JIS X 3017の読み方

前田修吾(@shugomaeda)

ネットワーク応用通信研究所(*NaCl*)

2011-07-17

本日のテーマ

- JIS X 3017の読み方
  - 規格の内容の話
- 標準化プロセスについてはRubyWorld Conferenceで
  - http://www.rubyworld-conf.org
- 本セッションの内容は、私個人の見解によるものであり、所属する企業・団体とは一切関係ありません

Refinementsはどうなった?

- 絶賛放置プレイ中
  - 多忙のため(第三子誕生、バイク購入…)
- 何か問題があるらしい
  - Method Shelters：Classboxes でも Refinements でもない別のやり方
    - この後16:10から小ホールにて

JIS X 3017とは

- JIS X 3017:2011「プログラム言語Ruby」
- Rubyの日本工業規格（JIS）
- 2011年3月22日に制定

JISを読む方法

- 日本規格協会から購入する
  - http://www.webstore.jsa.or.jp/
- 日本工業標準調査会のサイトで閲覧する
  - http://www.jisc.go.jp/app/JPS/JPSO0020.html
  - 無料(ただし、解説は閲覧不可)

目次

解説3.1 既存の実装との対応

ただし，Ruby 処理系の開発の中心は，より新しいバージョンであるMRI 1.9に移っているため，将来の版との互換性を重視しMRI 1.8 とMRI 1.9 との間で共通な部分だけを，この規格で規定することとした。したがって，互換性のない部分については，規格の範囲から除外するか，又は動作を規定しないことを明記することとした。

解説5 原案作成委員会の構成表

5 原案作成委員会の構成表

…

（委員長） 中 田 育 男 筑波大学名誉教授
（幹事） 吉 田 秀 逸 独立行政法人情報処理推進機構
（委員） 筧 捷 彦 早稲田大学

…

笹 田 耕 一 東京大学大学院（平成22年4月から）

…

まつもとゆきひろ 株式会社ネットワーク応用通信研究所

1 適用範囲(1)

この規格は，プログラム言語Rubyの構文規則及び意味規則を規定し，その規格適合処理系，規格厳密適合プログラム及び規格適合プログラムの要件を規定する。

1 適用範囲(2)

この規格は，次の事項を規定しない。
— 規格適合処理系が評価するプログラムテキストの大きさ又は複雑さの限界。
— この規格に適合する実装をサポートするために，データ処理システムが満たさなければならない最小要件。
— データ処理システム上でプログラムを実行する方法。
— 構文の誤り，又は実行時に発生したエラーを報告する方法。

1 適用範囲(3)

- 規定する範囲
  - 言語の構文と意味
  - 処理系・プログラムの要件
- 範囲外
  - リソースによって決まる制限
  - 下位のOSなどの要件
  - プログラムの実行方法やエラー表示

2 引用規格

- ISO/IEC 646:1991，Informationtechnology－ISO 7-bit coded characterset for information interchange
  - 文字コードの規格 (ASCII)
- IEC 60559:1989，Binary floating-pointarithmetic for microprocessor system
  - 浮動小数点数演算の規格 (IEEE 754)

3 規格適合性

- 規格厳密適合プログラム
  - 規格で規定された機能だけを使用
- 規格適合処理系
  - 規格厳密適合プログラムを規格どおりに実行
  - 拡張機能があってもよい
- 規格適合プログラム
  - 規格適合処理系で実行できるプログラム

4 用語及び定義

- オブジェクト・メソッド・クラスなどの基本的な用語を定義
- 詳細は後の箇条(章みたいなもの)で規定

用語定義の例

4.2
クラス
　オブジェクトであって，そのクラスのインスタンスと呼ばれる他のオブジェクトの集合の動作を定義するもの。
　注記　その動作は，インスタンスに対して呼び出すことのできるメソッドの集合である。

5 記法

- この規格で使う記法を定義
- 構文規則と意味規則の記述方法

構文規則

- 基本的にBNFで記述
- 正規表現風の拡張 (?, *, +)
- 否定先読み (先読み∉ { “$”, “@”, “{”})
- 《行終端子》や《空白類》の禁止・必須
- 概念的名前
  - 開始記号から辿れない非終端記号
  - 代入式と代入文をまとめたりする

構文規則の例

```
《論理AND式》::=
    《キーワードAND式》
  | 《演算子AND式》
《キーワードAND式》::
    《式》　[《行終端子》禁止]　“and”　《キーワードNOT式》
《演算子AND式》::
    《等価式》
  | 《演算子AND式》　[《行終端子》禁止]　“&&”　《等価式》
```

意味規則

- 自然言語で記述
- 評価の方法と評価結果の値を規定
- 省略されている場合は、その非終端記号が参照する他の非終端記号の意味規則に従う

意味規則の例

《論理AND式》は，次の手順で評価する。
a) 《論理AND式》が《等価式》である場合，その《等価式》を11.4.4で規定するとおりに評価する。
b) そうではない場合，次の手順を行う。
1) 《式》又は《演算子AND式》を評価する。評価結果の値をXとする。
2) Xが真の場合，《キーワードNOT式》又は《等価式》を評価する。評価結果の値をYとする。《キーワードAND式》又は《演算子AND式》の値はYとする。
3) そうではない場合，《キーワードAND式》又は《演算子AND式》の値はXとする。

6 基本概念

- 以下の基本概念を規定
  - オブジェクト
  - 変数とその種類
  - メソッド、ブロック
  - クラス、特異クラス及びモジュール
  - 真理値

用語の注意点

- 「スーパークラス」や「サブクラス」は、すべての祖先や子孫を表す
- 親は「直接のスーパークラス」、子は「直接のサブクラス」という
- たんに「インスタンス」という場合、サブクラスのインスタンスを含む
- サブクラスのインスタンスを含まない場合、「直接のインスタンス」という

使い方の例

- ObjectはFileのスーパークラス、Fileはオブジェクトのサブクラスである。
- IOはFileの直接のスーパークラス、FileはIOの直接のサブクラスである。
- "foo"はObjectのインスタンスである。
- "foo"はStringの直接のインスタンスである。

7 実行環境

- 仮想的なインタプリタの状態を表す
- 実行環境に対する操作として意味規則を記述
  - Rubyの動的な性質を反映
- 〖大域変数束縛集合〗以外はスタック
- RAILS_ENVとは関係ない

実行環境の属性

〖self〗	self
〖クラスモジュールリスト〗	クラス・モジュールのネスト (cref)
〖省略時可視性〗	デフォルトの可視性
〖局所変数束縛集合〗	ローカル変数の集合
〖呼出し時メソッド名〗	呼出し時のメソッド名
〖定義時メソッド名〗	定義時のメソッド名
〖ブロック〗	渡されたブロック
〖大域変数束縛集合〗	グローバル変数の集合

〖クラスモジュールリスト〗

```
# [[Object]]
module Foo
  # [[Object], [Foo, Object]]
  class Bar
    # [[Foo, Object], [Object],
    #  [Foo::Bar, Foo, Object]]
  end
end
```

- スタックトップはModule.nesting + Object
- 定数探索時などに使用

なぜ〖クラスモジュールリスト〗？

- 定数はなるべく静的に解決し(ているように見せ)たい
- 実際には実行時に探索
- 静的な構造を〖クラスモジュールリスト〗で表現している

ruby_class

- ruby_class (メソッドの定義先クラス) がない
- 『クラスモジュールリスト』のトップの先頭
- class_evalなどでちょっと困る
  - class_evalは定数探索には影響しないが、メソッドの定義先クラスだけ変えるため
  - 定数探索時はclass_evalが『クラスモジュールリスト』に積んだリストを無視

8 字句構造

- 字句構造を規定
- 《入力要素》は以下のいずれか
  - 《行終端子》
  - 《空白類》
  - 《コメント》
  - 《プログラム終端指示子》
  - 《字句》

8.2 プログラムテキスト

プログラムは，プログラムテキストで表現される。プログラムテキストは，《ソース文字》の並びである。《ソース文字》は，ISO/IEC 646:1991 の国際基準版（IRV：International Reference Version）にある文字である。その他の文字集合及びエンコーディングをサポートするかどうかは，未規定とする。

- ASCIIの範囲だけ規定

8.3 行終端子

5.2.3 に規定しているとおり，箇条8 及び15.2.15.4 以外では，《行終端子》は生成規則から省かれている。ただし，《行終端子》が現れてはならない場所，又は，現れなければならない場所をそれぞれ，特記項の禁止［5.2.4 d) 2)参照］又は必須［5.2.4 d) 3)参照］で示す。

- 改行は、意味をもつときと、無視されるときがある

改行が意味をもつ例(1)

- 文などの区切り

10.2 複合文
…
《分離子》::
“.”
,
| [《行終端子》必須]

改行が意味をもつ例(2)

- 演算子の前

11.4.2.2 単一代入
…
《単一変数代入式》::
《変数》［《行終端子》禁止］ “=” 《演算子式》

改行を特別扱いする理由

- 文などの区切り記号(;)を省略したい
  - [《行終端子》必須]
- 一つの文を複数行に分けたい
  - 《行終端子》を無視
- 曖昧なケース
  - [《行終端子》禁止]

8.4 空白類

5.2.3 に規定しているとおり，箇条8 及び15.2.15.4 以外では，《空白類》は生成規則から省かれている。ただし，《空白類》が現れてはならない場所，又は，現れなければならない場所をそれぞれ，特記項の禁止［5.2.4 d) 2)参照］又は必須［5.2.4 d) 3)参照］で示す。

- 《行終端子》同様

空白類が意味をもつ例(1)

11.3.2 メソッド実引数
…
《括弧なし実引数》が，“&”，“<<”，“+”，“-”，“*”，“/”及び“%”のいずれかの文字の並びで始まる場合，次の条件を満たさなければならない。
— その《括弧なし実引数》の直前に一つ以上の《空白類》が存在しなければならない。
— その文字の並びの直後に《空白類》が存在してはならない。

空白類が意味をもつ例(2)

11.4.4 2項演算子式
…
次のいずれかの演算子の直前に《空白類》がある場合，その演算子の直後にも一つ以上の《空白類》がなければならない。
－ 《ビット単位AND 式》の “&”
－ 《ビット単位シフト式》の “<<”

空白類が意味をもつ例(3)

```
《一次式メソッド呼出し》::
    …
  | 《メソッド識別子》
    （［《行終端子》禁止］　［《空白類》禁止］　《括弧付き実引数》）
    《ブロック》?
    …
```


空白類を特別扱いする理由

- メソッド引数の括弧を省略したい
- 曖昧なケースは空白の有無で区別

```
x + y           # (x) + (y)
x+y             # (x) + (y)
x +y            # x(+y)
p (1+2).to_s    # p((1+2).to_s)
p(1+2).to_s     # (p(1+2)).to_s
```

なぜこんなに複雑な文法なのか

- 文法の複雑性がプログラムの簡潔性をもたらす
  - ;や括弧の省略を禁止すれば文法は簡潔
  - しかし、プログラムは冗長になる
- 書き易さだけでなく読み易さが重要
  - attr_accessor :foo
- 文法の複雑性を普段意識する必要はない
  - 2項演算を「x +y」と書く人はいない

Q1

- Choose the correct behavior of the following code:

```
p {:a => 1,  :b => 2}
```

- A. It prints {:a=>1, :b=>2}.
- B. It prints nothing.
- C. It raises an exception.
- D. It causes a syntax error.

A1

- D. It causes a syntax error.

Q2

- Choose the correct behavior of the following code:

```
p {}
```

- A. It prints {}.
- B. It prints nothing.
- C. It raises an exception.
- D. It causes a syntax error.

A2

- B. It prints nothing.

解説

- {がブロックの開始とみなされるため
- 人間がわかってもRubyにはわからないことも
- 動作がおかしい時は括弧を付ける
  - この場合は、波括弧を省略してもいい

JISの規定

11.3.2 メソッド実引数
…
《括弧なし実引数》::
[先読み ∉ { “{” }] [《行終端子》禁止] 《実引数リスト》

13.4 特異クラス

- 処理系定義・未規定の箇所が多い
  - バージョン・実装間の差異がある

特異クラスのスーパークラス

13.4 特異クラス
…
特異クラスの直接のスーパークラスは，処理系定義とする。ただし，特異クラスは，それが関連付けられているオブジェクトのクラスのサブクラスでなければならない。

- 元のクラスのメソッドを呼ぶため

クラスの特異クラスのスーパークラス

直接のスーパークラスをもつクラスの特異クラスは，次の条件を満たさなければならない。
— Ec をクラスC の特異クラス，S をC の直接のスーパークラス，Es をS の特異クラスとする。このとき， Ec は，Es をスーパークラスの一つとしてもつ。

- 特異メソッドを継承するため

その他

特異クラスがクラスのスーパークラスになれるかどうかは，未規定とする［13.2.2 b) 2) i)及び15.2.3.3.1 c)参照］。
特異クラスがクラス変数の集合をもつかどうかは，処理系定義とする。

特異クラスの存在理由

- 特異メソッドを定義するため
  - 特にクラスの特異メソッド
- もともとは実装の詳細だった

Rubyらしさ

- 一見素直、でもちょっと意地悪
- ぱっと見はシンプル
  - よく使う機能は自然(x, y = 1, 2)
- 実は仕様は複雑
  - 特殊ケースは難しい(x = *[])
- 一貫性がないところも
  - 暗黙の型変換とか(規格では未規定)

何でこうなった?

- 仕様書がなかった

- By natarén (CC BY-NC-SA 2.0)

標準化で変わる？

- 変わらない

結論

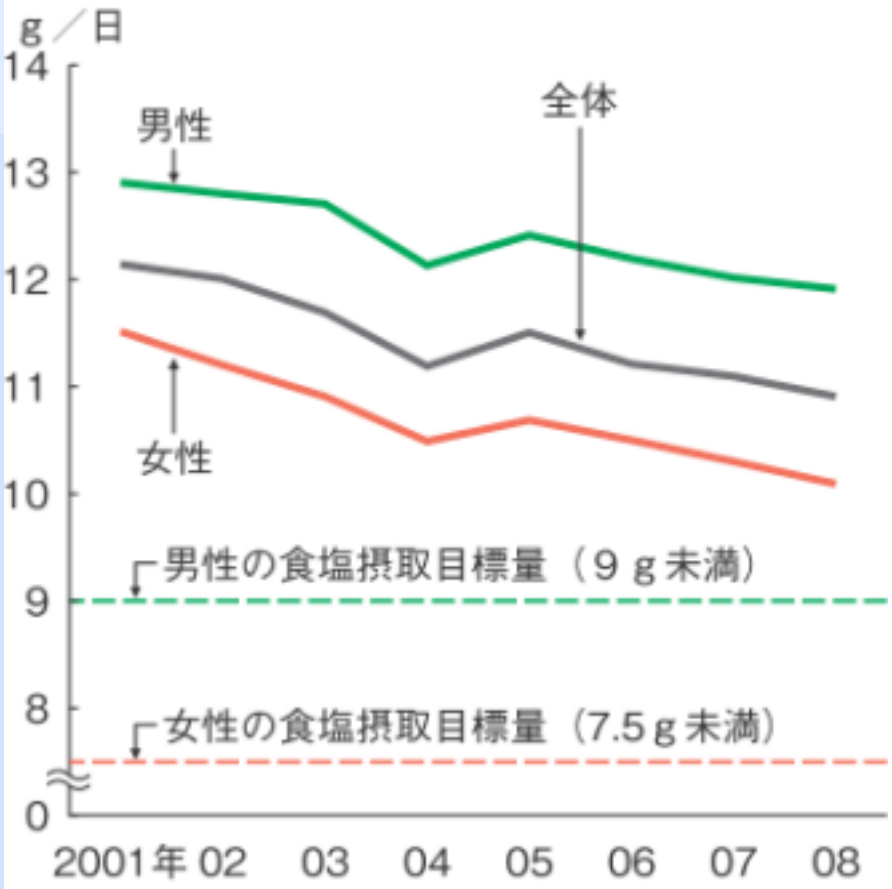
- Rubyは実は複雑
- だからこそ使いやすい
- まつもとさんは永遠のカウボーイプログラマ

NaCl設立10周年

- 2001年7月17日設立
- @gotoyuzoが取締役に就任

若者のNaCl離れが深刻

図２－30　食塩摂取量の推移（20歳以上）



資料：厚生労働省「国民健康・栄養調査」、「日本人の食事摂取基準（2010年版）」

まつもとさんのNaCl離れも深刻

2012年新卒者第三次採用募集

- 8/22(月)：会社説明会 @松江本社
- 8/23(火)：会社説明会 @東京支社
- 8/31(水)：応募〆切
- 中途の人も歓迎
- 詳細はhttp://www.netlab.jpで

Thank you

59