Minutes of the:  Ecma TC39-TG1
held in:  Phone conference
on:  3rd May 2006

Attendees

Meeting time 10am PDT.

On phone:

- Francis Cheng, Adobe Systems
- Jeff Dyer, Adobe Systems
- Gary Grossman, Adobe Systems
- Ed Smith, Adobe Systems
- Pratap Lakshman, Microsoft
- Brendan Eich, Mozilla Foundation
- Graydon Hoare, Mozilla Foundation
- Blake Kaplan, Mozilla Foundation
- Dave Herman, Northeastern University
- Lars Thomas Hansen, Opera Software
- Cormac Flanagan, UC Santa Cruz

Agenda

Note new meeting day of week for phone conferences.

- package semantics
- multiple compilation units
- formal type system
- other hot topics

Discussion

- package semantics
  - packages declare two namespaces, p.q#public and p.q#internal
  - import p.q is in part like use namespace p.q#public
  - expression starting p.q.x is rewritten to p.q#public::x
    - even within package p.q, p.q.x is rewritten, so x must be public
    - internal::x or just x would work unless ambiguity requires full path

- multiple compilation units
  - Dave: what if you have package p.q with an x use but no x def; now add x to p.q after the compiler dealt with the first x use
    - Jeff: package compilation ends at verification or loading
    - Dave: so packages do involve separate compilation units
Lars: see multiple compilation units for browser constraints

Ed: AS3 ignores redefinition
- two programs define utilities, pgm1 has A&B, pgm2 has B&C
- they load in the same runtime, in that order
- AS3 assumes first B is same as second B

Brendan: browsers and ES1-3 of course have writable function bindings
- so last one wins
- can we do better than last-wins for global functions or first-wins for classes and packages?

Ed: hard to share common utility packages without getting too fine-grained

Dave: what are use-cases for splitting up packages into multiple pieces?
- Jeff: Java examples to avoid overlarge files
- Dave: easy to unify that case at load time
- Ed: package with hash table and tree
  - pgm1 uses hash table
  - pgm2 uses tree

Dave: why are those in the same package? Ok, pick a better example

Ed: explored a signature checksum scheme to verify Bs don’t conflict

Ed: another example: graphing components for charting
- also accessibility addons to the charting package
- want accessibility stuff in a separate compilation unit

Brendan: first one wins is going to be hard to beat

Ed: Java does that within a classloader

Lars: anyone-wins is going to break on the web
- Brendan: yeah, many <script src> cases are like #include, some are more like block-scoped import

Ed: Flash has application domains outside the AS3 language
- you can create a sub-domain to isolate effects
- lookups start with super-domain then go to sub-domain

Pratap: CLR2 has app domains too

Gary: Flash took inspiration from that, similar

Dave: shadowing is not mutability

Brendan: browsers name modules by URI, so no subversion via shadowing

Ed: packages are namespaces are URIs, so do tie into security and http caching

Graydon: content hashing better than relying on DNS

- **formal type system** questions
  - String to Boolean
    - no controversy on if, while, for, &&, ||, ! converting
    - var x : Boolean = "hi"
    - Jeff: that converts in AS3 in bang or tilde
      - compatibility requires this without type annotations
      - Brendan: could be stricter
      - Jeff: refactoring hazard
      - Cormac: tradeoff between type-checking and convenience/migration
  - Return from constructors
    - Brendan: ES1-3 allow function constructor to return a different object
    - Ed: class constructor functions cannot return expr; at all
      - but class ctors can return; to bail early
      - Brendan: different from rule in functions
      - Dave: how does type system talk about type of constructor?
        - so could allow constructors to have Void return type
        - and they could even contain return void 0 or whatever
      - Jeff: in AS3, function f():void(...) means ... cannot return expr;
      - Dave: type Void means can’t return a value
Brendan: then need type Undefined too
  ▪ Dave: need to review proper tail calls in light of this
  ▪ Ed: try this:
    ▪ Void is type, has value undefined
    ▪ f():void implies extra syntax restriction against return expr;
      but otherwise doesn't affect type-checking, proper tail calls, etc.
  ▪ Dave: concerned about need to name Undefined or Null in unions, etc.
  ▪ Brendan, Ed: need special restriction on return expr; for
    ▪ constructors
    ▪ setters
    ▪ generators
  ▪ with discussion
    ▪ Using annotations and structural types, one can finally state the precise type of the
      object named in the with head
    ▪ let declarations in body of with work as elsewhere
    ▪ Apart from these orthogonal goods, can we reform with, or banish it?
      ▪ use strict could banish it to a { use dynamic; ... } block
      ▪ does this really help? migration vs. new code, why do people use with?
  ▪ Global object unknowns
    ▪ Brendan: which prototype proposes immutable String, etc.
    ▪ intrinsic::String vs. String wouldn't differ if we adopt that proposal
    ▪ s.intrinsic::charAt(i) would be different from s.charAt(i) for backward
      compatibility, to support AOP-ish hacking

class String {
    . . .
    intrinsic function charAt(i:uint):String {...}
    prototype function charAt(i:*):String {...}
}

- intrinsic proposal
  ▪ Use intrinsic instead of something like AS3 or ES4 for early binding.