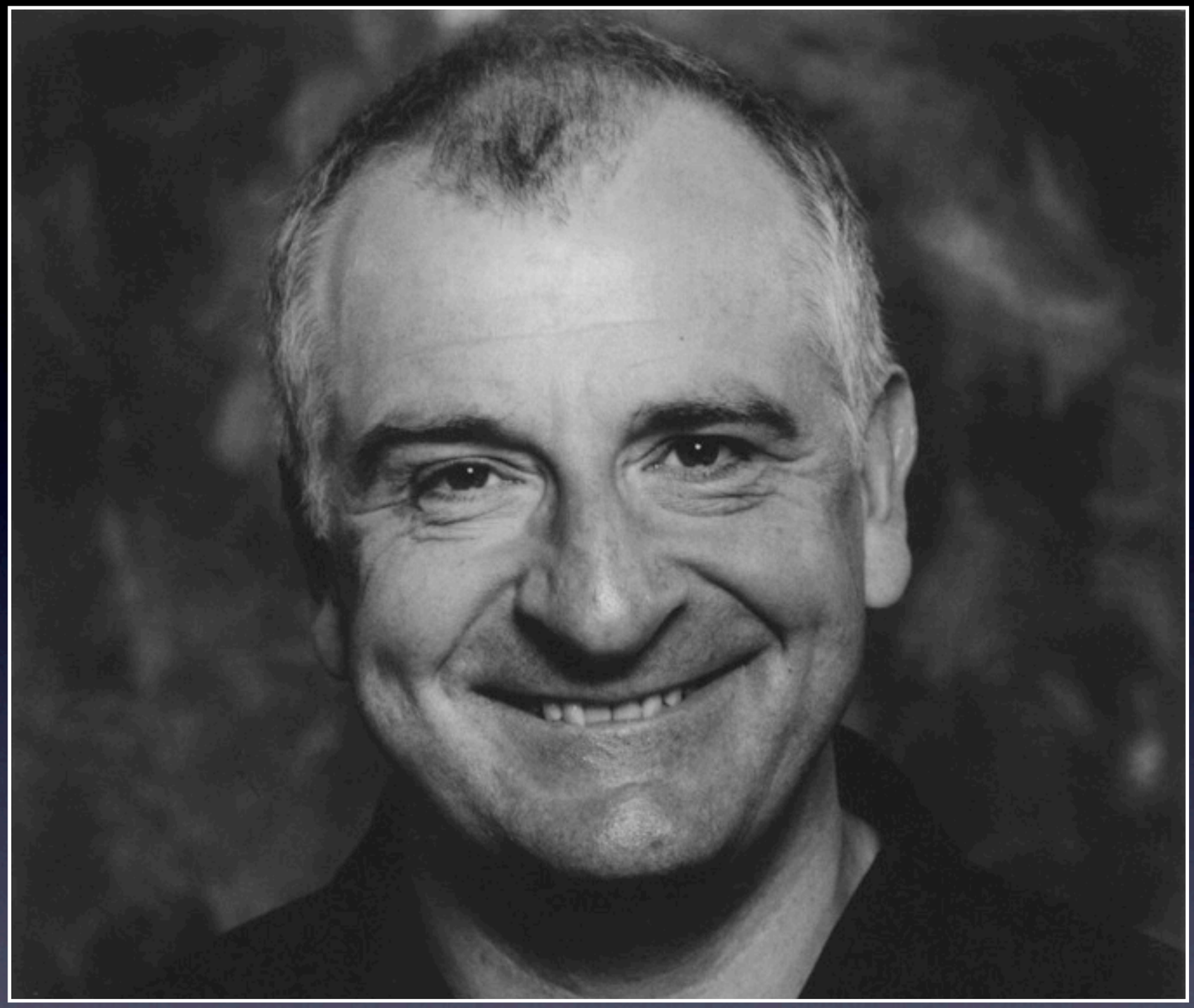
The Data-Reader's Guide to the Galaxy

Steve Miner
@miner

The Data-Reader's Guide to the Galaxy

Steve Miner
@miner

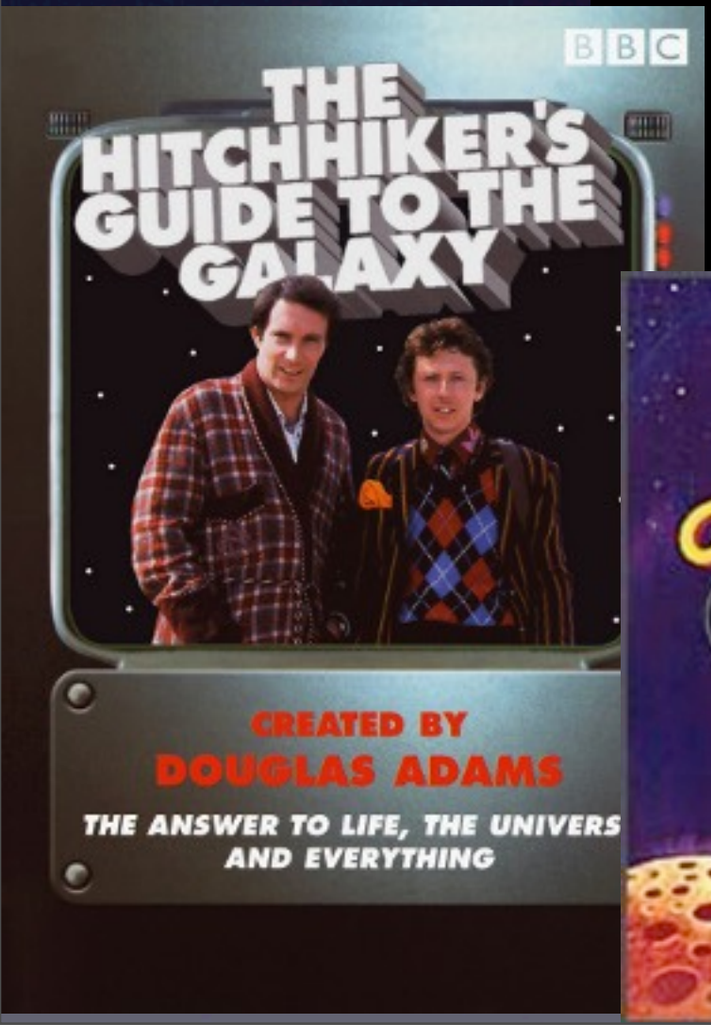
#inst "2013-03-18T09:50-07:00"



Douglas Adams


The Hitchhiker's Guide to the Galaxy
RADIO SHOW
LIVE!


DON'T LEAVE EARTH WITHOUT IT
THE HITCHHIKER'S GUIDE TO THE GALAXY
DON'T PANIC.


BBC
THE HITCHHIKER'S GUIDE TO THE GALAXY
CREATED BY
DOUGLAS ADAMS
THE ANSWER TO LIFE, THE UNIVERS AND EVERYTHING


The Hitchhiker's Guide to the Galaxy
DON'T PANIC!

The Guide

- *The standard repository for all knowledge and wisdom*
- Basics
- New in Clojure 1.5
- EDN
- Unorthodox ideas

Tagged Literals

- #my.ns/tag literal-data
- #inst “2013-03-18”

- Self-describing data
- Loosely coupled to implementation
- Data transfer

Extensible Reader

- Add new data literals
- Customized to application
- Limited form of CL reader macros
- Read-time vs. Compile time

data-readers

- data_readers.clj
  - literal map
  - tag symbols to var symbols
- (binding [*data-readers* {...}] *body*)

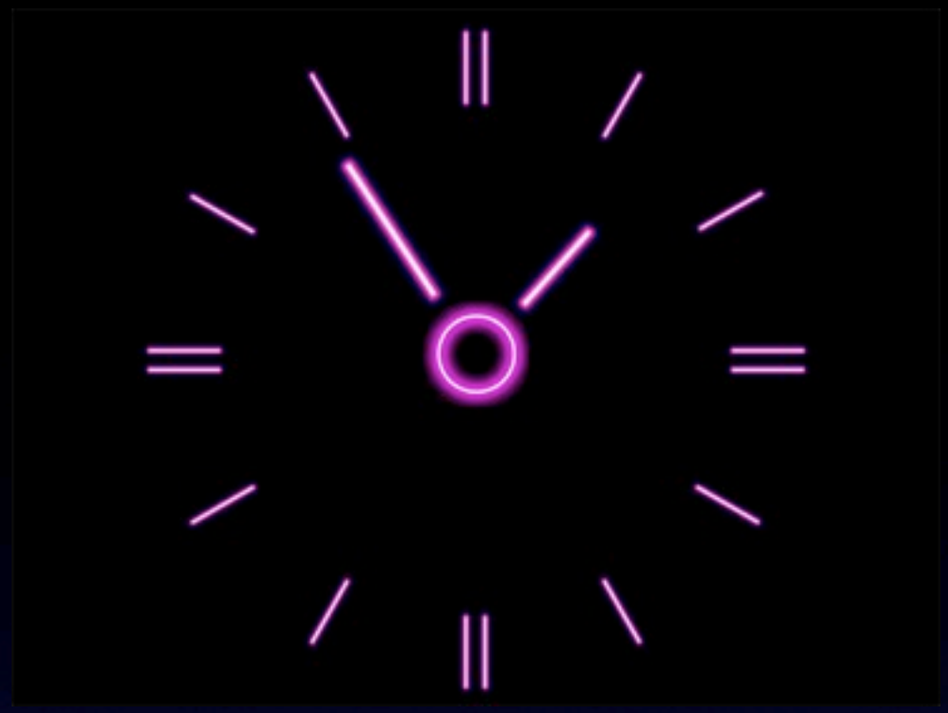
```
(defn my-reader [[a b]]
   (+ (* 10 a) b))

(binding [*data-readers*
          {'my.ns/tag #'my-reader}]
   (read-string "#my.ns/tag [4 2]"))

;=> 42
```

default-data-readers

- pre-defined by Clojure
- #uuid
- #inst



Time is an illusion.

Lunchtime doubly so.

#inst

- Instant in time per RFC 3339
- #inst “2013-03-18”
- #inst “2013-03-18T09:50-07:00”
- #inst “2013-03-18T16:50:00.000-00:00”

Date/Time

- java.util.Date
- java.util.Calendar
- java.sql.Timestamp
- Joda Time - clj-time
- JSR 310 - java.time in JDK 8

#uuid

- #uuid
  "4e0f694b-1901-4886-9101-5479fc0c9720"
- (java.util.UUID/randomUUID)

Examples

- #x/url “http://clojurewest.org”
- #x/base64 “ZnJvbSBvdGhlciBhbmltY”
- #x/coords [45.5241, -122.6820]

default-data-reader-fn

- new in Clojure 1.5
- defaults to nil, throws on unknown tag
- fn receives a tag (symbol) and a value
- should return a literal value

Example *ddrf*

- (defrecord TaggedValue [tag value])
- print-method TaggedValue
  - #tag value
- *default-data-reader-fn*
  - (->TaggedValue tag value)

Records vs. Tags

- record: #my.ns.Rec{:a 42}
- tagged: #my.ns/Rec {:a 42}

- *default-data-reader-fn*
  - factory for tag: my.ns/map->Rec
  - maybe check tag ns or Capital

```clojure
(defn record-tag-factory [tag]
  (resolve (symbol (str (namespace tag) "/map->" (name tag)))))

(defn default-reader [tag value]
  (if-let [factory (and (map? value)
                        (Character/isUpperCase (first (name tag)))
                        (record-tag-factory tag))]
    (factory value)
    (->TaggedValue tag value)))
```

Library Authors

- #my.ns/tag semantics
- Base literal value
- Implementation types
- Data-reader functions
- Print methods (maybe)
- data_readers.clj (maybe)

Printing

- See instant.clj
- Clojure prints using #inst for ju.Date, ju.Calendar and js.Timestamp
- *print-dup* ignored

Gotchas

- CLJ-1138 returning nil throws.
  - Return '(quote nil) instead.
- CLJ-1100 period in tag throws.
- Java Dates are mutable

```
(= #inst "2013-03-18T16:50Z"
   #inst "2013-03-18T09:50-07:00")
;=> true
(set! *data-readers* {'inst
#'clojure.instant/read-instant-
calendar})
(= #inst "2013-03-18T16:50Z"
   #inst "2013-03-18T09:50-07:00")
;=> false
```

read-eval Kerfuffel

- Ruby on Rails vulnerability
- Clojure ***read-eval*** true by default
- #=(dangerously) can execute code
- not safe for reading untrusted data
- CLJ-1153 and CLJ-904


DON'T PANIC

clojure.core/read

- Designed by *hyper intelligent, pan dimensional beings*
- **read** is for trusted input
- binding ***read-eval*** false
  - (pre 1.5) allowed Java constructors
  - #my.ns.Rec[42]

A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.

clojure.edn

- new EdnReader in Clojure 1.5
- no #=()
- no Java constructors, no records
- just safe EDN elements

clojure.edn/read

- arities - [ ] [stream] [opts stream]
- opts map - :eof, :readers, :default
- defaults to *in* and default-data-readers

clojure.edn/read-string

- arities - [string] [opts string]
- opts map - :eof, :readers, :default
- defaults to default-data-readers

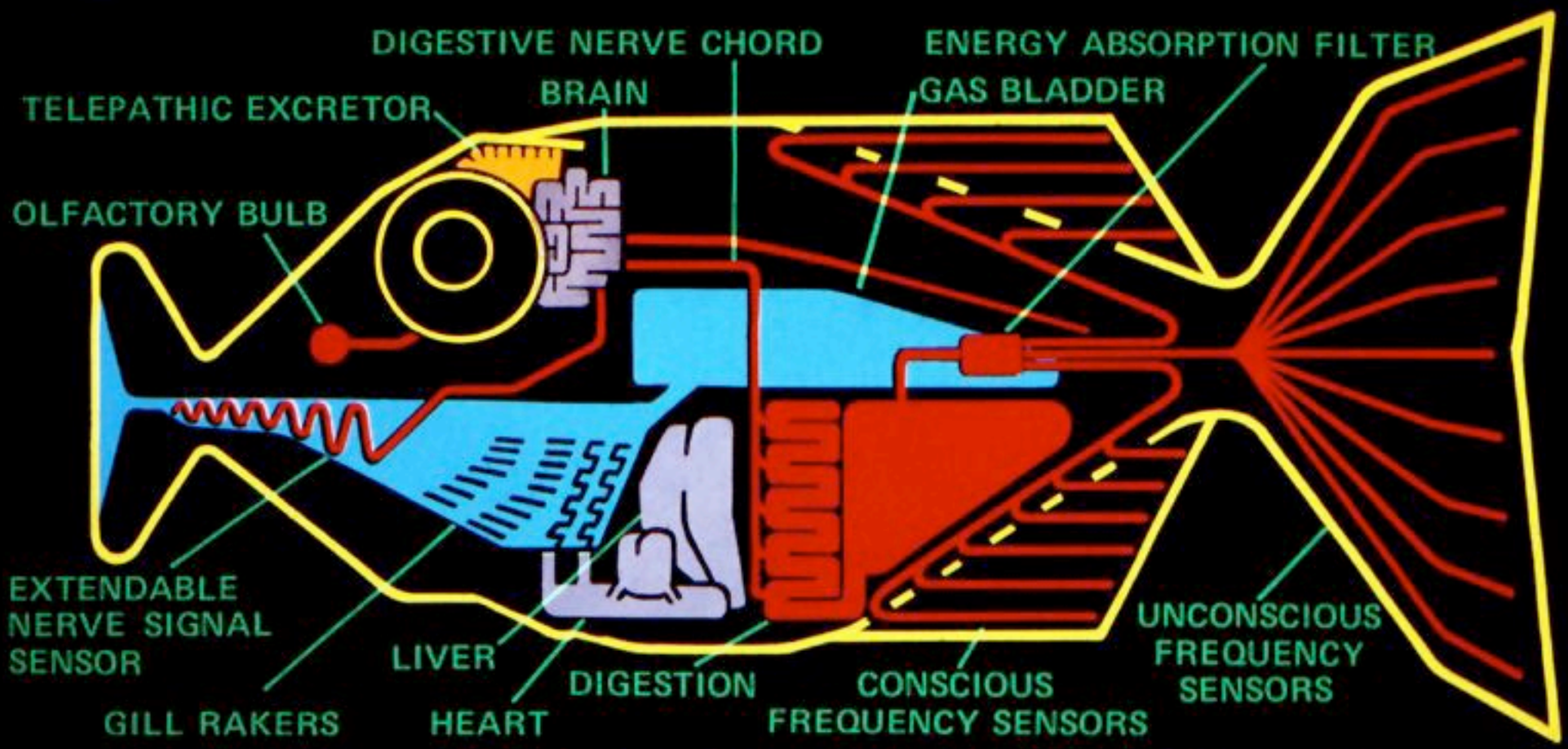
clojure.tools.reader

- Written in Clojure
- A complete Clojure reader
- An EDN-only reader
- Works with Clojure 1.3+

EDN

- Extensible Data Notation
- edn-format.org
- Clojure style values: { } [ ] ( ) sym :kw
- #tagged literals
- Implementations for other languages

BABEL FISH



THE BABEL FISH IS SMALL, YELLOW, LEECHLIKE,
AND PROBABLY THE ODDEST THING IN THE UNIVERSE.
IT FEEDS ON BRAIN WAVE ENERGY, ABSORBING ALL

The Guide on XML

In the beginning XML was created. This has made a lot of people very angry and has been widely regarded as a bad move.

XML

- XML 1.0 (1998) - working group of 11 experts and an interest group of 150 within W3C
- The Encyclopedia Galactica of data formats
- <xml /> is (s-expr) with better marketing

JSON

- "The Fat-Free Alternative to XML"
- "The good thing about reinventing the wheel is that you can get a round one." - Douglas Crockford
- not extensible

EDN vs. JSON

- extensible
- more types
- a little more syntax
- conveyance of values, not objects
- *slightly cheaper*

Unorthodox Ideas

- #x/roman
- #x/rpn
- spyscope
- #feature/condf

```
(defn roman-numeral-reader [s]
  (parse-roman (name s)))

#x/roman “XLII”
;=> 42

#x/roman XLII
;=> 42
```

```
#x/rpn [3 4 + 8 2 - *]
; compiler sees: (* (+ 3 4) (- 8 2))
;=> 42
```

Spyscope

- github.com/dgrnbrg/spyscope
- #spy/d (form)
- better than println
- minimally invasive

```
; Conditional feature reader
; github.com/miner/wilkins

(println #feature/condf
   [(and jdk1.6+ clj1.5.*) "reducers OK"
    else  "Don't push that button"])
```

Summary

- #tagged literals are mostly harmless
- clojure.edn is your safe place
- data-readers are open to crazy ideas

Towel Day

- *A towel is about the most massively useful thing an interstellar hitchhiker can have.*
- Tribute to Douglas Adams
- towelday.org - 25th of May
- *There's a frood who really knows where his towel is.*

The Data-Reader's Guide to the Galaxy

Steve Miner
@miner

#inst "2013-03-18T10:30-07:00"

So long and thanks for all the fish.

Extras

Google

- The answer to life the universe and everything
- Douglas Adams doodle

History of every major Galactic Civilization

- Phases: Survival, Inquiry and Sophistication
- How can we eat?
- Why do we eat?
- Where shall we have lunch?

Programming

- Phases: Survival, Inquiry and Sophistication
- How can we execute code?
- Why do we write code?
- Where shall we host our code?

Marvin

I'm fifty thousand times more intelligent than you and even I don't know the answer. It gives me a headache just trying to think down to your level.

Earth

An utterly insignificant little blue green planet whose ape-descended life forms are so amazingly primitive that they still think XML is a pretty neat idea.