

id: *name #Oliver
```

- A question mark and space (`?`) indicate a complex mapping key. Within a block collection, key: value pairs can start immediately following the dash, colon, or question mark.

```
? - Detroit Tigers
  - Chicago cubs
:
  - 2020-03-09

? [ New York Yankees,
    Atlanta Braves ]
: [ 2020-03-09, 2020-02-12,
    2020-01-14 ]
```

Full Length Example

- The following full-length example specifies the construct of YAML which includes symbols.

Example:

```
---
Time: 2020-03-09 15:01:42 -5
User: ed
Warning:
  This is an error message
  for the log file
---
Time: 2020-03-09 15:02:31 -5
User: ed
Warning:
  A slightly different error
  message.
---
Date: 2020-03-09 15:03:17 -5
User: ed
Fatal:
  Unknown variable "bar"
Stack:
  - file: TopClass.py
    line: 23
    code: |
      x = MoreObject("345\n")
  - file: MoreClass.py
    line: 58
    code: |-
      foo = bar
```