YAML

YAML (*YAML Markup Language*) یک زبان سریال سازی داده ای قابل خواندن است . معمولا برای فایل های پیکربندی استفاده می شود ، اما می تواند در بسیاری از برنامه های کاربردی که داده ها در آن ذخیره می شوند (مانند خروجی اشکال زدایی) یا انتقال (به عنوان مثال هدر سند) استفاده شود.
بسیاری از برنامه های کاربردی ارتباطی YAML را هدف قرار می دهد اما دارای نحو همانند XML را از بین SGML حداقلی است که عمدا سازگاری با می برد Python- [1] . با استفاده از هر دو تیزر برای نشان دادن تودرتو، و یک فرم جمع و style

جورتر که از [] برای لیست ها استفاده می کند و {} برای نقشه ها [1] ساخت YAML 1.2 supereset از JSON . [2]

YAML	
فرمت فایل نام فایل	`.yaml, .yml`
نوع رسانه های اینترنتی	*not registered*
انتشار اولیه	11 2001 مه
آخرین نسخه	1.2 (نسخه سوم) (1 2009 اکتبر)
نوع فرمت	تبادل اطلاعات
فرمت باز ؟	بله
سایت اینترنتی	yaml .org

YAML انواع داده های سفارشی مجاز هستند، اما بومی اسکالرها (مانند رشته ها ، عدد صحیح و شناورها)، لیست ها و آرایه های انجمنی (که همچنین به نام هاش، نقشه ها یا واژه نامه ها شناخته می شود) را کد گذاری می کند. این نوع داده ها بر اساس زبان برنامه نویسی پرل است، هرچند که همه زبان های برنامه نویسی سطح بالا معمولا از مفاهیم بسیار مشابه استفاده می کنند. نحو کولون محور استفاده شده برای بیان جفت های کلیدی با الهام از هدر های پست الکترونیکی به عنوان تعریف تعریف شده است ، و جدا RFC 0822 شده در) قرض گرفته شده است MIME کننده سند "---" از استفاده می شود ، C توالی فرار از . (RFC 2045 و فضای خالی برای رشته های چند خط الهام گرفته است . لیست ها و هش ها می توانند HTML از حاوی لیست توزیع شده و هش ها باشند که ساختار درختی را تشکیل می دهند . نمودارهای دلخواه

XML میتوانند با استفاده از نامهای یامل (مشابه با SOAP) در نظر گرفته YAML [1] . نمایان شوند در شده است که خواندن و نوشتن در جریان، یک ویژگی الهام گرفته از SAX . [1]

برای بسیاری YAML پشتیبانی از خواندن و نوشتن از زبانهای برنامه نویسی در دسترس است. [3] بعضی و محیط [4] Emacs از ویراستاران کد منبع مانند های تک توسعه مجتمع [5][6][7] دارای ویژگی هایی را آسان تر می کنند، مانند YAML هستند که ساختارهای توزیع تاشو یا به طور خودکار خطاهای نحو را برجسته می کنند.

تاریخ و نام

YAML (/ˈjæməl/) برای ، قافیه با شتر [2] اولین بار توسط کلارک ایوانز در سال 2001 پیشنهاد شد، [8] که آن را همراه با طراحی Ingy DOT NET

YAML شده [9] و اورن بن کاربر. [9] نوشته اصلی *گفته شد به معنی حال یکی دیگر از زبان نشانه گذاری* ، [10] ارجاع مقاصد آن به عنوان زبان نشانه گذاری با یکی دیگر از ساختار، اما پس از آن به *است زبان YAML* عنوان تغییر کاربری داده شد *نشانه گذاری نمی* ، یک مخفف بازگشتی ، به تشخیص خود هدف به عنوان داده گرا، به جای سند نشانه گذاری.

طراحی

نحو

ورق تقلب فشرده و همچنین مشخصات کامل در سایت رسمی موجود است. [11] زیر خلاصه ای از عناصر اساسی است.

تمام مجموعه ی کاراکتر یونیکد را به YAML زبان جز بعضی از کاراکترهای کنترل پذیرش می کند . تمام کاراکترهای پذیرفته شده ممکن است در سند ممکن است در YAML استفاده شوند. سند YAML رمزگذاری شود UTF-32 و UTF-16 ، UTF-8 اجباری نیست، اگر سازنده باید UTF-32 (هرچند باشد JSON سازگاری). [12]

- انداختن فضای باز برای نشان دادن ساختار استفاده می شود؛ با این حال، کاراکترهای تب هرگز به عنوان تورفتگی مجاز نیستند.
- نظرات با علامت شماره (#) شروع می شود، می تواند هر نقطه از یک خط شروع شود و تا انتهای خط ادامه یابد. نظرات باید از صفات دیگر از طریق کاراکترهای سفید جدا شوند. [13] اگر آنها در داخل یک رشته ظاهر شوند، پس آنها علامت شماره (#) هستند.

- لیست اعضا با یک خطای برجسته (-) با یک عضو در هر خط و یا در محدوده مربع ([ ]) تعریف شده و با فضای کاما ( ) جدا می شوند , .
- آرایه های انجمنی با استفاده از فضای کولون ( ) در فرم فرم نمایش داده می شوند : *مقدار* ، هر یک های درون ( ) و braces در هر خط یا محصور در { } , : ( ) جدا از فضای کاما.
  - یک کلید آرایه انجمنی ممکن است با علامت سوال (?) پیش فورده شود تا اجازه داده شود کلیدهای چندگانه لیبرال به صورت یکنواخت ارائه شوند.
- به طور معمول نقل قول (scalars) رشته ها نیستند، اما ممکن است در دو جایزه (") یا یک نقل قول (') تعبیه شوند .

- در داخل نقل قول های دوگانه، کاراکترهای خاص ممکن است با دنباله های فرار از نوع C که با backslash (\ ) آغاز می شوند، نمایان شوند . با توجه به اسناد و مدارک تنها فرار اکتال پشتیبانی می شود \0 .

- اسکالرهای بلوک با دندانه گذاری با اصلاح کننده (fold) های اختیاری برای حفظ خطوط (|) یا خطوط جدید تعریف می شوند (>).
- اسناد چندگانه در یک جریان تک با سه خط (-) جدا می شوند (---).
  - سه دوره (. . .) به صورت اختیاری یک سند در یک جریان را پایان می دهد.
- گره های تکرار شده در ابتدا توسط ampersand (&) مشخص می شوند و سپس با ستاره (*) نامگذاری می شوند.

- گره ممکن است با یک نوع یا برچسب با استفاده از نقطه علامت تعجب (! !) و سپس رشته ای گسترش یابد، برچسب URI که می تواند به یک گذاری می شود.
- در یک جریان ممکن است پیش از YAML مدارک دستورات تشکیل شده از یک نشانه درصد (%) و به دنبال آن پارامتر نام و پارامتر فضایی باشد. تعریف شده است YAML 1.1 دو دستورالعمل در:
    - برای شناسایی نسخه YAML٪ دستور در یک سند مشخص شده استفاده YAML می شود.
    - به عنوان میانبر برای TAG ٪دستور استفاده می شود. این URI پیشوندهای میانبرها می توانند در برچسب های گره نوع استفاده شوند.

مستلزم آن است که کولون و کاما به عنوان YAML جدا کننده لیست بعنوان یک فضای به کار گرفته شوند تا مقادیر اسکالر حاوی علائم تعبیه شده (مانند `5,280` یا `http://www.wikipedia.org` به) طور کلی بدون نیاز به نقل قول محصور شوند.

برای استاندارد YAML در sigil دو علامت اضافی سازی آینده ممکن است: در sign (`@`) و accent hail (`` ` ``).

اجزای اصلی

یک سبک درون خطی برای نشان دادن آرایه YAML ها و لیست های انجمنی ارائه می دهد. در اینجا یک نمونه از اجزای تشکیل شده است.

فرمت بلوک متعارف از یک خط + فاصله برای ایجاد یک مورد جدید در لیست استفاده می کند.

```
---  # فیلم های مورد علاقه
- کازابلانکا
- شمال توسط شمال غربی
- مرد که آنجا نبود
```

فرمت درون خطی اختیاری توسط کاما + فضای محدود شده و درون براکت (مشابه با JSON) محصور شده است . [14]

```
---  # لیست خرید
[ شیر ، کدو تنبل ، تخم مرغ ، آب میوه ]
```

کلید ها از مقادیر با یک کولون + فضا جدا می شوند. YAML بلوک های تند، که معمولا در فایل های داده رایج هستند، از تند و خطوط جدید برای جدا کردن جفت کلید / ارزش استفاده می کنند. بلوک های YAML درون خطی که در جریان داده رایج هستند، از کاما + فضا برای جدا کردن جفت کلید / ارزش بین پایه ها استفاده می کنند.

```
---   #
   متخلخل بلوک نام  :   John
Smith
   33   : سن
--- # Inline Block { : نام
33 : سن ، جان اسمیت )
```

رشته ها نیازی به نقل قول ندارند. دو راه برای نوشتن رشته های چند خطی وجود دارد: یکی از خطوط جدید ذخیره (با استفاده از | شخصیت) و یک خط جدید (با استفاده از > شخصیت) که هر دو به دنبال یک شخصیت جدید هستند.

```
داده ها :  |
    یک بار یک مرد کوتاه از Ealing
بود
    که در یک اتوبوس به
Darjeeling رفت
        در درب گفت:
        لطفا بر روی کف پاشنه""
گفت:
    پس او با دقت روی سقف پاره شد
```

به طور پیش فرض، بازده اصلی (از خط اول) و فضای سفالی فضای خالی است، هرچند رفتارهای دیگر می تواند به صراحت مشخص شود.

```
داده  :  >
        متن  پیچیده
        خواهد  شد  تا  خورده
        را  به  یک
        پاراگراف

        خطوط  خالی  نشانگر
        شکست  پاراگرافها  هستند
```

متن رول، خطوط جدید را به فضاها تبدیل می کند و فضاهای برجسته را حذف می کند.

نام : جان اسمیت ، سن : { 33 } -

نام : سن مری اسمیت -

سال 27 :

مردان : [جان اسمیت ، بیل جونز]

زنان :

- مری اسمیت
- سوزان ویلیامز

اجزاء پیشرفته

را از قابلیت های دیگر زبان YAML دو ویژگی که های سریال سازی داده تشخیص می دهند ساختار [15] و تایپ داده است.

ذخیره سازی اسناد متعدد را در YAML سازه های یک فایل واحد، استفاده از مراجع برای گره های تکراری و استفاده از گره های دلخواه به عنوان کلیدها را قادر می سازد. [15]

برای وضوح، فشردگی و اجتناب از خطاهای ورود فراهم می کند گره های گره (استفاده YAML ،داده &) و مراجع (با استفاده از *). اشاره به کار لنگر برای تمام انواع داده (نگاه کنید به کشتی به مرجع در مثال زیر).

در زیر یک مثال از یک صف در یک ترتیب سنج دستگاه است که در آن دو مرحله به طور مکرر مورد

استفاده قرار می گیرند بدون اینکه هر بار به طور کامل توضیح داده شود.

```
# پروتکل ترتیب سنج برای جراحی لیزر چشم
---
- گام :  & id001
# تعریف id001 برچسب لنگر و
    ابزار : عمل لیزیک
2000
    pulseEnergy :  5.4
    pulseDuration :  12
    تکرار : 1000
    spotSize :  1mm
ساخته

- گام :  & id002
```

```
    instrument :
Lasic 2000
    pulseEnergy :       5.0
    pulseDurration :     10
    repetition :        500
    spotSize :          2mm
-  step :  * id001
# با) اشاره به اولین مرحله anchor
& id001)
-  step :  * id002
# به گام دوم
-  گام :
    << :  * id001
    spotSize :  2mm
# فقط این کلید را دوباره تعریف می
کند، اشاره به استراحت از & id001
-  step :  * id002
```

Explicit data typing is seldom seen in the majority of YAML documents since YAML autodetects simple types. Data types can be divided into three categories: core, defined, and user-defined. Core are ones expected to exist in any parser (e.g. floats, ints, strings, lists, maps, ...). Many more advanced data types, such as binary data, are defined in the YAML specification but not supported in all implementations. Finally YAML defines a way to extend the data type definitions locally to accommodate user-defined classes, structures or primitives (e.g. quad-precision floats).

YAML autodetects the datatype of the entity. Sometimes one wants to cast the datatype explicitly. The most common situation is where a single-word string that looks like a number, boolean or tag requires disambiguation by surrounding it with quotes or using an explicit datatype tag.

```
---
a: 123
# an integer
b: "123"
# a string, disambiguated
by quotes
c: 123.0
# a float
```

```yaml
d: !!float 123
# also a float via explicit
data type prefixed by (!!)
e: !!str 123
# a string, disambiguated
by explicit type
f: !!str Yes
# a string via explicit
type
g: Yes
# a boolean True (yaml1.1),
string "Yes" (yaml1.2)
h: Yes we have No bananas
# a string, "Yes" and "No"
disambiguated by context.
```

Not every implementation of YAML has every specification-defined data type. These built-in types use a double exclamation sigil prefix (`!!`). Particularly interesting ones not shown here are sets, ordered maps, timestamps, and hexadecimal. Here's an example of base64 encoded binary data.

```
---
picture: !!binary |
 R0lGODdhDQAIAIAAAAAAANn
 Z2SwAAAAADQAIAAACF4SDGQ
 ar3xxbJ9p0qa7R0YxwzaFME
 1IAADs=
```

Many implementations of YAML can support user-defined data types for object serialization. Local data types are not universal data types but are defined in the application using the YAML parser library. Local data types use a single exclamation mark (`!`).

```
---
myObject:  !myClass { name:
Joe, age: 15 }
```

Example

Data structure hierarchy is maintained by outline indentation.

```
---
receipt:    Oz-Ware
Purchase Invoice
date:       2012-08-06
customer:
    first_name:   Dorothy
    family_name:  Gale

items:
    - part_no:   A4786
      descrip:   Water
Bucket (Filled)
      price:     1.47
      quantity:  4

    - part_no:   E1628
```

```
        descrip:   High
Heeled "Ruby" Slippers
        size:      8
        price:     133.7
        quantity:  1

bill-to:  &id001
    street: |
            123 Tornado
Alley
            Suite 16
    city:   East
Centerville
    state:  KS

ship-to:  *id001
```

```
specialDelivery:  >
    Follow the Yellow Brick
    Road to the Emerald
City.
    Pay no attention to the
    man behind the curtain.
...
```

Notice that strings do not require enclosure in quotations. The specific number of spaces in the indentation is unimportant as long as parallel elements have the same left justification and the hierarchically nested elements are indented further. This sample document defines an associative array with 7 top level keys: one of the keys, "items",

contains a 2-element list, each element of which is itself an associative array with differing keys. Relational data and redundancy removal are displayed: the "ship-to" associative array content is copied from the "bill-to" associative array's content as indicated by the anchor (`&`) and reference (`*`) labels. Optional blank lines can be added for readability. Multiple documents can exist in a single file/stream and are separated by `---`. An optional `...` می توان در انتهای یک فایل استفاده کرد (مفید برای سیگنال دادن پایان دادن به ارتباطات جریان بدون بستن لوله).

ویژگی های

تعریف ردیف

Because YAML primarily relies on outline indentation for structure, it is especially resistant to delimiter collision. YAML's insensitivity to quotes and braces in scalar values means one may embed XML, JSON or even YAML documents inside a YAML document by simply indenting it in a block literal (using `|` or `>`):

```
---
example: >
        HTML goes into YAML
without modification
message: |
```

```
        <blockquote
style="font: italic 12pt
Times">
        <p>"Three is always
greater than two,
           even for large
values of two"</p>
        <p>--Author
Unknown</p>
        </blockquote>
date: 2007-06-01
```

YAML may be placed in JSON by quoting and escaping all interior quotes. YAML may be placed in XML by escaping reserved characters (`<` , `>` , `&` , `'` ,

") and converting whitespace, or by placing it in a CDATA section.

Non-hierarchical data models

Unlike JSON, which can only represent data in a hierarchical model with each child node having a single parent, YAML also offers a simple relational scheme that allows repeats of identical data to be referenced from two or more points in the tree rather than entered redundantly at those points. This is similar to the facility IDREF built into XML.[16] The YAML parser then expands these references into the fully populated data structures they imply when read in, so

whatever program is using the parser does not have to be aware of a relational encoding model, unlike XML processors, which do not expand references. This expansion can enhance readability while reducing data entry errors in configuration files or processing protocols where many parameters remain the same in a sequential series of records while only a few vary. An example being that "ship-to" and "bill-to" records in an invoice are nearly always the same data.

Practical considerations

YAML is line-oriented and thus it is often simple to convert the unstructured output of existing programs into YAML format while having them retain much of the look of the original document. Because there are no closing tags, braces, or quotation marks to balance, it is generally easy to generate well-formed YAML directly from distributed print statements within unsophisticated programs. Likewise, the whitespace delimiters facilitate quick-and-dirty filtering of YAML files using the line-oriented commands in grep, awk, perl, ruby, and python.

In particular, unlike markup languages, chunks of consecutive YAML lines tend to be well-formed YAML documents themselves. This makes it very easy to write parsers that do not have to process a document in its entirety (e.g. balancing opening and closing tags and navigating quoted and escaped characters) before they begin extracting specific records within. This property is particularly expedient when iterating in a single, stateless pass, over records in a file whose entire data structure is too large to hold in memory, or for which reconstituting the entire structure to extract one item would be prohibitively expensive.

Counterintuitively, although its indented delimiting might seem to complicate deeply nested hierarchies, YAML handles indents as small as a single space, and this may achieve better compression than markup languages. Additionally, extremely deep indentation can be avoided entirely by either: 1) reverting to "inline style" (i.e. JSON-like format) without the indentation; or 2) using relational anchors to unwind the hierarchy to a flat form that the YAML parser will transparently reconstitute into the full data structure.

Security

YAML is purely a data representation language and thus has no executable commands.[17] While validation and safe parsing is inherently possible in any data language, implementation is such a notorious pitfall that YAML's lack of an associated command language may be a relative security benefit.

However, YAML allows language-specific tags so that arbitrary local objects can be created by a parser that supports those tags. Any YAML parser that allows sophisticated object instantiation to be executed opens the potential for an injection attack. Perl parsers that allow loading of objects of arbitrary class

create so-called "blessed" values. Using these values may trigger unexpected behavior, e.g. if the class uses overloaded operators. This may lead to execution of arbitrary Perl code.

The situation is similar for Python parsers. According to the PyYAML documentation:[18]

> *Note that the ability to construct an arbitrary Python object may be dangerous if you receive a YAML document from an untrusted source such as the Internet. The function*

> *yaml.safe_load limits this ability to simple Python objects like integers or lists.*

Data processing and representation

The YAML specification identifies an *instance document* as a "Presentation" or "character stream".[19] The primary logical structures in a YAML instance document are scalar, sequence, and mapping.[20] The YAML specification also indicates some basic constraints that apply to these primary logical structures. For example, according to the specification,

mapping keys do not have an order. In every case where node order is significant, a sequence must be used.[21]

Moreover, in defining conformance for YAML processors, the YAML specification defines two primary operations: *dump* and *load*. All YAML-compliant processors must provide *at least* one of these operations, and may optionally provide both.[22] Finally, the YAML specification defines an *information model* or "representation graph", which must be created during processing for both *dump* and *load* operations, although this representation

need not be made available to the user through an API.[23]

مقایسه با JSON

JSON syntax is a basis of YAML version 1.2, which was promulgated with the express purpose of bringing YAML "into compliance with JSON as an official subset".[24] Though prior versions of YAML were not strictly compatible,[25] the discrepancies were rarely noticeable, and most JSON documents can be parsed by some YAML parsers such as Syck.[26] This is because JSON's semantic structure is equivalent to the optional "inline-style" of writing YAML. While

extended hierarchies can be written in inline-style like JSON, this is not a recommended YAML style except when it aids clarity.

YAML has many additional features lacking in JSON, including comments, extensible data types, relational anchors, strings without quotation marks, and mapping types preserving key order.

مقایسه با XML

YAML lacks the notion of tag attributes that are found in XML. Instead YAML has

extensible type declarations (including class types for objects).

YAML itself does not have XML's language-defined document schema descriptors that allow, for example, a document to self-validate. However, there are several externally defined schema descriptor languages for YAML (e.g. Doctrine, Kwalify and Rx) that fulfill that role. Moreover, the semantics provided by YAML's language-defined type declarations in the YAML document itself frequently relaxes the need for a validator in simple, common situations. Additionally, YAXML, which represents YAML data structures in XML, allows

XML schema importers and output mechanisms like XSLT to be applied to YAML.

خصوصیات پیاده سازی

Some implementations of YAML, such as Perl's YAML.pm, will load an entire file (stream) and parse it *en masse*. Conversely, YAML::Tiny only reads the first document in the stream and stops. Other implementations like PyYaml are lazy and iterate over the next document only upon request. For very large files in which one plans to handle the documents independently, instantiating the entire file before processing may be

prohibitive. Thus in YAML.pm, occasionally one must chunk a file into documents and parse those individually. Fortunately, YAML makes this easy since this simply requires splitting on the document separator, which is `m/^--- -$/` (once whitespace is stripped) as a regular expression in Perl.

Simple YAML files (e.g. key value pairs) are readily parsed with regular expressions without resort to a formal YAML parser. YAML emitters and parsers for many popular languages written in the pure native language itself exist, making it portable in a self-contained

manner. Bindings to C-libraries also exist when speed is needed.

همچنین

- AsciiDoc
- Comparison of data serialization formats
- List of lightweight markup languages
- OGDL
- Plist
- S-expression
- Simple Outline XML
- Xupl

پیوند ها

1. *"Yet Another Markup Language (YAML) 1.0 / Working Draft"* . *10 Dec 2001.*

2. *"YAML Ain't Markup Language (YAML™) Version 1.2"* . *yaml.org. Retrieved 13 September 2015.*

3. *yaml.org*

4. *"Yaml Mode"* . *EmacsWiki. 2015-06-12. Retrieved 2016-12-05.*

5. *aukaost. "Pretty YAML - Packages - Package Control"* . *Packagecontrol.io. Retrieved 2016-12-05.*

6. *"yaml | Eclipse Plugins, Bundles and Products - Eclipse Marketplace"* . *Marketplace.eclipse.org. Retrieved 2016-12-05.*

7. *Ruth Kusterer. "NetBeans IDE - Ruby and Ruby on Rails Development" . Netbeans.org. Retrieved 2016-12-05.*

8. *Evans, Clark (May 11, 2001). "YAML Draft 0.1" . Yahoo! Tech groups: sml-dev. Retrieved 2008-08-02.*

9. *"YAML Ain't Markup Language: About" . Retrieved 2010-06-16.*

10. *"Yet Another Markup Language (YAML) 1.0" . Retrieved 2008-11-24.*

11. *"Reference Card" . YAML.org. Retrieved March 21, 2014.*

12. *"YAML Ain't Markup Language (YAML™) Version 1.2" . Yaml.org. Retrieved 2016-09-07.*

13. *"YAML Ain't Markup Language (YAML™) Version 1.2" . Yaml.org. Retrieved 27 May 2015.*

14. *"Cloud Based Management apps" . JigoCloud.com. Retrieved 2016-09-28.*

15. *"YAML 1.2 specification of Structures" . Yaml.org. Retrieved 22 April 2014.*

16. *"Extensible Markup Language (XML) 1.0 (Second Edition)" . W3.org. Retrieved 27 May 2015.*

17. *A proposed "yield" tag will allow for simple arithmetic calculation.*

18. *"PyYAML Documentation, Loading YAML" . Pyyaml.org. Retrieved 2016-09-28.*

19. *"Ain't Markup Language (YAML) Version 1.1" . YAML.org. Retrieved 2016-09-28.*

20. *Additional, optional-use, logical structures are enumerated in the YAML types repository."Language-Independent Types for YAML Version 1.1" . Yaml.org. Retrieved 2007-11-04.The tagged types in the YAML types repository are optional and therefore not essential for conformant YAML processors. "The use of these tags is not mandatory."*

21. *"YAML Ain't Markup Language (YAML) Version 1.1" . Yaml.org. Retrieved 27 May 2015.*

22. *"Ain't Markup Language (YAML) Version 1.1" . Yaml.org. Retrieved 2016-09-28.*

23. *"YAML Ain't Markup Language (YAML) Version 1.1" . Yaml.org. Retrieved 27 May 2015.*

24. *"YAML Ain't Markup Language (YAML™) Version 1.2" . Yaml.org. Retrieved 27 May 2015.*

25. *The incompatibilities were as follows: JSON allows extended character sets like UTF-32 and had incompatible unicode character escape syntax relative to YAML; YAML required a space after separators like comma, equals, and colon while JSON does not. Some non-standard*

implementations of JSON extend the grammar to include Javascript's `/*...*/` *comments. Handling such edge cases may require light pre-processing of the JSON before parsing as in-line YAML. See also [1]* .

26. *Parsing JSON with SYCK* . *Note that e.g. Symfony's YAML parser does not support line breaks inside [] or {} structures, which is a major incompatibility with JSON.*

پیوند به بیرون

- YAML Quick Reference card
- YAML.org
- YAML Specification

- YAML improves on XML . Intro to YAML in Python
- YAML Validator YAML Lint
- YAML Validator YAML Validator
