Email Archive: yaml-core (read-only)

Re: [Yaml-core] Scope for YAML 1.2

From: Tim Parkin <tim@po...> - 2006-02-03 22:29

```
Clark C. Evans wrote:
> A few clarifications; my send-trigger-finger got prematurely activated.
>
> On Fri, Feb 03, 2006 at 04:11:35PM -0500, Clark C. Evans wrote:
> | Last week we had a brief conversation between Oren, Ingy and myself
> | on the topic of JSON compatibility -- which I believe should be the
> | limited scope for YAML 1.2; here are a few declarations to help set
> | the direction for 1.2 activity:
> |
> |   1. Changes to the specification should first and formost include
> |      clarifications of trouble-spots from the past specification;
> |      ideally, listing those as a separate 1.1 Errata.
> |
> |   2. No new features will be included in YAML 1.2, the sole goal is to
> |      make YAML as compatible with JSON as possible; in particular, only
> |      the "in-flow" forms of constructs will be changed (other than
> |      corrections) since these productions are the ones relating to JSON.
> |
> |   3. Constructs in YAML which are valid JSON but have _different_
> |      interpretation will become syntax errors.  This implies that not
> |      all JSON will be valid YAML (unfortunately); a particular example
> |      of currently valid YAML that will become invalid: {"key":"value"}
>
>   In JSON, {"key":"value"} is equivalent to { 'key': 'value' }
>   In YAML, {"key":"value"} is equivalent to { 'key:value': '' }
>

> There are a few other edge cases like this that we should make
> illegal YAML; a different interpretation is seen as the _worst_
> case scenerio.
>
> |   4. I'd like to consider adding /* c-style */ comments to in-line YAML
> |      as these are valid JSON.
>                          ^ Javscript
>
> This was a typo, sorry.  JSON does not support /* -- */ nor //
> style comments; unfortunately, I've seen many documents which
> claim to be JSON that included /* c-style */ comments; so I
> thought we could at least consider using these comments in
> the very limited scope of our JSON-like sub-syntax..
>
> |   5. Any further changes needed to harmonize YAML /w Javascript or
> |      reduce the chance of error when using JSON in the context of
> |      a YAML processor.  In particular, we will consider making illegal
> |      things which are valid Javascript (but illegal JSON) that happen
> |      to also be YAML, but with a different meaning.
> |
> | Kind Regards,
> |
```

> | Clark
> |
> |
> | ----------
> | This SF.net email is sponsored by: Splunk Inc. Do you grep through log files
> | for problems? Stop! Download the new AJAX search engine that makes
> | searching your log files as easy as surfing the web. DOWNLOAD SPLUNK!
> | http://sel.as-us.falkag.net/sel?cmd=lnk&kid=103432&bid=230486&dat=121642
> | _____
> | Yaml-core mailing list
> | Yaml-core@...
> | https://lists.sourceforge.net/lists/listinfo/yaml-core
>
>

I hope I'm not offending anyone if I suggest that a suitable goal for
yaml is to simplify the language a little to help people write parsers
for it? It's been pointed out many times that the spec really
overcomplicates things for developers. If there was a suitable way of
writing a preparser to handle some of the context sensitive nature of
the grammar and then a standard production set that could handle the
remainder I think a lot of people would have started using yaml. Syck
seems to be (correct me if I'm wrong) pretty much the de-factor yaml
standard implementation. I don't want to break yaml, but I really think
there is a space for a yaml-lite that sits somewhere the current spec
and json; Allowing all the ease of use and flexibility without the
'special cases' that I don't think add to the predictability of the grammar.

I'd be happy to help on this as I've felt the pain of trying to develop

a production based grammar myself. I'd be interested in peoples opinions
in the matter and I hope I'm not offending anybody; I think yaml is
great and we're using it in pretty much every we do, I just think it
could be so much more.

'hope I'm not being to honest'ly yours

Tim

Thread View

Thread	Author	Date
[Yaml-core] Scope for YAML 1.2	Clark C. Evans <cce@cl...>	

Re: [Yaml-core] Scope for YAML 1.2

From: Clark C. Evans <cce@cl...> - 2006-02-03 21:49

A few clarifications; my send-trigger-finger got prematurely activated.

On Fri, Feb 03, 2006 at 04:11:35PM -0500, Clark C. Evans wrote:
| Last week we had a brief conversation between Oren, Ingy and myself
| on the topic of JSON compatibility -- which I believe should be the
| limited scope for YAML 1.2; here are a few declarations to help set
| the direction for 1.2 activity:
|
| 1. Changes to the specification should first and formost include
| clarifications of trouble-spots from the past specification;
| ideally, listing those as a separate 1.1 Errata.
|
| 2. No new features will be included in YAML 1.2, the sole goal is to
| make YAML as compatible with JSON as possible; in particular, only

```
|      the "in-flow" forms of constructs will be changed (other than
|      corrections) since these productions are the ones relating to JSON.
|
|   3. Constructs in YAML which are valid JSON but have _different_
|      interpretation will become syntax errors.  This implies that not
|      all JSON will be valid YAML (unfortunately); a particular example
|      of currently valid YAML that will become invalid: {"key":"value"}

  In JSON, {"key":"value"} is equivalent to { 'key': 'value' }

  In YAML, {"key":"value"} is equivalent to { 'key:value': '' }

There are a few other edge cases like this that we should make
illegal YAML; a different interpretation is seen as the _worst_
case scenerio.

|   4. I'd like to consider adding /* c-style */ comments to in-line YAML
|      as these are valid JSON.
                         ^ Javscript

This was a typo, sorry.  JSON does not support /* -- */ nor //
style comments; unfortunately, I've seen many documents which
claim to be JSON that included /* c-style */ comments; so I
thought we could at least consider using these comments in
the very limited scope of our JSON-like sub-syntax..

|   5. Any further changes needed to harmonize YAML /w Javascript or
|      reduce the chance of error when using JSON in the context of
|      a YAML processor.  In particular, we will consider making illegal
|      things which are valid Javascript (but illegal JSON) that happen
|      to also be YAML, but with a different meaning.
|
| Kind Regards,
|
| Clark
|
|
| -------------------------------------------------------
| This SF.net email is sponsored by: Splunk Inc. Do you grep through log files
| for problems?  Stop!  Download the new AJAX search engine that makes
| searching your log files as easy as surfing the  web.  DOWNLOAD SPLUNK!
| http://sel.as-us.falkag.net/sel?cmd=lnk&kid=103432&bid=230486&dat=121642
| _____________________________________________
| Yaml-core mailing list
| Yaml-core@...
| https://lists.sourceforge.net/lists/listinfo/yaml-core
```

Re: [Yaml-core] Scope for YAML 1.2

From: Tim Parkin <tim@po...> - 2006-02-03 22:29

```
Clark C. Evans wrote:
> A few clarifications; my send-trigger-finger got prematurely activated.
>
> On Fri, Feb 03, 2006 at 04:11:35PM -0500, Clark C. Evans wrote:
> | Last week we had a brief conversation between Oren, Ingy and myself
> | on the topic of JSON compatibility -- which I believe should be the
> | limited scope for YAML 1.2; here are a few declarations to help set
> | the direction for 1.2 activity:

> |
> |   1. Changes to the specification should first and formost include
> |      clarifications of trouble-spots from the past specification;
> |      ideally, listing those as a separate 1.1 Errata.
> |
> |   2. No new features will be included in YAML 1.2, the sole goal is to
> |      make YAML as compatible with JSON as possible; in particular, only
> |      the "in-flow" forms of constructs will be changed (other than
> |      corrections) since these productions are the ones relating to JSON.
> |
> |   3. Constructs in YAML which are valid JSON but have _different_
> |      interpretation will become syntax errors.  This implies that not
> |      all JSON will be valid YAML (unfortunately); a particular example
> |      of currently valid YAML that will become invalid: {"key":"value"}
>
```

> In JSON, {"key":"value"} is equivalent to { 'key': 'value' }
> In YAML, {"key":"value"} is equivalent to { 'key:value': '' }
>
> There are a few other edge cases like this that we should make
> illegal YAML; a different interpretation is seen as the _worst_
> case scenerio.
>
> | 4. I'd like to consider adding /* c-style */ comments to in-line YAML
> | as these are valid JSON.
> ^ Javscript
>
> This was a typo, sorry. JSON does not support /* -- */ nor //
> style comments; unfortunately, I've seen many documents which
> claim to be JSON that included /* c-style */ comments; so I
> thought we could at least consider using these comments in
> the very limited scope of our JSON-like sub-syntax..
>
> | 5. Any further changes needed to harmonize YAML /w Javascript or
> | reduce the chance of error when using JSON in the context of
> | a YAML processor. In particular, we will consider making illegal
> | things which are valid Javascript (but illegal JSON) that happen
> | to also be YAML, but with a different meaning.
> |
> | Kind Regards,
> |
> | Clark
> |
> |
> | ---
> | This SF.net email is sponsored by: Splunk Inc. Do you grep through log files
> | for problems? Stop! Download the new AJAX search engine that makes
> | searching your log files as easy as surfing the web. DOWNLOAD SPLUNK!
> | http://sel.as-us.falkag.net/sel?cmd=lnk&kid=103432&bid=230486&dat=121642
> | _____
> | Yaml-core mailing list
> | Yaml-core@...
> | https://lists.sourceforge.net/lists/listinfo/yaml-core
>
>

I hope I'm not offending anyone if I suggest that a suitable goal for
yaml is to simplify the language a little to help people write parsers

for it? It's been pointed out many times that the spec really
overcomplicates things for developers. If there was a suitable way of
writing a preparser to handle some of the context sensitive nature of
the grammar and then a standard production set that could handle the
remainder I think a lot of people would have started using yaml. Syck
seems to be (correct me if I'm wrong) pretty much the de-factor yaml
standard implementation. I don't want to break yaml, but I really think
there is a space for a yaml-lite that sits somewhere the current spec
and json; Allowing all the ease of use and flexibility without the
'special cases' that I don't think add to the predictability of the grammar.

I'd be happy to help on this as I've felt the pain of trying to develop
a production based grammar myself. I'd be interested in peoples opinions
in the matter and I hope I'm not offending anybody; I think yaml is
great and we're using it in pretty much every we do, I just think it
could be so much more.

'hope I'm not being to honest'ly yours

Tim

Re: [Yaml-core] Scope for YAML 1.2

From: Clark C. Evans <cce@cl...> - 2006-02-04 00:14

On Fri, Feb 03, 2006 at 03:05:14PM -0800, Ben Wilhelm wrote:
| Tim Parkin wrote:
| >I hope I'm not offending anyone if I suggest that a suitable goal for
| >yaml is to simplify the language a little to help people write parsers
| >for it? It's been pointed out many times that the spec really
| >overcomplicates things for developers.

```
|
| I just thought I'd take the opportunity to mention that if there was a
| good solid YAML parser for C++, I would already be using it in several
| projects. I don't know all the reasons why there isn't a YAML C++
| wrapper yet, but simplifying the spec certainly couldn't hurt.

Tim,

    You certainly are not offending anyone; I've had the same
    experiences.

    There are several complications -- the problem is, I just don't
    know what would make things easier vs what would reduce usability.
    Perhaps we should start a thread that would list items that are
    particularly problematic but which bring little value?

Ben,


    Thanks for your word of support.

Best,

Clark
```

Re: [Yaml-core] Scope for YAML 1.2

From: Tim Parkin <tim@po...> - 2006-02-04 12:11

```
Clark C. Evans wrote:
> Tim,
>
>     You certainly are not offending anyone; I've had the same
>     experiences.
>
>     There are several complications -- the problem is, I just don't
>     know what would make things easier vs what would reduce usability.
>     Perhaps we should start a thread that would list items that are
>     particularly problematic but which bring little value?
>

The core of my proposal would be to make yaml a lot more attractive to
implementors. I see the only way to make this possible is to make
sufficient simplifications to the language such that it is possible to
use standard parsing tools to build a parser. Obviously the inclusion of
indentation makes the grammar context sensitive; but if a simple,
documentable algorithm could be used to pre-parse the yaml and replace
the indentation de-dentation with appropriate markers, generally
available ebnf grammar parsers should be able to be built which will
perform well (it's a well constrained problem thats been addressed in
detail many times before).

I'm not saying I know yaml inside out and it (which I really should
having worked with it for so long) and may not be straightforward to
just drop a few productions and make everything still compatible; but if
a big change is being considered, perhaps it would be good to consider
as well?

An example of one of the issues I've seen is the fact that list elements
don't need to be indented (a very minor special case that just adds
confusion to both users and implementers).

I'd be happy to work on this if it's seen as an important goal. I'm also
sure that a simple to understand yaml spec that can be easily built and
adapted is exceptionally desirable by a lot of people.

Perhaps it needs be a new mini grammar that fits between json and yaml -
I don't have a fixed idea at the moment but I know the goals.


Tim

p.s. the new python site uses yaml as well as restructured text to build
the whole site. beta.python.org is the most up to date copy at the moment.
```

Re: [Yaml-core] Scope for YAML 1.2

From: Kirill Simonov <xi@ga...> - 2006-02-04 17:06

```
Hi Clark,

>   In JSON, {"key":"value"} is equivalent to { 'key': 'value' }
>   In YAML, {"key":"value"} is equivalent to { 'key:value': '' }

I can't get it. Is it really so? How can "key":"value" be parsed as
'key:value' ? Perhaps it's '"key":"value"'?

Anyway I believe it should be parsed as { 'key': 'value' }, because it's the
syntax used in Python, Javascript, and many other languages. Any different
interpretation is just too surprising.

For YAML 1.2 goals, I would also like to see some attention paid to the bugs
in the production rules. Making a complete YAML parser is already too
difficult, but with bugs in productions it's just impossible. Moreover I
volunteer to find the bugs provided someone else fixes them :)

Is the source of the spec open? Is there an cvs repository for it?

--
xi
```

Re: [Yaml-core] Scope for YAML 1.2

From: Andy <yamlcore@th...> - 2006-02-04 20:15

```
On Sat, 2006-02-04 at 19:02 +0200, Kirill Simonov wrote:
> Hi Clark,
>
> >   In JSON, {"key":"value"} is equivalent to { 'key': 'value' }
> >   In YAML, {"key":"value"} is equivalent to { 'key:value': '' }

>
> I can't get it. Is it really so? How can "key":"value" be parsed as
> 'key:value' ? Perhaps it's '"key":"value"'?
>
> Anyway I believe it should be parsed as { 'key': 'value' }, because it's the
> syntax used in Python, Javascript, and many other languages. Any different
> interpretation is just too surprising.

I find it unintuitive also, but remember that in YAML whitespace is
important.  Values that are butted up against each other without
intervening whitespace are like a series of string literals in C.  The
construct:

    char *x = "hello" ", goodbye";

will produce a single string "hello, goodbye".  I can see the use of
this in YAML when some members of the literals need to be quoted, but
the fact that the colon is outside the quotes makes that confusing.  Why
shouldn't be inside the quotes?  Why does a colon not need to be quoted
(inside quotes)?  There are multiple quoting constructs, and this makes
whitespace (or lack thereof) another kind of quote/escape method, but
it's defined by the lack of the character.

I agree that the above is an obscure feature that only increases the
learning curve and decreases the chances of alternative parsers being
written.

While we're throwing out our opinions and ideas... I like the !!type
syntax, but it seems to have limited use when going from one language to
another, especially with the more complex types.  A simple designation
```

of type or class as a "hint" would be useful. It's not obvious how strict the import of a structure with type information should be -- if the specified class can not be found, should an error be thrown? But then you can't access the data, even if the language has an "anonymous"/classless object syntax like Javascript, PERL and PHP do. And what's the standard for how that type information gets stored in languages that don't have a place to store metadata about a data structure (in perl, you can bless a ref to any string pretty much, and store some limited amount of type data in that string without having to actually define the class. you can do something similar with javascript and PHP, by defining additional fields or creating classes at run time (also possible in perl), but this can get hairy as you move your data from one language to another because there's no common place to store that information.

It seems there are too many ways to express the same thing. The different ways the same structure can be represented in both block and inline styles (I'm not sure if I'm using the right terms here) is confusing. Especially when hand-editing YAML, which I understood to be one of its strengths.

At the first intro you have to YAML, it seems really simple, and then as you read further and read the spec, it ends up seeming overly complex. I looked at YAML because I wanted something that could be a common serialization method between languages that could be represented using the built-in types in the language without having to write a custom parser for each document, something that XML does not have sufficiently

standardized, I think. E4X goes a long way to removing that problem from XML, but YAML does it without adding any specific new syntax for accessing the data structures to any of the supported languages.

--
Andy <yamlcore@...>

Re: [Yaml-core] Scope for YAML 1.2

From: Clark C. Evans <cce@cl...> - 2006-02-04 21:13

On Sat, Feb 04, 2006 at 07:02:05PM +0200, Kirill Simonov wrote:
| Hi Clark,
|
| > In JSON, {"key":"value"} is equivalent to { 'key': 'value' }
| > In YAML, {"key":"value"} is equivalent to { 'key:value': '' }
|
| I can't get it. Is it really so? How can "key":"value" be parsed as
| 'key:value' ? Perhaps it's '"key":"value"'?

Correct, "key":"value" is a syntax error in YAML; I think the example I had in mind was key:value, which is invalid JSON -- however, it is so close as to be confusing.

In YAML, { key:1 } is equivalent to { 'key:1': '' }
In JSON, { key:1 } is a syntax error, since all keys need
to be quoted; however in Javascript it is { 'key': 1 }

| Anyway I believe it should be parsed as { 'key': 'value' }, because
| it's the syntax used in Python, Javascript, and many other languages.
| Any different interpretation is just too surprising.

Indeed. It's a consequence of 2 different features of YAML that behave in a rather un-intuitive manner:

(a) In order to create un-ordered sets, you can use the mapping
syntax where omitting the : '' is permitted. Hence, a value
of { one, two }, is equivalent to { 'one': '', 'two': '' }

(b) In order to allow for perl-packages, and date/time to be
included without requiring quoting, we permit values using
a colon to be unquoted: 10:20 is a valid value.

The change to YAML that I'm thinking is:

- Permit the colon _only_ in plain scalar values in the "block"

```
    form (but not in keys):

      ---

      time: 10:00 AM
      alternatives:
        - 1:00 PM
        - 9:00 AM

  - Conversly, do not allow the colon in any block key, nor in any
    in-line mapping or sequence:

    # all illegal
    ---
    10:20: ten-twenty
    ---
    { time: 10:00AM }
    ---
    [ 1:00 PM ]
    ---
    { 10:20: ten-twenty }
    ...

| For YAML 1.2 goals, I would also like to see some attention paid to the bugs
| in the production rules. Making a complete YAML parser is already too
| difficult, but with bugs in productions it's just impossible. Moreover I
| volunteer to find the bugs provided someone else fixes them :)

Super.  I'd like to release this as an errata.


| Is the source of the spec open? Is there an cvs repository for it?

I's currently private on my server; but there is no reason why
it shouldn't be opened up.

Best,

Clark
```

Re: [Yaml-core] Scope for YAML 1.2

From: Clark C. Evans <cce@cl...> - 2006-02-04 22:30

```
On Sat, Feb 04, 2006 at 02:16:22PM -0600, Andy wrote:
| While we're throwing out our opinions and ideas... I like the !!type
| syntax, but it seems to have limited use when going from one language to
| another, especially with the more complex types.  A simple designation
| of type or class as a "hint" would be useful.

How they type information is used should really be configurable by
the library you use; in a "resolver".  I think one "commendation"
was just to issue warnings for unknown types; but still load them

as strings/mappings/arrays, etc.

| And what's the standard for how that type information gets stored in
| languages that don't have a place to store metadata about a data
| structure (in perl, you can bless a ref to any string pretty much, and
| store some limited amount of type data in that string without having to
| actually define the class.

I don't know if there is a standard; but you can always use a 'shadow'
mechianism.  You matain a "parallel" array that associates each object
(via the address, or id() in python, etc.) with additional type
information about the object.

In the Syck library, it defines 3 "YAML Classes", each of which holds
information about type, and even style information.

| It seems there are too many ways to express the same thing.

Well, in this case; JSON is a wonderful option.  Although what JSON
lacks is:
```

```
  (a) a mechanism to denote the "type" of a node; and
  (b) an alias mechanism to handle re-occurances of the
      same node within the tree.

Perhaps a good activity would be identifying a simple extension
of JSON that provided _only_ those two features.  We could then
work on making this the "inline" form of YAML, and perhaps even
call it "canonical" (if one could have such a thing).

| At the first intro you have to YAML, it seems really simple, and then as
| you read further and read the spec, it ends up seeming overly complex.

I'd say for any given usage, it's overly complex.  However, if you
take a half-dozen uses that were brought up on the list over the
past few years, it does a nice job with each one.  In most programming
languages there are many ways to write a "loop", depending upon the
particular needs of the loop.  Unlike JSON (one-way-to-do-it), YAML
has taken the opposite approach -- which can be a liability.

| I looked at YAML because I wanted something that could be a common
| serialization method between languages that could be represented using
| the built-in types in the language without having to write a custom
| parser for each document

This seems to be a very important, and emerging use-case that isn't
that compatible with the "human readability" requirement of YAML.

I think a JSON+TYPEs is something we should perhaps work-on; and then
come up with a standard set of types that cross-languages could use.
We worked quite a bit on the type-library for YAML, and the best ways
to encode them into text; perhaps our syntax is too flexible though.
However, it'd be nice if what ever type system did emerge for a
JSON+TYPEs it would use URIs, and ideally ones that could be easily
written in a friendly-way using YAML type shorthands (see the spec).

Thanks tons for your suggestion Andy!


Best,

Clark
```

[Yaml-core] Re: Scope for YAML 1.2

From: Jeff Kowalczyk <jtk@ya...> - 2006-02-05 02:27

```
Clark C. Evans wrote:
> help set the direction for 1.2 activity:

This is more a comment on the process for YAML-1.2; I'd like to see
real-world parser design interacting with the spec process. YAML has an
uphill climb to find a niche with JSON's popularity, and XML's
entrenchment.

I think if we go another long stretch without pure scripting language
parsers, YAML is probably not going to fulfill its potential (for
widespread adotion).

I'm speaking only as an interested user who would like to introduce YAML
to various useful projects and frameworks where I can identify a need.
From that point of view, it would be tremendous if the YAML-1.2
development process could proceed with elements of the following:

- A draft of the YAML-1.2 spec is produced.

- The draft is published in a public repository, commit privileges can be
obtained through some reasonable process (i.e. more committers).

- A recommended reference parser design accompanies the spec draft
(push/pull, etc.) based on identified use cases. UML and pseudo code for
the reference parser design might even be appropriate as an appendix of
the spec.

- Reference parser projects are started in various scripting languages
```

(perl, python, ruby, javascript etc.), based on availability of volunteers and/or sponsorship. This is to unify the efforts of various eager elements within the community, so that usable parsers exist even before the spec is finished. The reference parsers are developed in the same repository as the spec.

- Reference parsers share a common design, to demonstrate the best-practice parser design in different languages.

- The draft spec and the reference parser design is exercised completely by literate unit and functional tests for each reference parser. For python, 'doctests' are what I have in mind, other scripting languages

likely have something similar. JSON interoperability would be a component of these doctests.

- User documentation is built up as part of the doctest process.

- The draft is updated based on feedback about the spec itself, and about issues encountered during parser implementation.

After some number of draft cycles:

- The points where C will be needed for performance are identified (hopefully common across reference parsers).

- A portable C library is developed to service those performance hotspots.

- C hotspot library language bindings for the reference parsers are developed and integrated.

- The reference parsers always retain their fallback capability when the C extension is not available.

As the spec nears final status:

- A portable C parser of the common reference design is written which also uses the common C hotspot library.

- C reference parser language bindings are developed.

Once the spec is marked final:

- As of the YAML-1.2 announcement, is part of YAMLs 'draw' that pure scripting language, scripting with C extension, and pure C YAML-1.2 parsers are immediately available, along with comprehensive documentation via the 'doctests'.

- Profit! ;)

Thanks for reading through this naive wishing-for-a-pony scenario.

Surveying the list of YAML users, it seems that a large fraction of people here are experienced enough to have written parsers of some sort. The fraction that could contribute to a directed implementation of a reference design in their favorite language is probably much larger. More people could contribute to doctests and editing thereof.

With a fairly open subversion repository (to support all the cross-project branching and merging), I submit for your consideration that the total process of spec-to-usable-implementations will not be slower because of the parser-driven process. I suspect it might even converge on the 'right' YAML for the current marketplace sooner.

Thanks,

Re: [Yaml-core] Re: Scope for YAML 1.2

From: Tim Parkin <tim@po...> - 2006-02-05 11:25

Hi,

I've created a temporary home to persist some of the discussion
(simpleyaml.dyndns.org).

I think Jeff has put forward a good plan, although I'd like to propose a
slight shift in emphasis.

* Develop the simplest possible parser that can do what yaml currently
does which uses existing parsing tools (ebnf)

* Write the spec based on 1)

Step number one will obviously be guided by a 'feature' list with
priorities such that the trade of between a feature and it's impact on
the resultant parsers can be assessed.

An example of the kind of change I'd propose is something like

* any indentation is seen as the beginning of a block
* a return to the previous level of indentation or more is a dedent

Now I'd have to spend a little time looking at the implications of this
but at a glance

* yaml becomes a lot more 'line based' (if you want to use block style)
* explicit indent would be needed for a lot of the current implicit edge
cases

I also think a lot of the edge cases appear to be around the use of
inline block sequences and the like.. are these really needed?

Tim

Fwd: [Yaml-core] Scope for YAML 1.2

From: TRANS <transfire@gm...> - 2006-02-05 15:40

Oops, meant to send this to entire list.

---------- Forwarded message ----------
From: TRANS <transfire@...>
Date: Feb 5, 2006 5:25 AM
Subject: Re: [Yaml-core] Scope for YAML 1.2
To: "Clark C. Evans" <cce@...>

I'm going to go a bit further out here, I suppose its something for
1.3+ but I want to mention them now just so they are put "in mind".

1) Interpolation, both of whole nodes and also substrings.

2) Has thought been given to making Kwalify a YAML standard?

Thanks,
T.

Re: Fwd: [Yaml-core] Scope for YAML 1.2

From: Andy <yamlcore@th...> - 2006-02-05 21:18

On Sun, 2006-02-05 at 15:39 +0000, TRANS wrote:

On Sun, 2006-02-05 at 15:39 +0000, TRANS wrote:
> 2) Has thought been given to making Kwalify a YAML standard?

I think YAML does need some kind official validator and validation
syntax, at a minimum to help test parsers and generators. It would also
be helpful to be able to validate not just an input document, but also
the context into which it will be loaded, to ensure that all the types
are available/defined in the environment or that the loaded data can be
represented reasonably accurately in some accessible format.

My memory is a little hazy as to Kwalify's features, but often times
actual data is stored in mapping keys, not just the values (I don't
think kwalify supported this, at least at the last time I looked at it).
It would be nice to have those validated also. For example, a mapping
of order numbers to customer names, the keys (order numbers) should be
restricted to a specific format like /(ABC|XYZ)\d+(abcdef)?/. The
validator should be able to represent and check this kind of
restriction.

I don't know if I mentioned this before, but I've ported Kwalify to
perl. It's based off one of the older versions of Kwalify, the project
I needed it for has been on hold for 6 months or so so I have not

updated it for the latest versions of Kwalify.

http://phux.net/websvn/listing.php?repname=kwalify-perl

--
Andy <yamlcore@...>

Re: [Yaml-core] Re: Scope for YAML 1.2

From: Tim Parkin <tim@po...> - 2006-02-06 12:20

Tim Parkin wrote:
> Hi,
>
> I've created a temporary home to persist some of the discussion
> (simpleyaml.dyndns.org).

Link now works (needed a redirect)

Re: [Yaml-core] Scope for YAML 1.2

From: Kirill Simonov <xi@ga...> - 2006-02-10 19:59

Hi Clark,

On Sat, Feb 04, 2006 at 04:12:25PM -0500, Clark C. Evans wrote:
> | For YAML 1.2 goals, I would also like to see some attention paid to the bugs
> | in the production rules. Making a complete YAML parser is already too
> | difficult, but with bugs in productions it's just impossible. Moreover I
> | volunteer to find the bugs provided someone else fixes them :)
>
> Super. I'd like to release this as an errata.

I've added a wiki page with the problems I've found so far:
http://trac.xitology.org/pysyck/wiki/SpecBugs

> | Is the source of the spec open? Is there an cvs repository for it?
>
> I's currently private on my server; but there is no reason why
> it shouldn't be opened up.

Could you possibly do it? I'd like to see the great progress that the YAML
spec is making. ;)

Seriously, I want to be sure that I always read the latest revision. It
will allow to see diff-s between revisions too.

It will also make submitting patches easier.

--
xi

Re: [Yaml-core] Re: Scope for YAML 1.2

From: <ingy@tt...> - 2006-02-16 21:40

On 04/02/06 21:27 -0500, Jeff Kowalczyk wrote:
> Clark C. Evans wrote:
> > help set the direction for 1.2 activity:
>
> This is more a comment on the process for YAML-1.2; I'd like to see
> real-world parser design interacting with the spec process. YAML has an
> uphill climb to find a niche with JSON's popularity, and XML's
> entrenchment.

That is a good warning, but remember that YAML is primarily a
serialization language, where XML and JSON are not. (JSON can only
serialize "typeless" nodes). I think YAML has a niche, regardless of
whether it is *hot* right now.

That said, if the implementations don't get solid, YAML will never
really take off.

> I think if we go another long stretch without pure scripting language
> parsers, YAML is probably not going to fulfill its potential (for
> widespread adotion).

Syck is really carrying us, isn't it? But what a great job it is doing!

Syck is already very popular in Ruby and Perl6/Pugs. In Perl5 and Python
there are now good bindings.

YAML.pm is the pure Perl implementation I maintain. It is woeful
compared to Syck, but seems to get most jobs done.

> I'm speaking only as an interested user who would like to introduce YAML
> to various useful projects and frameworks where I can identify a need.
> >From that point of view, it would be tremendous if the YAML-1.2
> development process could proceed with elements of the following:

>
> - A draft of the YAML-1.2 spec is produced.
>
> - The draft is published in a public repository, commit privileges can be
> obtained through some reasonable process (i.e. more committers).

I can do that. Let's keep adopt the Perl6/Pugs model. I'll host the spec on
subversion on yaml.com (which I own), and I'll give committer bits to whomever
drops by and asks for one on this list or on #yaml (irc.freenode.net)

> - A recommended reference parser design accompanies the spec draft
> (push/pull, etc.) based on identified use cases. UML and pseudo code for
> the reference parser design might even be appropriate as an appendix of
> the spec.

Cool.

> - Reference parser projects are started in various scripting languages
> (perl, python, ruby, javascript etc.), based on availability of volunteers
> and/or sponsorship. This is to unify the efforts of various eager elements
> within the community, so that usable parsers exist even before the spec is
> finished. The reference parsers are developed in the same repository as
> the spec.
>
> - Reference parsers share a common design, to demonstrate the
> best-practice parser design in different languages.

>
> - The draft spec and the reference parser design is exercised completely
> by literate unit and functional tests for each reference parser. For
> python, 'doctests' are what I have in mind, other scripting languages
> likely have something similar. JSON interoperability would be a component
> of these doctests.
>
> - User documentation is built up as part of the doctest process.
>
> - The draft is updated based on feedback about the spec itself, and about
> issues encountered during parser implementation.
>
>
> After some number of draft cycles:
>
>
> - The points where C will be needed for performance are identified
> (hopefully common across reference parsers).
>
> - A portable C library is developed to service those performance hotspots.
>
> - C hotspot library language bindings for the reference parsers are
> developed and integrated.
>
> - The reference parsers always retain their fallback capability when the C
> extension is not available.
>
>
> As the spec nears final status:
>
>

> - A portable C parser of the common reference design is written which also
> uses the common C hotspot library.
>
> - C reference parser language bindings are developed.
>
>
> Once the spec is marked final:
>
>
> - As of the YAML-1.2 announcement, is part of YAMLs 'draw' that pure
> scripting language, scripting with C extension, and pure C YAML-1.2
> parsers are immediately available, along with comprehensive documentation
> via the 'doctests'.
>
> - Profit! ;)

:)

That's a lot of stuff, that is basically saying, let's work together as an
organized community with a plan, and move forward.

I'm all for it.

Please join #yaml irc channel those who are interested...

Cheers, Ingy

>
>
> Thanks for reading through this naive wishing-for-a-pony scenario.
>
> Surveying the list of YAML users, it seems that a large fraction of people
> here are experienced enough to have written parsers of some sort. The
> fraction that could contribute to a directed implementation of a reference
> design in their favorite language is probably much larger. More people
> could contribute to doctests and editing thereof.
>
> With a fairly open subversion repository (to support all the cross-project
> branching and merging), I submit for your consideration that the total
> process of spec-to-usable-implementations will not be slower because of
> the parser-driven process. I suspect it might even converge on the 'right'
> YAML for the current marketplace sooner.
>
> Thanks,
>
>
>
> ---
> This SF.net email is sponsored by: Splunk Inc. Do you grep through log files
> for problems? Stop! Download the new AJAX search engine that makes
> searching your log files as easy as surfing the web. DOWNLOAD SPLUNK!

> Searching your log files as easy as surfing the Web. DOWNLOAD SPLUNK!
> http://sel.as-us.falkag.net/sel?cmd=lnk&kid=103432&bid=230486&dat=121642
> _____
> Yaml-core mailing list
> Yaml-core@...
> https://lists.sourceforge.net/lists/listinfo/yaml-core