YAML

http://www.yaml.org/

국민대학교 컴퓨터공학부

강 승 식

YAML: YAML Ain't Markup Language

- Yet Another Markup Language? '가벼운 마크업 언어'로 사용

- YAML is a human friendly data serialization standard for all programming languages.
- '사람이 쉽게 읽을 수 있는' 데이터 직렬화 양식.

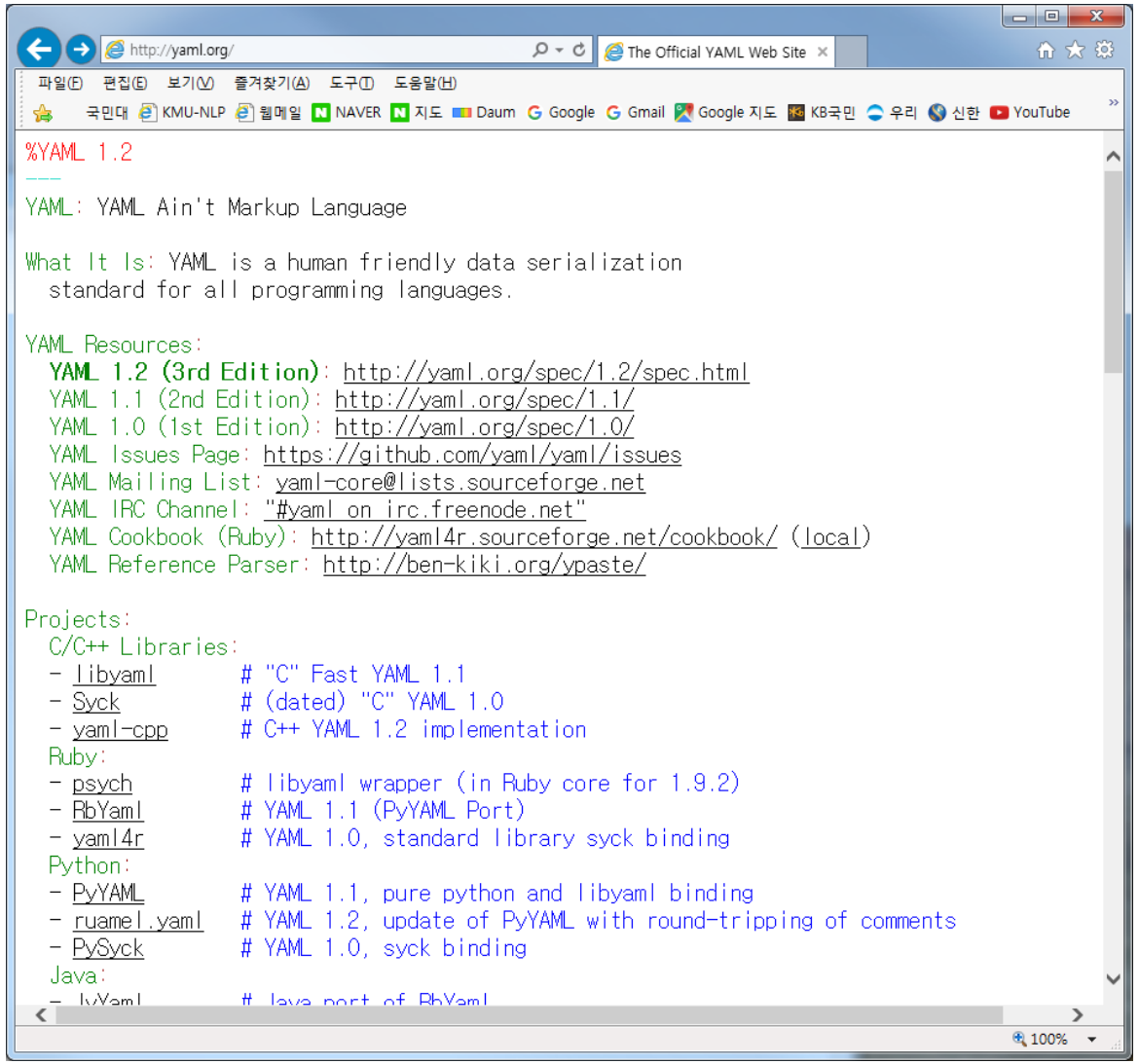
- 2001년, 클라크 에반스: Ingy dot Net, Oren Ben-Kiki
- XML, C, 파이썬, 펄, RFC2822에서 정의된 e-mail 양식에서...

- The purpose of YAML
  - represent typical data types in human-readable notation
- YAML is (almost) a superset of JSON
  - Many more capabilities (lists, casting, etc.)
  - Except: YAML doesn’t handle escaped Unicode characters
  - Consequently, JSON can be parsed by YAML parsers
- When JSON isn’t enough, consider YAML

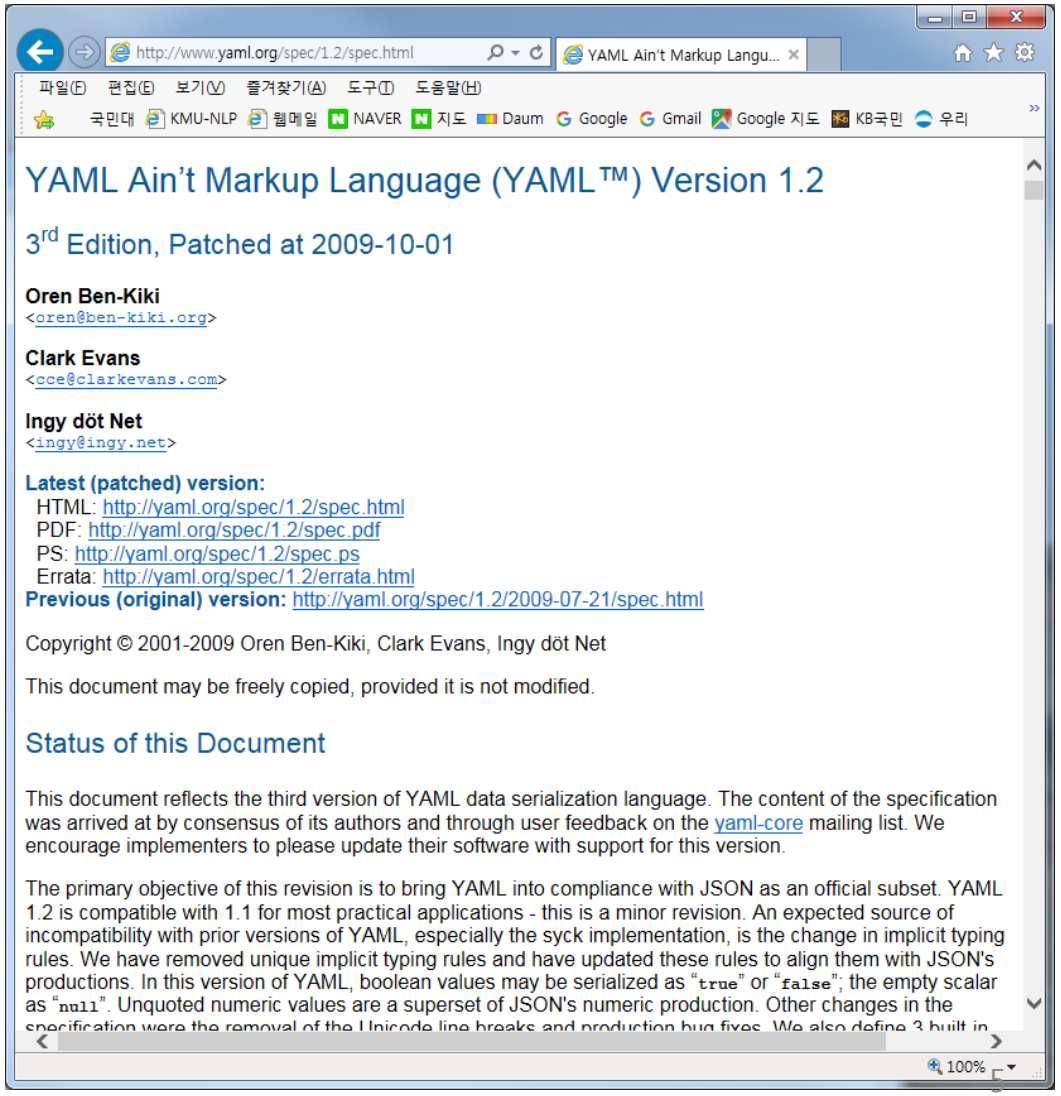
YAML Resources

- **YAML 1.2 (3rd Edition)**: http://yaml.org/spec/1.2/spec.html
- YAML 1.1 (2nd Edition): http://yaml.org/spec/1.1/
- YAML 1.0 (1st Edition): http://yaml.org/spec/1.0/
- YAML Issues Page: https://github.com/yaml/yaml/issues
- YAML Mailing List: yaml-core@lists.sourceforge.net
- YAML IRC Channel: "#yaml on irc.freenode.net"
- YAML Cookbook (Ruby): http://yaml4r.sourceforge.net/cookbook/ (local)
- YAML Reference Parser: http://ben-kiki.org/ypaste/

YAML 1.2


http://yaml.org/
The Official YAML Web Site
%YAML 1.2

YAML: YAML Ain't Markup Language
What It Is: YAML is a human friendly data serialization
standard for all programming languages.
YAML Resources:
YAML 1.2 (3rd Edition): http://yaml.org/spec/1.2/spec.html
YAML 1.1 (2nd Edition): http://yaml.org/spec/1.1/
YAML 1.0 (1st Edition): http://yaml.org/spec/1.0/
YAML Issues Page: https://github.com/yaml/yaml/issues
YAML Mailing List: yaml-core@lists.sourceforge.net
YAML IRC Channel: "#yaml on irc.freenode.net"
YAML Cookbook (Ruby): http://yaml4r.sourceforge.net/cookbook/ (local)
YAML Reference Parser: http://ben-kiki.org/ypaste/
Projects:
C/C++ Libraries:
- libyaml # "C" Fast YAML 1.1
- Syck # (dated) "C" YAML 1.0
- yaml-cpp # C++ YAML 1.2 implementation
Ruby:
- psych # libyaml wrapper (in Ruby core for 1.9.2)
- RbYaml # YAML 1.1 (PyYAML Port)
- yaml4r # YAML 1.0, standard library syck binding
Python:
- PyYAML # YAML 1.1, pure python and libyaml binding
- ruamel.yaml # YAML 1.2, update of PyYAML with round-tripping of comments
- PySyck # YAML 1.0, syck binding
Java:


http://www.yaml.org/spec/1.2/spec.html
YAML Ain't Markup Langu...
YAML Ain't Markup Language (YAML™) Version 1.2
3rd Edition, Patched at 2009-10-01
Oren Ben-Kiki
<oren@ben-kiki.org>
Clark Evans
<cce@clarkevans.com>
Ingy döt Net
<ingy@ingy.net>
Latest (patched) version:
HTML: http://yaml.org/spec/1.2/spec.html
PDF: http://yaml.org/spec/1.2/spec.pdf
PS: http://yaml.org/spec/1.2/spec.ps
Errata: http://yaml.org/spec/1.2/errata.html
Previous (original) version: http://yaml.org/spec/1.2/2009-07-21/spec.html
Copyright © 2001-2009 Oren Ben-Kiki, Clark Evans, Ingy döt Net
This document may be freely copied, provided it is not modified.
Status of this Document
This document reflects the third version of YAML data serialization language. The content of the specification was arrived at by consensus of its authors and through user feedback on the yaml-core mailing list. We encourage implementers to please update their software with support for this version.
The primary objective of this revision is to bring YAML into compliance with JSON as an official subset. YAML 1.2 is compatible with 1.1 for most practical applications - this is a minor revision. An expected source of incompatibility with prior versions of YAML, especially the syck implementation, is the change in implicit typing rules. We have removed unique implicit typing rules and have updated these rules to align them with JSON's productions. In this version of YAML, boolean values may be serialized as "true" or "false"; the empty scalar as "null". Unquoted numeric values are a superset of JSON's numeric production. Other changes in the

- Collections
- Structures
- Scalars
- Tags

1. Collections

- Block sequences indicate each entry
  - with a dash and space (“- ”)
- Mappings
  - use a colon and space (“: ”) to mark each key: value pair
- Comments begins with ‘#’

YAML's block collections use indentation for scope and begin each entry on its own line. Block sequences indicate each entry with a dash and space ("- "). Mappings use a colon and space (": ") to mark each key: value pair. Comments begin with an octothorpe (also called a "hash", "sharp", "pound", or "number sign" - "#").

Example 2.1. Sequence of Scalars (ball players)

```
- Mark McGwire
- Sammy Sosa
- Ken Griffey
```

Example 2.2. Mapping Scalars to Scalars (player statistics)

```
hr:  65    # Home runs
avg: 0.278 # Batting average
rbi: 147   # Runs Batted In
```

Example 2.3. Mapping Scalars to Sequences (ball clubs in each league)

```
american:
  - Boston Red Sox
  - Detroit Tigers
  - New York Yankees
national:
  - New York Mets
  - Chicago Cubs
  - Atlanta Braves
```

Example 2.4. Sequence of Mappings (players' statistics)

```
-
  name: Mark McGwire
  hr:   65
  avg:  0.278
-
  name: Sammy Sosa
  hr:   63
  avg:  0.288
```

YAML also has flow styles, using explicit indicators rather than indentation to denote scope. The flow sequence is written as a comma separated list within square brackets. In a similar manner, the flow mapping uses curly braces.

Example 2.5. Sequence of Sequences

```
- [name        , hr, avg  ]
- [Mark McGwire, 65, 0.278]
- [Sammy Sosa  , 63, 0.288]
```

Example 2.6. Mapping of Mappings

```
Mark McGwire: {hr: 65, avg: 0.278}
Sammy Sosa: {
    hr: 63,
    avg: 0.288
  }
```

2. Structures

- 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.
- 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.

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.

Example 2.7. Two Documents in a Stream (each with a leading comment)

```
# Ranking of 1998 home runs
---
- Mark McGwire
- Sammy Sosa
- Ken Griffey

# Team ranking
---
- Chicago Cubs
- St Louis Cardinals
```

Example 2.8. Play by Play Feed from a Game

```
---
time: 20:03:20
player: Sammy Sosa
action: strike (miss)
...
---
time: 20:03:47
player: Sammy Sosa
action: grand slam
...
```

Repeated nodes (objects) are first identified by an anchor (marked with the ampersand - "&"), and are then aliased (referenced with an asterisk - "*") thereafter.

Example 2.9. Single Document with Two Comments

```
---
hr: # 1998 hr ranking
  - Mark McGwire
  - Sammy Sosa
rbi:
  # 1998 rbi ranking
  - Sammy Sosa
  - Ken Griffey
```

Example 2.10. Node for "Sammy Sosa" appears twice in this document

```
---
hr:
  - Mark McGwire
  # Following node labeled SS
  - &SS Sammy Sosa
rbi:
  - *SS # Subsequent occurrence
  - Ken Griffey
```

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.

Example 2.11. Mapping between Sequences

```
? - Detroit Tigers
  - Chicago cubs
:
  - 2001-07-23

? [ New York Yankees,
    Atlanta Braves ]
: [ 2001-07-02, 2001-08-12,
    2001-08-14 ]
```

Example 2.12. Compact Nested Mapping

```
---
# Products purchased
- item    : Super Hoop
  quantity: 1
- item    : Basketball
  quantity: 4
- item    : Big Shoes
  quantity: 1
```

3. Scalars

- literal style (indicated by "|")
  - Newlines are preserved
- folded style (denoted by ">")
  - Newlines become spaces

Example 2.13. In literals, newlines are preserved

```
# ASCII Art
--- |
  \//||\/||
  // ||  ||__
```

Example 2.14. In the folded scalars, newlines become spaces

```
--- >
  Mark McGwire's
  year was crippled
  by a knee injury.
```

- YAML's flow scalars: plain style and two quoted styles

- Two quoted styles
  - double-quoted style provides escape sequences.
  - single-quoted style is useful when escaping is not needed

- All flow scalars can span multiple lines; line breaks are always folded.

Example 2.17. Quoted Scalars

```
unicode: "Sosa did fine.\u263A"
control: "\b1998\t1999\t2000\n"
hex esc: "\x0d\x0a is \r\n"

single: '"Howdy!" he cried.'
quoted: ' # Not a ''comment''.'
tie-fighter: '|\-*-/|'
```

Example 2.18. Multi-line Flow Scalars

```
plain:
  This unquoted scalar
  spans many lines.

quoted: "So does this
  quoted scalar.\n"
```

4. Tags

- In YAML, untagged nodes are given a type depending on the application.

- Generally use the ***seq***, ***map*** and ***str*** types from the fail safe schema.
- A few examples also use the ***int***, ***float***, and ***null*** types from the JSON schema.
- The repository includes additional types such as ***binary***, ***omap***, ***set*** and others.

Example 2.19. Integers

```
canonical: 12345
decimal: +12345
octal: 0o14
hexadecimal: 0xC
```

Example 2.20. Floating Point

```
canonical: 1.23015e+3
exponential: 12.3015e+02
fixed: 1230.15
negative infinity: -.inf
not a number: .NaN
```

Example 2.21. Miscellaneous

```
null:
booleans: [ true, false ]
string: '012345'
```

Example 2.22. Timestamps

```
canonical: 2001-12-15T02:59:43.1Z
iso8601: 2001-12-14t21:59:43.10-05:00
spaced: 2001-12-14 21:59:43.10 -5
date: 2002-12-14
```

- Explicit typing is denoted with a tag using the exclamation point (“!”) symbol.
- Global tags are URIs and may be specified in a tag shorthand notation using a handle.
- Application-specific local tags may also be used.

Example 2.23. Various Explicit Tags

```
---
not-date: !!str 2002-04-28

picture: !!binary |
 R01GOD1hDAAMAIQAAP//9/X
 17unp5WZmZgAAAOfn515eXv
 Pz7Y6OjuDg4J+fn5OTk6enp
 56enmleECcgggoBADs=

application specific tag: !something |
 The semantics of the tag
 above may be different for
 different documents.
```

Example 2.24. Global Tags

```
%TAG ! tag:clarkevans.com,2002:
--- !shape
  # Use the ! handle for presenting
  # tag:clarkevans.com,2002:circle
- !circle
  center: &ORIGIN {x: 73, y: 129}
  radius: 7
- !line
  start: *ORIGIN
  finish: { x: 89, y: 102 }
- !label
  start: *ORIGIN
  color: 0xFFEEBB
  text: Pretty vector drawing.
```

Example 2.25. Unordered Sets

```
# Sets are represented as a
# Mapping where each key is
# associated with a null value
--- !!set
? Mark McGwire
? Sammy Sosa
? Ken Griff
```

Example 2.26. Ordered Mappings

```
# Ordered maps are represented as
# A sequence of mappings, with
# each mapping having one key
--- !!omap
- Mark McGwire: 65
- Sammy Sosa: 63
- Ken Griffy: 58
```

Full Length Examples

Below are two full-length examples of YAML. On the left is a sample invoice; on the right is a sample log file.

Example 2.27. Invoice

```
--- !<tag:clarkevans.com,2002:invoice>
invoice: 34843
date   : 2001-01-23
bill-to: &id001
    given  : Chris
    family : Dumars
    address:
        lines: |
            458 Walkman Dr.
            Suite #292
        city    : Royal Oak
        state   : MI
        postal  : 48046
ship-to: *id001
product:
    - sku         : BL394D
      quantity    : 4
      description : Basketball
      price       : 450.00
    - sku         : BL4438H
      quantity    : 1
      description : Super Hoop
      price       : 2392.00
tax  : 251.42
total: 4443.52
comments:
    Late afternoon is best.
    Backup contact is Nancy
    Billsmer @ 338-4338.
```

Example 2.28. Log File

```
---
Time: 2001-11-23 15:01:42 -5
User: ed
Warning:
  This is an error message
  for the log file
---
Time: 2001-11-23 15:02:31 -5
User: ed
Warning:
  A slightly different error
  message.
---
Date: 2001-11-23 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
```

YAML example

```
---
firstName: John
lastName: Smith
age: 25
address:
  streetAddress: 21 2nd Street
  city: New York
  state: NY
  postalCode: 10021
phoneNumber:
- type: home
  number: 212 555-1234
- type: fax
  number: 646 555-4567
gender:
  type: male
```

YAML C Library: http://pyyaml.org/wiki/LibYAML

- http://pyyaml.org/download/libyaml/yaml-0.1.7.tar.gz