

CoffeeScript クイックリファレンス

coffeescript.org

本稿は、E. Hoigaard さん著 “Smooth CoffeeScript” に付属の “CoffeeScript Quick Reference” の日本語訳です。

一般¹

- 空白は重要
- 行の結びは式の結び - セミコロン不要
- セミコロンは一行に複数式を押し込むために使える
- 関数や *if* 文、*switch*、*try/catch* 中のコードのブロックを囲むのに中括弧 { } の代わりにインデントを使うこと
- コメントは # で始まり、行の終わりまで続く

関数

- 関数は、括弧内のオプションなパラメータリストと矢印、オプションな関数本体で定義される。空の関数はこんな感じ: ->
- 関数を呼び出すのに引数を渡す際、括弧はほとんどの場合不要。暗黙のコールは、行かブロック式の終わりに至るまで引数とみなす。
- 関数は引数のデフォルト値を持つことが可能。デフォルト値を変えるには null でない引数を渡すこと。

オブジェクトと配列

- オブジェクトと配列は JavaScript と同様
- プロパティそれぞれが 1 行ずつリストされる時はコンマはオプション
- YAML のように、明示的な中括弧の代わりにインデントを使ってオブジェクトを生成可能
- *class* のような予約語は文字列としてクォートせずにオブジェクトのプロパティとして使用可能

構文スコープと変数安全性

- 変数は使用されたとき暗黙のうちに宣言される (*var* キーワードなし).
- コンパイラは変数が構文スコープ内で宣言されることを保証する。外の変数は、それがスコープ内のときは内の関数の中で再宣言されない
- 内の変数を使って外の変数をシャドーはできず、ただ外の変数を参照する。だから深く入れ子になった関数で外部変数の名前を再利用することは避けること
- CoffeeScript の出力は、グローバルな名前空間を偶然汚すことを難しくするように匿名関数でラップされる
- 他のスクリプト用にトップレベル変数を作るには、To create top-level variables for other scripts, attach them as properties on *window* か CommonJS の *exports* 上のプロパティとしてそれらを付属すること。以下を使うこと:
`exports ? this`

三点リーダー (スプラッツ)

- 三点リーダー ... は可変の数の *arguments* オブジェクトの代わりに使うことが可能で関数定義と関数呼び出しの両方で利用可能

ループと内包

- 内包 *for ... in* は配列、オブジェクト、レンジ上で有効に作用する
- 内包はオプションな *when* ガード節と現在の配列添字 (*for value, index in array*) の値と一緒にループに取って代わる
- 配列内包は式であり、戻り値になったり割り当てられたり可能
- 内包は *each/forEach*, *map* or *select/filter* に取って代わるかもしれない
- ループの開始と終了がわかっている時は (整数ステップ)、レンジを使うこと
- 固定サイズの増分でステップするには *by* を使うこと
- 変数に内包の値を割り当てる時は、CoffeeScript は繰り返しそれぞれの結果を配列に集める
- ループが副作用のためだけなら *null* か *undefined*, *true* を返すこと
- オブジェクトのキーと値のプロパティ上で繰り返すには *of* を使うこと
- オブジェクト上で直接定義されたキー上で繰り返すには *for own key, value of object* を使うこと
- 唯一の低レベルループは *while* ループ。それは式として使用可能で、ループの間繰り返しそれぞれの結果を含む配列を返す
- *until* は *while not* と同等
- *loop* は *while true* と同等
- *do* キーワードは閉包のラップを差し込み、どんな引数も転送し、渡された関数を呼び出す

Try/Catch/Finally

- *try/catch* 文は JavaScript と同じ (ただし、式)

If, Else, Unless と条件付き割り当て

- *if/else* は括弧や中括弧なしで記述可能
- 複数行の条件文はインデントで境界を定める
- *if* と *unless* は、後置形ですなわち文の終わりで使用可能
- *if* 文は式として使用可能。?: は不要

連鎖比較

- 値が範囲内かテストするには連鎖比較を使うこと:
`minimum < value < maximum`

配列スライスとレンジを使ったスライス

- 配列のスライスを抽出するためにレンジが使用可能
- 二点リーダーを使った [3..6] では範囲は境界含めて (3, 4, 5, 6)
- 三店リーダーを使った [3...6] では範囲は終端を除外して (3, 4, 5)
- 配列の一部をスライスし新たな値で置き換えるため、同じシンタックスが割り当てと一緒に使用可能
- 文字列はイミュータブル (不変) でスライス不可能

埋め込み JavaScript

- CoffeeScript 内で JavaScript コードを埋め込むにはバッククォート ‘‘ を使うこと

¹ Y. Ichikawa 2554/2011 Rev.

すべてが式

- 関数は最後の値を返す
- 戻り値は実行の分岐それぞれからフェッチされる
- 明示的な `return` を使うことで関数本体から早く戻る
- 変数宣言はスコープの一番上にあり、だからたとえ変数が以前に見えなくても一連の式の中で割り当てが使用可能
- 文が式の一部として使われる時、文は閉包ラッパを使って式に変換される。これにより内包の結果の変数への割り当てが可能
- 以下のものは式ではない: `break`, `continue`, `return`

演算子とエーリアス

- CoffeeScript は `==` を `===` に、`!=` を `!==` にコンパイルする。JavaScript の `==` 演算子に等価なものはない
- エーリアス `is` は `===` と等価で、`isnt` は `!==` に対応する
- 論理演算子エーリアス: `and` は `&&`, `or` は `||`, `not` は `!` のエーリアス
- `while` や `if/else`, `switch/when` 文の中で同じ行に本体を置くために `then` キーワードが使用可能
- ブーリアン `true` のエーリアスは `on` と `yes` (YAML と同様)
- ブーリアン `false` のエーリアスは `off` と `no`
- 一行文のため `unless` が `if` の逆として使用可能
- `this.something` の代わりに `@property` や `@method` を使うこと
- 配列上の存在をテストするには `in` を使うこと
- オブジェクト-キーの存在をテストするには `of` を使うこと

存在演算子

- 変数が存在するかチェックするには存在演算子 `?` を使うこと。`?` は変数が `null` か `undefined` でない限り `true` を返す
- 数や文字列での `||=` よりも安全な条件付き割り当てとして `?=` を使うこと
- プロパティの連鎖内での `null` 参照を吸い取るために存在演算子のアクセサ変体 `?.` が使用可能
- 基底値が `null` や `undefined` であり得る場合、ドットアクセサ `.` の代わりに `?.` を使うこと。プロパティすべてが存在したなら期待される結果を返し、連鎖が壊れたら代わりに `undefined` を返す

クラスと継承、スーパー

- 他のほとんどすべてのオブジェクト指向言語と同様のオブジェクト指向
- `class` 構造によって、一つの割り当て可能な式の中で、クラスの名付け、`extends` でのスーパークラスの設定、プロトタイプ的プロパティの割り当て、`constructor`, の定義が可能
- リフレクションをサポートするため、コンストラクタ関数は `class` 名として指名される
- より低レベルの演算子: `extends` 演算子は適切なプロトタイプのセットアップを助ける。`::` はオブジェクトのプロトタイプへのアクセスを与える。`super()` は直系原型の同じ名前のメソッドをコールする
- クラス定義は実行可能なコードブロックであり、それはメタプログラミングで使用可能
- クラス定義の文脈の中で、`this` はクラスオブジェクト自体 (`constructor` 関数) であり、なので静的なプロパティは `@property: value` を使って割り当て可能であり、親クラスで定義された関数は `@inheritedMethodName()` でコール可能

多重割り当て (Destructuring Assignment)

- 複雑な配列やオブジェクトからの値の抽出を便利にするため、CoffeeScript は多重割り当てを実装する
- 配列やオブジェクトリテラルを値に割り当てる時、右辺の値を左辺の変数に割り当てるため CoffeeScript は両辺をそれぞれ分割し、分割それぞれに対してマッチングを図る
- 最も簡単なケースは並列割り当て `[a,b] = [b,a]` である
- 複数の値を返す関数と一緒に使用可能
- どんな深さの配列と一緒に使用可能であり、深く入れ子になったプロパティを得るために入れ子のオブジェクトと一緒に使用可能であり、二点三点リーダーと一緒に組み合わせ可能

関数束縛

- 関数を定義して `this` の現在の値にそれを束縛するのに太い矢印 `=>` が使用可能
- コールバックベースのライブラリを使う時、`each` に渡すイテレータ関数や `bind` と一緒に使うイベントハンドラ関数の生成のためにこれは役立つ
- `=>` を使って生成された関数は関数が定義されたところでの `this` のプロパティにアクセス可能

Switch/When/Else

- `switch` 文はそれぞれのケースの後で `break` 不要
- `switch` は戻り値にもでき割り当てもできる式である
- フォーマットは以下の通り: `switch` 条件, `when` 節, `else` デフォルト節
- `when` 節それぞれに関してコンマで分離された多重値を与えることが可能。値のどれかに一致したらその節が実行される

文字列改変とヒアドキュメント, ブロックコメント

- シングルクォートされた文字列はリテラルである。エスケープ文字にはバックslashを使うこと
- ダブルクォートされた文字列は `#{ ... }` を使って改変される値を許す
- 多行文字列が許される
- ヒアドキュメント `'''` はフォーマットされたテキストやインデントに注意を要するテキストのために (あるいはクォートやアポストロフィのエスケープ避けるために) 使用可能
- ヒアドキュメントを始めるインデントレベルはずっと保持されるので、テキストはコードの本体と合わせて整列可能
- ダブルクォートヒアドキュメント `"""` は改変を許す
- ブロックコメント `###` はヒアドキュメントに似ていて生成されたコード内に残される

拡張正規表現

- 拡張正規表現は `///` で区切られ、ヒアドキュメントやブロックコメントに似ている
- それらは内部の空白を無視し、コメントに含むことが可能

エーリアス

```
and : &&    or   : ||    not  : !
is  : ==    isnt : !=
yes : true   no   : false
on  : true   off  : false
```

雑記

- Twitter コメントは `@nextliteracy` まで

<http://autotelicum.github.com/Smooth-CoffeeScript/>