Rhymes with "camel"

What Is YAML?

YAML was born in 2001 quirkly named as "Yet Another Markup Language".

Today it has morphed rebelliously into "YAML Ain't Markup Language".

A YAML file's basic structure is a map or you could call it a dictionary or object. Your choice.

```yaml
version 3.1

command: ["sh", "-c", "command1", "command2"]

docker:
  - image: alpine
  - image: mongo:2.8
    command: [mongod, --smallfile]
  - image: postgres:14.2
```

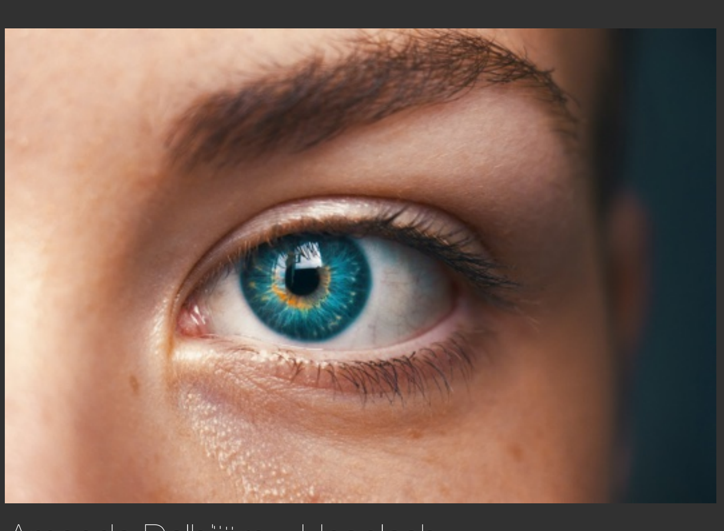
Key Value

You can use all kinds of types for values: numbers, booleans and the ever popular strings.

Python-style indentation indicates nesting and […] for lists and {…} for maps. JSON is valid YAML. Are we having fun yet?

Tab characters are not valid in YAML so you need to have your editor substitute spaces to indent. Use an online YAML validator to check for syntax errors: www.yamllint.com/

As you can see, YAML is a human-readable data serialization format. It can be used for configuration files and data exchange between languages with different data structures.



Amanda Dalbjörn - Unsplash

Philip Mercier
The Shutz Family and their Friends on a Terrace (1725)

If I stop being fancy, its a text file that tells a computer what to do.

Yeah, but what are some YAML Use Cases?

DevOps and CI/CD

Docker, Kubernetes and CI/CD pipelines use YAML for specifying deployment configurations, describe workflow processes and provide build instructions.

Application Configuration

YAML can specify your application settings such as database connection parameters, API keys or environment variables.

Data Exchange

YAML serializes data to exchange between different programming languages by representing data in a standard format for all to consume.

Infrastructure as Code

YAML can define infrastructure configurations to help automate infrastructure provisioning and management.

We hope you have enjoyed this short tour of YAML and have learned a little bit more about this useful technology. We have provided a cheat sheet below for you to better understand it's power.

YAML Cheat Sheet

```yaml
%YAML 1.1   # Reference card
---
Collection indicators:
  '? ' : Key indicator.
  ': ' : Value indicator.
  '- ' : Nested series entry indicator.
  ', ' : Separate in-line branch entries.
  '[]' : Surround in-line series branch.
  '{}' : Surround in-line keyed branch.
Scalar indicators:
  '''' : Surround in-line unescaped scalar ('' escaped ').
  '"'  : Surround in-line escaped scalar (see escape codes below).
  '|'  : Block scalar indicator.
  '>'  : Folded scalar indicator.
  '-'  : Strip chomp modifier ('|-' or '>-').
  '+'  : Keep chomp modifier ('|+' or '>+').
  1-9  : Explicit indentation modifier ('|1' or '>2').
         # Modifiers can be combined ('|2-', '>+1').
Alias indicators:
  '&'  : Anchor property.
  '*'  : Alias indicator.
Tag property: # Usually unspecified.
  none    : Unspecified tag (automatically resolved by application).
  '!'     : Non-specific tag (by default, "!!map"/"!!seq"/"!!str").
  '!foo'  : Primary (by convention, means a local "!foo" tag).
  '!!foo' : Secondary (by convention, means "tag:yaml.org,2002:foo").
  '!h!foo': Requires "%TAG !h! <prefix>" (and then means "<prefix>foo").
  '!<foo>': Verbatim tag (always means "foo").
Document indicators:
  '%'  : Directive indicator.
  '---': Document header.
  '...': Document terminator.
Misc indicators:
  ' #' : Throwaway comment indicator.
  '`@' : Both reserved for future use.
Special keys:
  '='  : Default "value" mapping key.
  '<<' : Merge keys from another mapping.
Core types: # Default automatic tags.
  '!!map' : { Hash table, dictionary, mapping }
  '!!seq' : { List, array, tuple, vector, sequence }
  '!!str' : Unicode string
More types:
  '!!set' : { cherries, plums, apples }
  '!!omap': [ one: 1, two: 2 ]
Language Independent Scalar types:
  { ~, null }              : Null (no value).
  [ 1234, 0x4D2, 02333 ]   : [ Decimal int, Hexadecimal int, Octal int ]
  [ 1_230.15, 12.3015e+02 ]: [ Fixed float, Exponential float ]
  [ .inf, -.Inf, .NAN ]    : [ Infinity (float), Negative, Not a number ]
  { Y, true, Yes, ON  }    : Boolean true
  { n, FALSE, No, off }    : Boolean false
  ? !!binary >
      R0lG...BADS=
  : >-
      Base 64 binary value.
Escape codes:
 Numeric   : { "\x12": 8-bit, "\u1234": 16-bit, "\U00102030": 32-bit }
 Protective: { "\\": '\', "\"": '"', "\ ": ' ', "\<TAB>": TAB }
 C         : { "\0": NUL, "\a": BEL, "\b": BS, "\f": FF, "\n": LF, "\r": CR,
               "\t": TAB, "\v": VTAB }
 Additional: { "\e": ESC, "\_": NBSP, "\N": NEL, "\L": LS, "\P": PS }
...
```

Indest Labs