YAML Fundamentals

YAML Basic Syntax

Maaike van Putten
Software Developer & Trainer

www.brightboost.nl

Version Check

This version was created by using:

- YAML 1.2.2 (October 2021)

Overview

What YAML is

Basic syntax

- Indentation
- Sequences / lists
- Mapping / dictionary
- Scalars
- Comments
- Documents

YAML

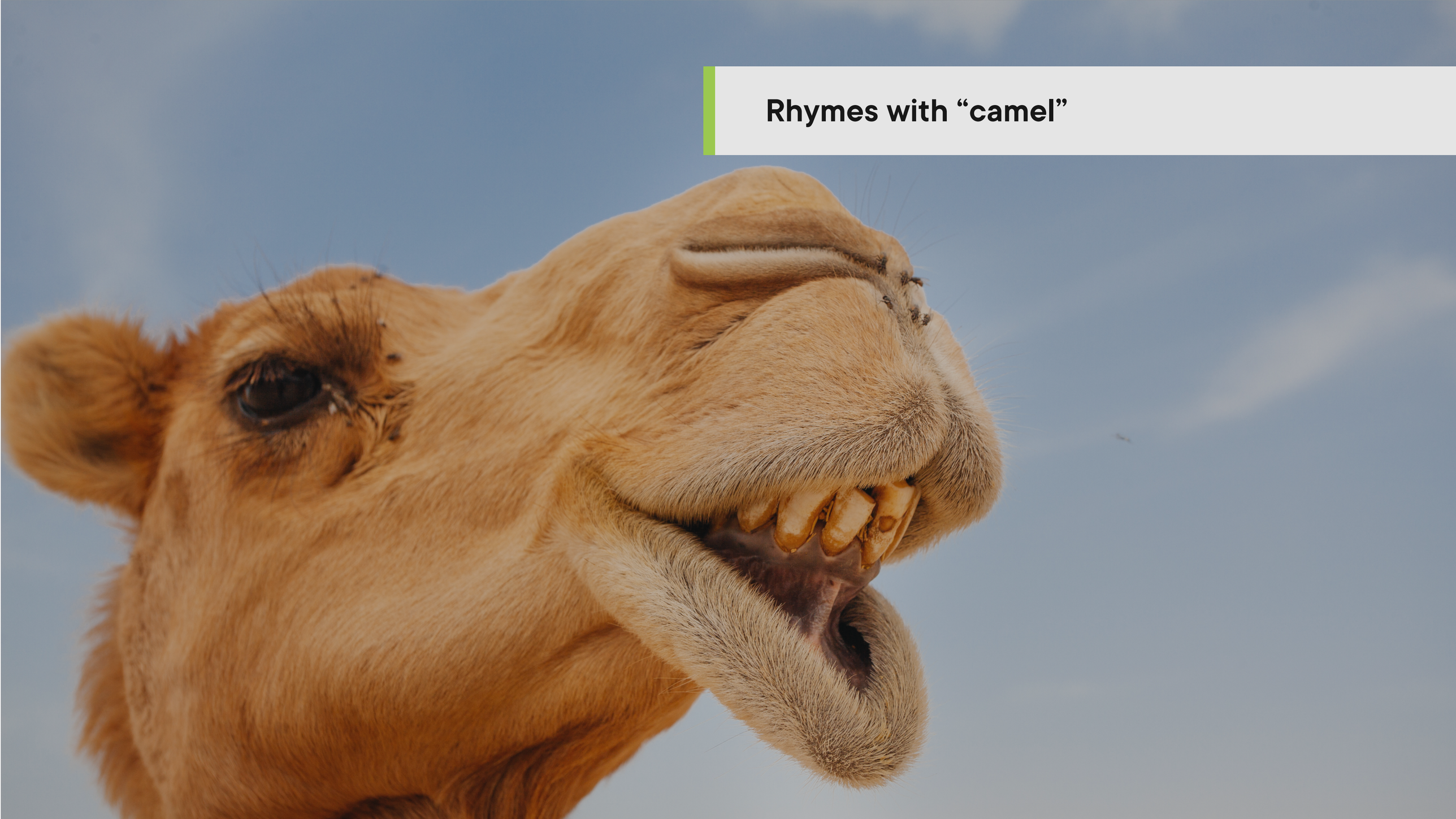
Yet Another Markup Language.

YAML

~~Yet Another Markup Language.~~
YAML Ain’t Markup Language.

Show docs

Rhymes with “camel”



What Is YAML?

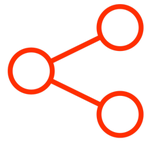
Data serialization language

Unicode printable characters

Increasingly popular over the last few years

Often used for data about hosts and infrastructure

YAML Use Cases



Cross-language data sharing

Configuration files

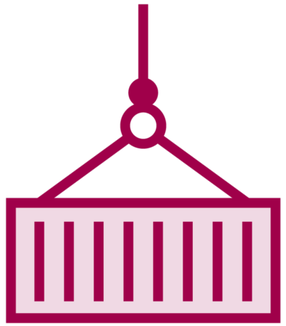
Log files

Object persistence

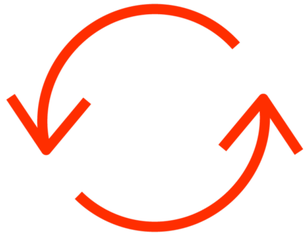
Working with languages such as PHP, JS, Python, Perl and Ruby

YAML Features

Human readable

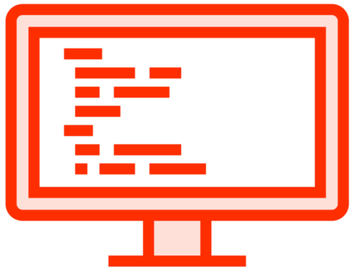


Portable

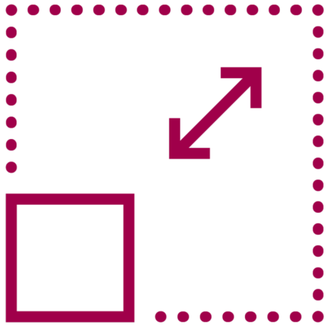


Consistent

Match native data structure



Expressive



Extensible

YAML and JSON

JSON

Data interchange format focused on being interchangeable and easy to process for programming languages.

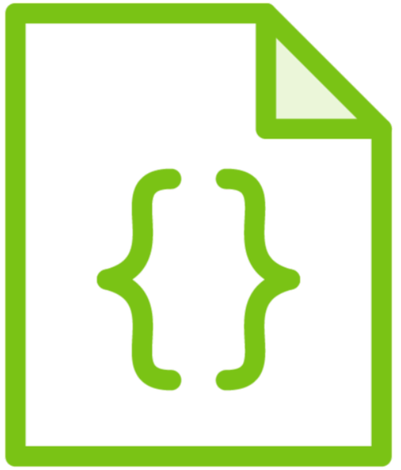
YAML

Superset of JSON.
Data interchange format. Focus on being human readable and supporting native structures.

Block vs Flow Style

Block style

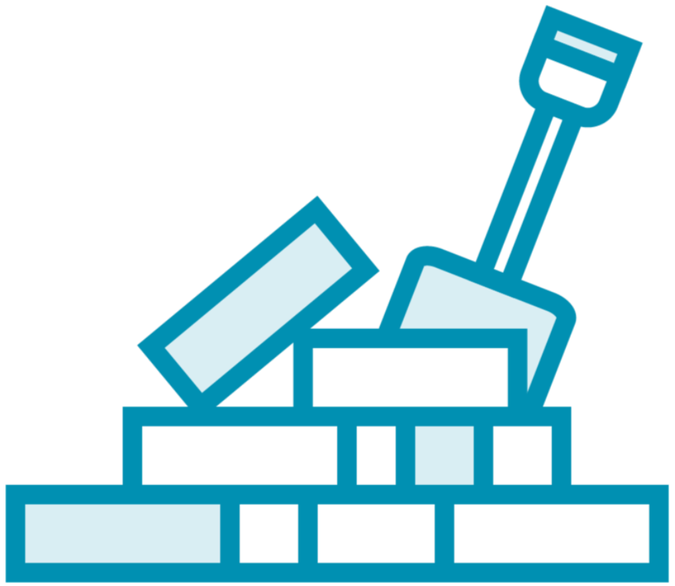
YAML's format, indentation to give the data structure.



Flow style

JSON style format, using characters as explicit indicators to indicate structure.

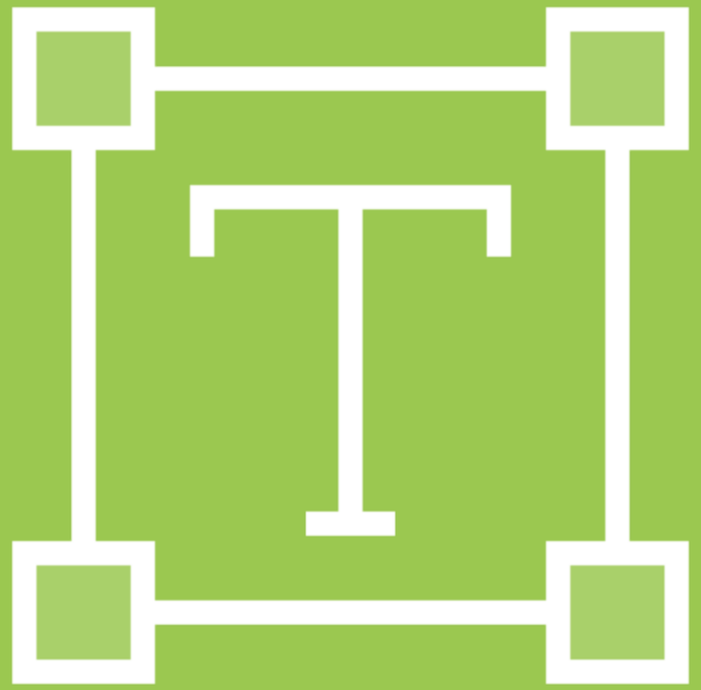
YAML Building Blocks



Sequence

Mapping

Scalar



Indentation

Indentation is done with one or more spaces, but never with tabs.

Sequence

Arrays / lists

Different styles:

- Block style
- Flow style

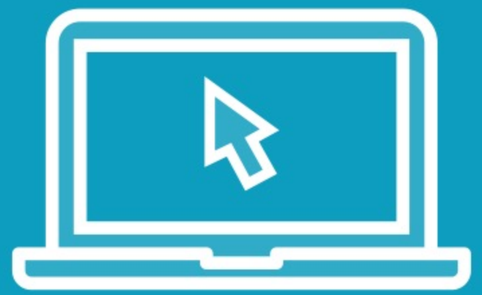
YAML Sequence Block Style

```
- first item in the list
- second item in the list
- third item in the list
```

YAML Sequence Flow Style

[just, a list, in flow style]

Demo

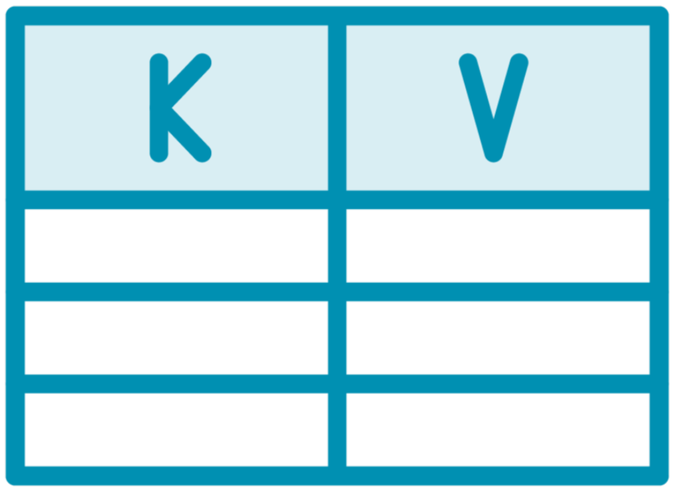


Sequence

Block style: using indentation and –

Flow style: using [ ] and ,

Mapping



Key-value pair

Indicated by : (colon and space)

Alternative terms: dictionary, hash, key-value pair

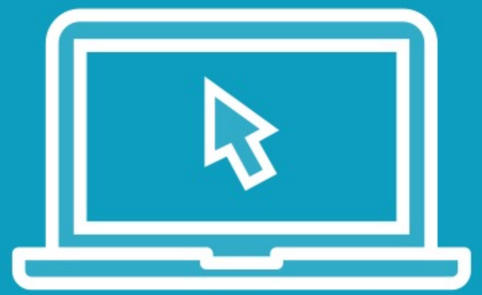
YAML Mapping Block Style

```
key: value
list:
 - first item
 - second item
 - third item
flowlist: [just, a list, in flow style]
```

YAML Mapping Flow Style

{ key: value, another: value, flowlist: [just, a list, in flow style] }

Demo

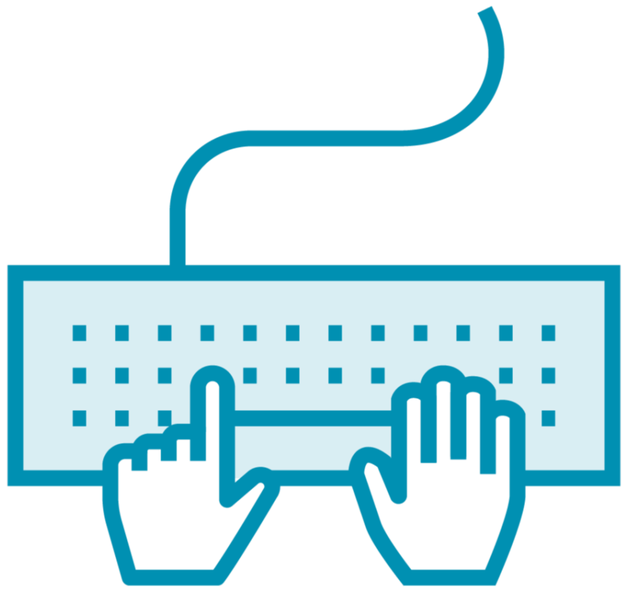


Mapping

Block style: using colon and space

Flow style: using { }, : and ,

Scalar



Basic values: strings, numbers, booleans, dates

Different styles:

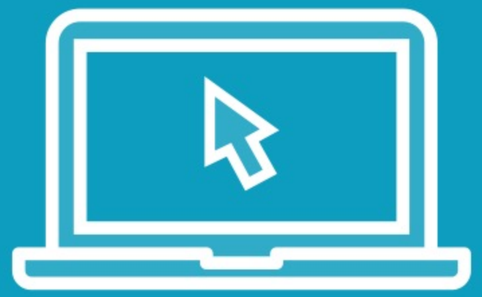
- Flow scalar
- Block scalar

**Escaping with **

YAML Scalar

```
tool: yaml
version: 1.2
awesome: true
nothing: null
duplicateKeys: not allowed
```

Demo



Basic scalars

For different data types

- String
- Numeric
- Date
- Boolean

Comment

Presentation detail

No effect on serialization tree

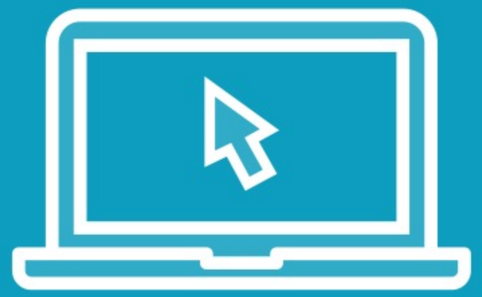
Meant for humans

Start with a “#”

YAML Comment

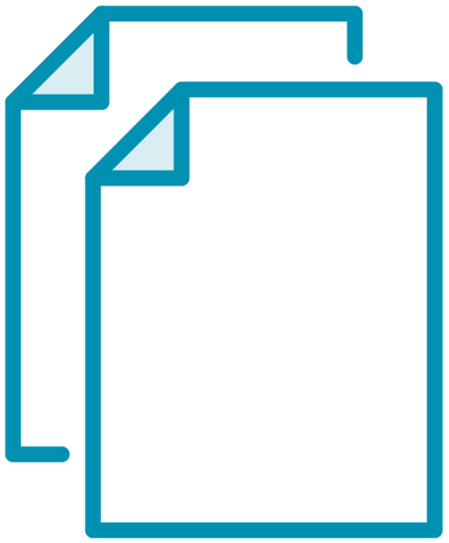
```
# this is a comment
# it will be lost during parsing
```

Demo



Comments

Document



One file can contain multiple documents

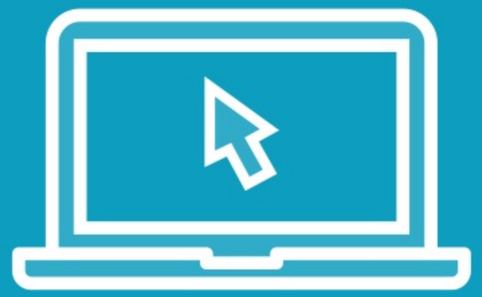
Documents are separated by 3 hyphens (---)

Can be ended with suffix of 3 dots (...)

YAML Document

```
tool: yaml
version: 1.2
awesome: true
nothing: null
duplicateKeys: not allowed
---
This is a new document
---
One file can contain multiple documents
```

Demo



Multi documents

Summary

- **Sequences**
- **Mappings**
- **Scalars**
- **Comments**
- **Documents**

Up Next:
YAML Advanced Syntax

YAML Advanced Syntax

Maaike van Putten
Software Developer & Trainer

www.brightboost.nl

Overview

- Folding and chomping
- Nested sequences
- Nested mappings
- Combining sequences and mappings
- Dates in markdown files
- Repeated nodes
- Adding data types

Multi-line Strings: Folding and Chomping

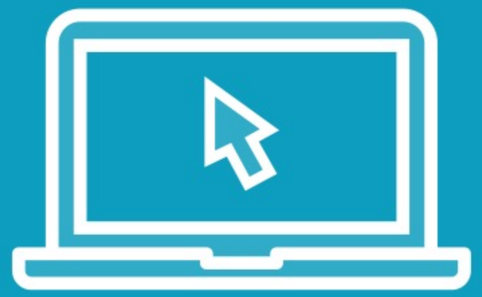
Folding

Dealing with new lines in multi-line strings

Chomping

How to deal with trailing newlines in multi-line strings

Demo



https://yaml-multiline.info/

Nested Sequences

- [1, 2, 3]
- [A, B, C]

Sequence of sequences

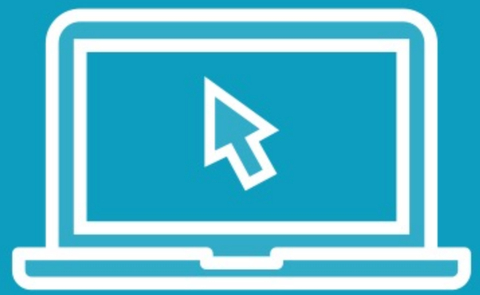
Combining different styles possible

YAML Nested Sequences

```yaml
flow_list:
 - [of, lists]
 - [and, lists]

block_list:
 -
   - of
   - lists
 -
   - and
   - lists
```

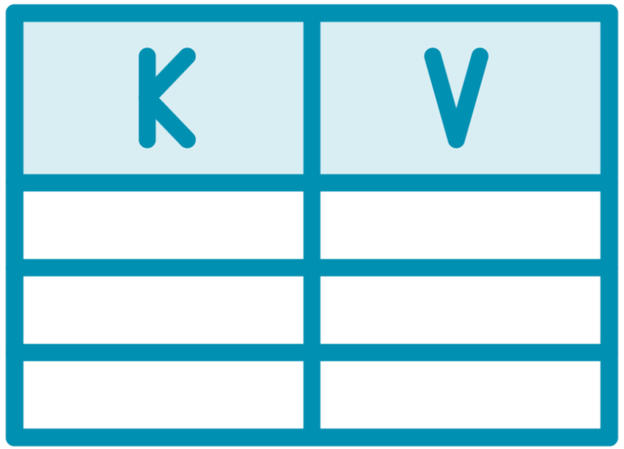
Demo



Sequence of sequence

Different styles

Nested Mappings



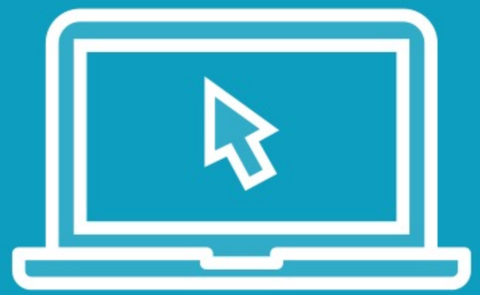
Mapping of mappings

Combining different styles possible

YAML Nested Mappings

```yaml
person:
  name: Maaike
  age: 30
  address:
    streetname: Langstraat
    number: 1
    zipcode: 1234AB
    country: The Netherlands
  flow-mapping: { where: inside, what: mapping }
```

Demo



Mapping of mapping

Different styles

Combining Sequences and Mappings

Map scalar to sequence

Sequence of mappings

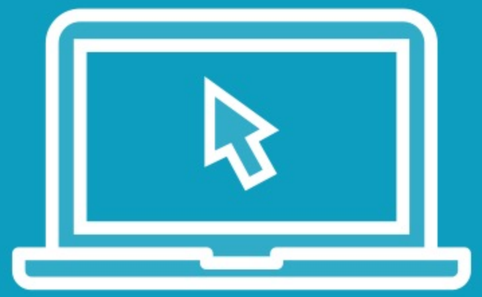
Mapping of sequence of mappings

Et cetera

YAML Combining Sequences and Mappings

```yaml
languages:
  - programming:
      - frontend:
          - html
          - css
          - js
      - backend:
          - java
          - python
          - c#
  - data serialization:
      - yaml
      - json
```

Demo



Map scalar to sequence

Sequence of mappings

Mapping of sequence of mappings

Dates

Scalar

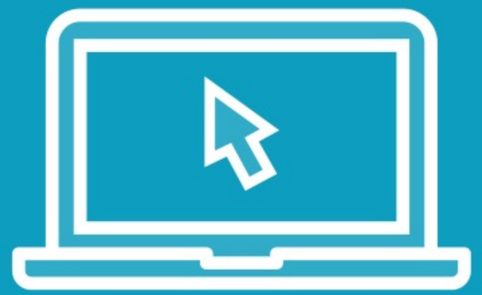
Date and timestamp

Explicit tag to turn date into string

YAML Dates

timestamp: 2022-03-22T22:19:56.10+02:00
simple-date: 2022-03-22

Demo

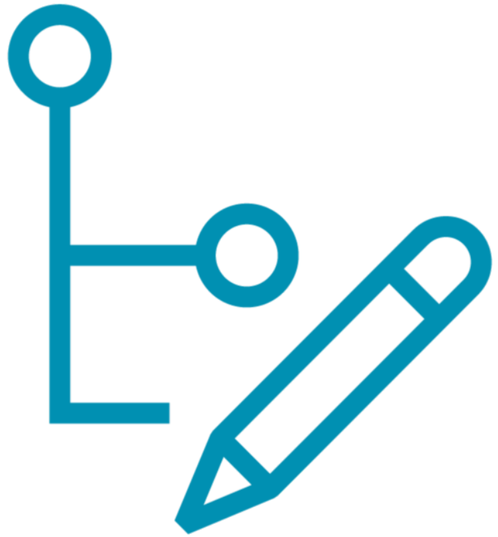


Date

Timestamp

Explicit string tag

Repeated Nodes



Cleaner YAML: DRY

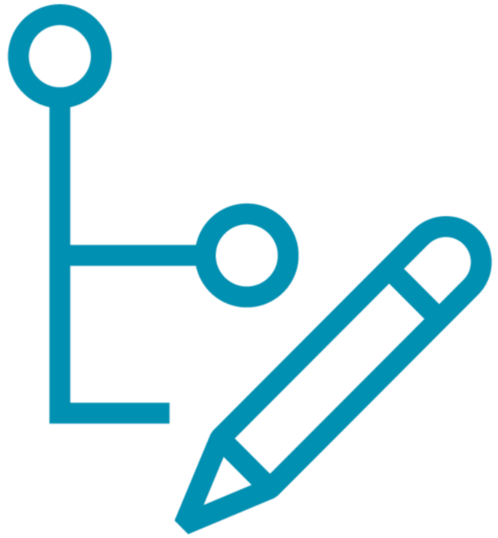
Anchor name (&name) and next referenced with alias (*name)

YAML Repeated Nodes

```yaml
value: &repeated Hi there
new_value: *repeated

person: &person
  name: Maaike
  age: 30
  address:
    streetname: Langstraat
    number: 1
    zipcode: 1234AB
    country: The Netherlands
another_person: *person
```

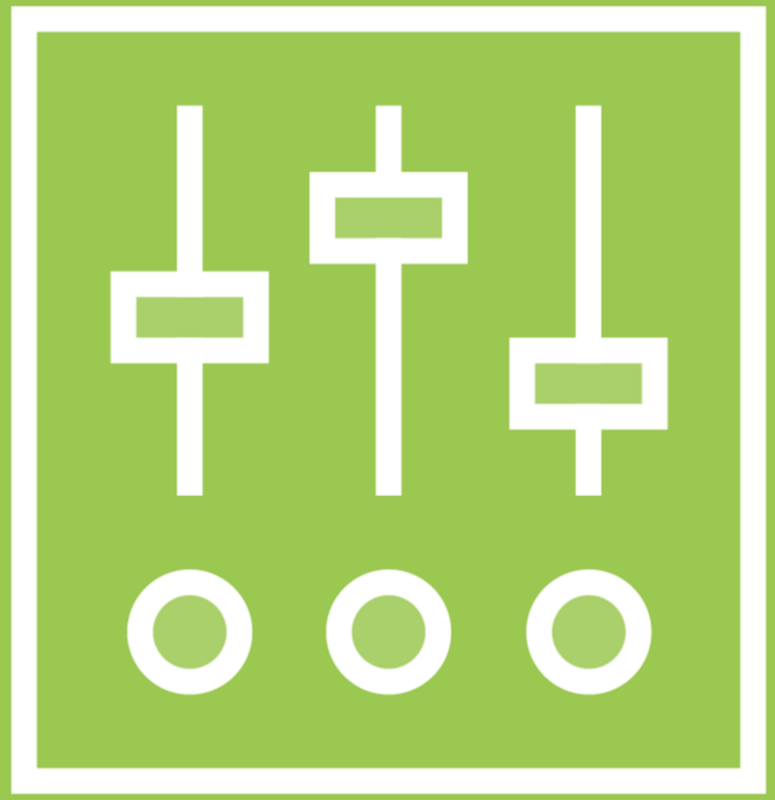
Repeated Nodes



Cleaner YAML: DRY

Anchor name (&name) and next referenced with alias (*name)

Overwrite part of the repeated node with merge key <<:



What if You'd Want to Modify the Alias?

Overwrite part of the repeated node with merge key <<:

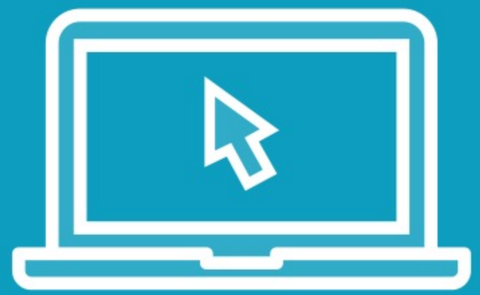
YAML Repeated Nodes Merge

```yaml
person: &person
  name: Maaike
  age: 30
  address:
    streetname: Langstraat
    number: 1
    zipcode: 1234AB
    country: The Netherlands

another_person: *person

yet_another_person:
  <<: *person
  name: maria
```

Demo



- Need for repeated nodes
- Simple repeated node
- Repeated block
- Overwrite part of the alias

Working with Tags

Tags give a node a type

Untagged nodes are given a type

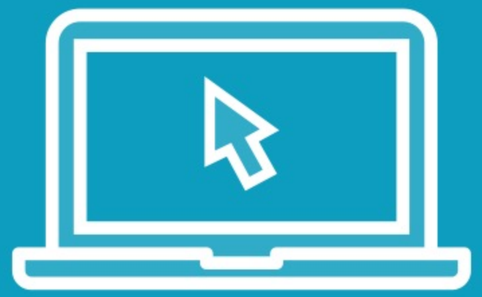
Explicit typing can be done with the !

Schemas specify the tags

YAML Working with Tags

```
not-a-date: !!str 2022-03-21
not-a-number: !!str 20
unnecessary: !!int 20
```

Demo



Comments

Working with Schemas

Default: Failsafe schema, JSON schema and core schema

Used to create custom data types with language specific schemas

Programming languages have their own schema that map YAML to the language native data structure

Interoperable schemas use global tags to represent the same data across different languages

Default Schemas

Failsafe Schema

Three tags: string, mapping, sequence

!!str: a string value

!!map: a mapping

!!seq: a sequence

!! Is shorthand for
!<tag:yaml.org,2002:...>

JSON Schema

Failsafe tags

!!null: a null value

!!bool: boolean

!!int: integer

!!float: float

Core Schema

Extension of the JSON schema

Same tags as the JSON schema, but extended tag resolution

Recommended default schema

Recommended to base custom schemas on this one

Up Next:
Parsing and Validation

Parsing and Validation

Maaike van Putten
Software Developer & Trainer

www.brightboost.nl

Overview

Parsing

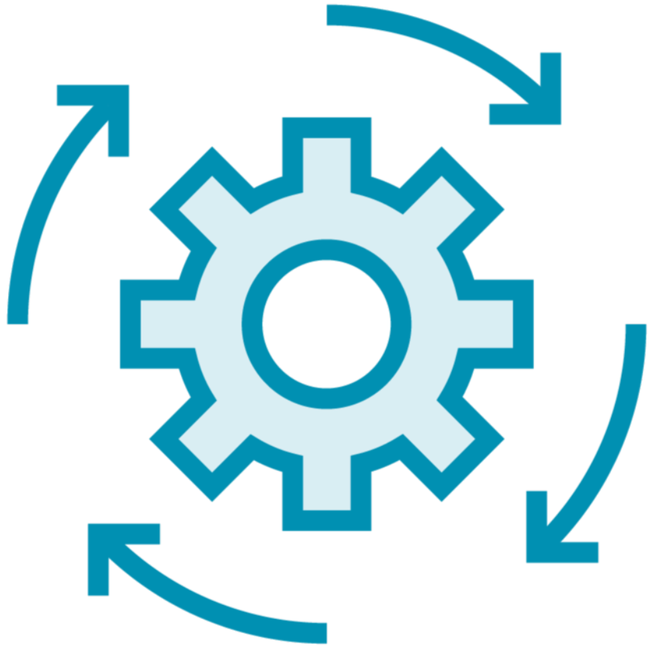
Validating

Parsing and validating YAML with Python and Java

Debugging YAML

Common YAML mistakes

Parsing



Reading text and converting it

Outcome could for example be objects in the memory

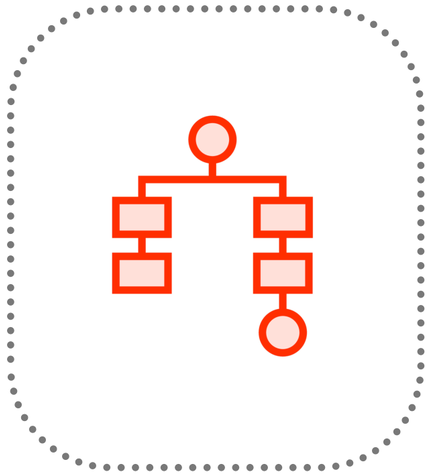
Converts the text in a YAML file to native data structures

Processing YAML

YAML

Magic

Native data structure

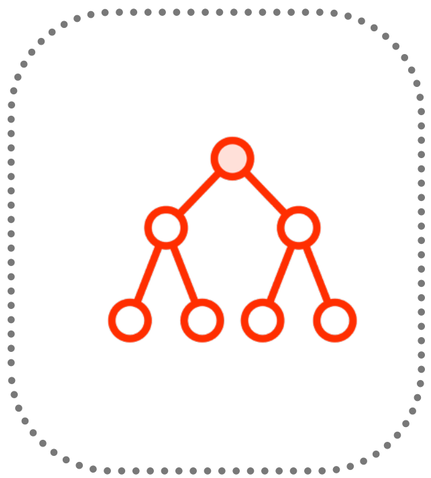


Processing YAML

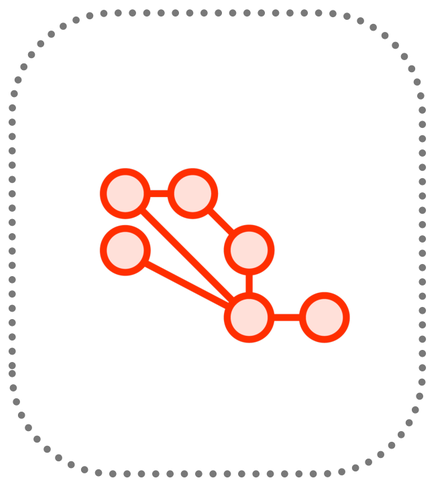
Load into application

YAML - character stream

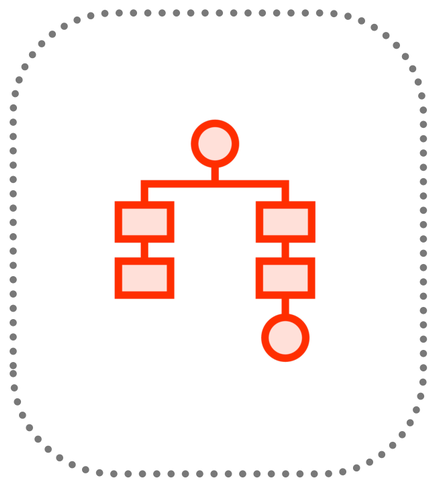
Serialization – event tree



Representation – node graph



Native data structure

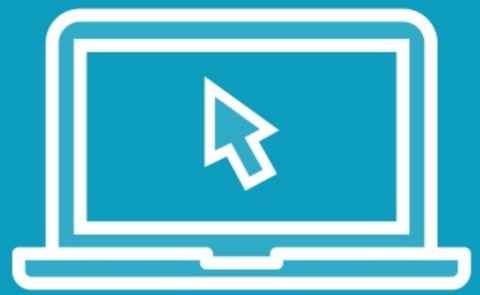


Dump (to YAML)

Validating YAML

- Making sure the syntax of the content is valid
- Checking against formatting rules of YAML
- Find problems with certain parts of the file
- External tools needed

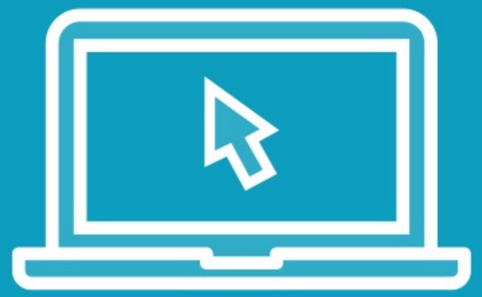
Demo



Validating YAML with Java

Parsing YAML with Java

Demo

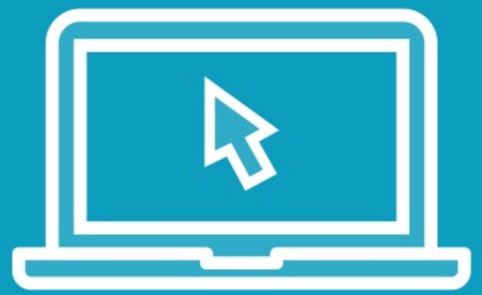


Validating YAML with Python

Parsing YAML with Python

Custom data types and Python

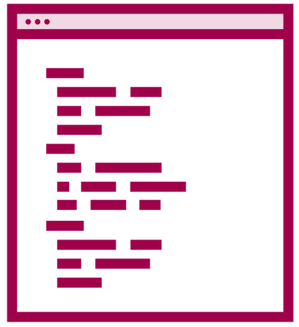
Demo



Debugging a YAML file

Common Errors

Quotes



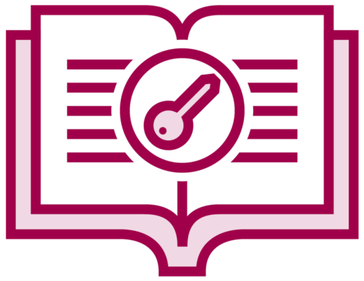
Indentation

Number type

Duplicate keys

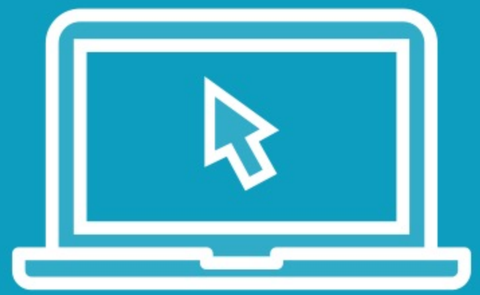
[1,2,3]

Accidental list entry



Lists as keys in languages that don't allow this

Demo



Spotting the common errors

Solving the common errors

Up Next:
YAML in Practice

YAML in Practice

Maaike van Putten
Software Developer & Trainer

www.brightboost.nl

Overview

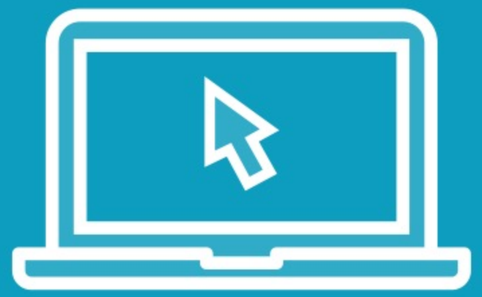
YAML in everyday life

- Kubernetes
- Docker Compose
- Pipeline tools

Compare YAML and JSON

Compare YAML and XML

Demo



Unscramble Kubernetes config file

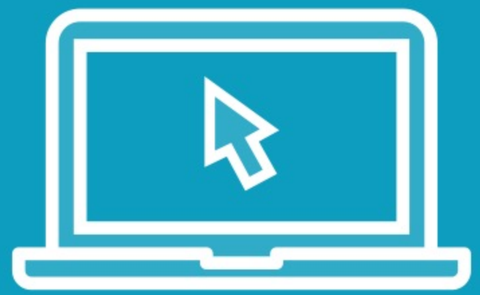
YAML vs JSON

YAML	JSON
Can be parsed with a YAML	Can be parsed with a YAML parser
Comments with a #	Comments not allowed
Objects and lists are denoted with indentation or {} and []	Objects and lists are denoted with {} and []
String quotes are optional, can be double or single	String quotes are mandatory, must be double quotes
Root node can be any valid data type	Root node must be object or list
Standard for configuration	Standard for APIs

YAML vs XML

YAML	XML
Data serialization language	Markup language
Easier to read	Harder to read
Querying YAML relies on many different external tools	Well established options to query data in XML file
Typically, preferred option for configuration	Decreasing in popularity because of JSON and YAML

Demo



Have a look at three files containing the same data

- JSON
- YAML
- XML

Summary

- YAML usage
- Basic building blocks
- Parsing YAML
- YAML in the wild

Good luck on your YAML journey!