Ruby - Misc #15744

Improvement needed to documentation of 'Literals'

04/02/2019 05:34 PM - CaryInVictoria (Cary Swoveland)

Status:	Open
Priority:	Normal
Assignee:	

Description

Documentation of "Literals" for v2.6.0 is given here: https://docs.ruby-lang.org/en/2.6.0/syntax/literals_rdoc.html. (I don't think it has been changed for some time.) It gives examples of literals but does not provide a definition. It is comparable to defining an array by giving a few examples. I believe a definition is needed.

I would like to suggest a definition, but I confess I don't know what a Ruby literal is. A definition is attempted at this Wiki for computer programming generally: https://en.wikipedia.org/wiki/Literal_(computer_programming).

I suspect a Ruby literal is an object whose value is in some sense "known" at compile-time. For example, I would think 1, 1.0 and { a: [1, 'cat', ['dog', ['pig', ..10]]] } are literals but { v=>1, 2=>3 } in h = { v=>1, 2=>3 }, v being a variable, is not. Or is it? If the previous line of code had been v=3, Ruby could, at compile-time, infer that the line could be replaced with h = {3=>1, 2=>3}, in which case it would be "known". This example is meant to illustrate why I earlier said "in some sense".

History

#1 - 04/02/2019 08:05 PM - shevegen (Robert A. Heiler)

I am not sure if your definition can be used. For example, you refer to "compile time", but what is meant with compile time? As far as I know, ruby is not compiled in the sense of generating some binaries/binary executable, such as when you have a .c file and compile it via gcc; perhaps we can use the term compile differently (targeting the RubyVM and/or the JIT), but I think the term seems a bit misplaced when referring to literals.

The way I understand the documentation is that literals refers to most "entities" that you can find in a given .rb file, like, valid syntax components. The explanation "Literals create objects you can use in your program." probably refers to literals that can be used without any further ado, e. g. "abc" versus String.new("abc").

I guess your main point is a question as to whether a variable is a literal or not.

IF that is your question then I somewhat agree - the documentation could state whether a variable in itself is a literal. (I assume it is not because the value is not known upfront, unlike e. g. a number such as 42). So the documentation could be changed to reflect this, if you refer to this.

I am not sure if I got all your comment - I think the ruby team is fine updating the documentation; just need to know what has to be changed/improved, possibly why, and to also ideally provide a way for text to be copy/pasted into it. :D

I don't know myself either but here is a start:

"Variables in ruby, such as x = 2 or y = '42', are not literals."

Or something like that, perhaps with a slightly longer explanation (I am guessing myself too).

#2 - 04/05/2019 12:22 AM - CaryInVictoria (Cary Swoveland)

shevegen (Robert A. Heiler) wrote:
...I don't know myself either...

Robert,

I was only musing about a how a "literal" might be defined. My only point is that I think a definition is required, since the term is frequently used in discussing Ruby code. Alas, I don't think it's possible to define anything by example, and while I'm not convinced that a satisfactory definition is within reach, I believe it's worth a try.

#3 - 06/24/2019 02:27 AM - luke-gru (Luke Gruber)

IMO, literals are purely syntactic constructs. That doesn't mean they always behave the same as objects created by other means, ex: when calling methods, but as far as the definition of a literal, it's just a syntactic thing. Perhaps that could be mentioned in the docs, if people agree with that. I thought the docs made that fairly clear, going through all the syntax for different types of literals, but it could be more explicit.

Some object creation through literals does happen to behave differently though, ex: "a literal string" does not call String.new nor String#initialize, same for Arrays, Hashes, lambdas, etc... Perhaps this could be mentioned, but it's probably too much information, and would complicate a document that seems to be aimed at Ruby beginners. Also, string literals can be created frozen with the magic comment, but that's just an optimization.

As for whether variables are literals, no I don't think so. Variables can be referred to in a literal, ex in [a,b,c,d] the array is a literal, but the variables aren't themselves literals.

As for whether literals must be able to be embedded into ruby bytecode to be considered literals, I disagree. This is just an optimization that can be done with some literals.

Do others think this document needs clarification?