The Julian Way

Learning to think in Julia

Stefan Karpinski

Topics

object-oriented *vs.* multiple dispatch

built-in types *vs.* user-defined types

type parameters *vs.* field values

object-oriented *vs.* multiple dispatch

object-oriented *vs.* multiple dispatch

Rules of thumb:

- x.f(y) $\Longrightarrow$ f(x,y)
- classes $\Longrightarrow$ types
- namespaces $\Longrightarrow$ modules
- namespaces = class $\Longrightarrow$ flatten

object-oriented *vs.* multiple dispatch

Example:

- Bloomberg API

object-oriented *vs.* multiple dispatch

Take-aways:

- encourages factoring out common “verbs”

  c.f. Steve Yegge’s “Kingdom of Nouns” essay [1], about o-o languages

- results in flatter hierarchies

  modules are largish pools of related nouns and verbs

[1] http://steve-yegge.blogspot.com/2006/03/execution-in-kingdom-of-nouns.html

built-in types *vs.* user-defined types

built-in types *vs.* user-defined types

Rules of thumb:

- classes $\Longrightarrow$ types

  maybe dicts?

- also consider transposing the representation:

  vector of objects $\Longrightarrow$ bundle of “primitive” vectors

  maybe data frames?

- representation of built-ins is good but behavior isn’t?

  you might want a thin wrapper type

built-in types *vs.* user-defined types

Examples:

- simple vector of objects transposition
- k-ary tree representation
- EnvHash
- Images

built-in types *vs.* user-defined types

Take-aways:

- scripting languages, C & Fortran encourage using built-in types
- object-oriented languages encourage user-defined classes
- Julia lets you do either – there is no single correct answer

  this makes it harder to decide, but you can get a better fit

  try using built-ins first, see how it goes

  use custom types if it simplifies / improves

type parameters *vs.* field values

type parameters *vs.* field values

Rules of thumb:

- do you want to dispatch on it?
- do you want specialized code generated for it?
- are arrays likely to be homogenous or heterogenous wrt this?

type parameters *vs.* field values

Examples:

- SIUnits
- ModInt
- time zones
- fizz buzz

type parameters *vs.* field values

Take-aways:

- it's easy to go a little hog wild with Julia's type system at first
- err on the side of simplicity

  values are usually better than parameters

  most popular languages don't even have parametric types
- think about C++ templates

  decent mental model for specialization & code generation