Minutes of the: Ecma TC39-TG1
held in: Mountain View (Mozilla)
on: 21st September 2006

Attendees

- Jeff Dyer, Adobe Systems
- Steven Johnson, Adobe Systems
- Dan Smith, Adobe Systems
- Edwin Smith, Adobe Systems
- Michael O'Brien, Mbedthis
- Brendan Eich, Mozilla Foundation
- Graydon Hoare, Mozilla Foundation
- Dave Herman, Northeastern University
- Lars Hansen, Opera Software
- Iaian Lamb, Yahoo!
- Julien Lecomte, Yahoo!

Agenda

- eval
- yield
- proposals review

Notes

- eval
  - resurrected_eval
  - method on global object
  - do we need eval(s, [oN, ..., o1])
    - where o1 is the head of the replacement scope chain

- global self-name
  - globals
  - just a property name, can be bound by sandboxing code as it wishes

- yield
  - should make yield e and let (h) e use the same nonterminal for e
  - either: over-parenthesize
  - or: non-terminal for e is AssignmentExpression
  - resolved: use AssignmentExpression

- documentation comments
  - lars: why not use a comment?
  - doug’s proposal does not reflect as doc
  - how does this relate to decorators?
- need a unified proposal
- want real-world use-cases so we don’t miss anything that should be ES4

- dave/brendan shorter/more-compositional function expression forms
  - let function f(args) { body } © let f(args) { body } OR let f(args) expr

- normative grammar issues
  - we have resolved to eliminate reference types in the spec
  - should we restrict optional reference types in the grammar? no
  - should we make the spec’s normative grammar be LL(1) or LR(1)? yes
  - jeff to take on formalizing the grammar, mob will help

proposals review

- **type parameters**
  - class A.<T> extends T { ... } should not be allowed
  - similar such questions may arise
  - dave: C# type non-erasure vs. Java erasure

- **builtin classes**
  - lacks intrinsic, jeff to update

- **structural types and typing of initializers**
  - do we allow any TypeExpression after : in initialiser annotation? yes
  - do we allow any Expression after : in initialiser annotation? uhhh...
    - dave:
      - we want static type checker recognizing static constraint
      - casts for dynamic constraints (less common)
      - more common static constraint case should have lightweight syntax
      - agreement on these two points:
        - want {p: 42} : {p: int} where the annotated type is a TypeExpression
        - want cast T (E)
      - what about to?
        - want x to T where grammatically T is TypeExpression
        - confusion about foo().to(x) being backwards - should be from?
        - entertain proposals in the wiki for nice dynamic-to/is API syntax
      - resolved: want infix operator syntax for static case: TyExpr on right
      - what about is?
        - alternative is to match to and require static TyExpr on right
        - allowing any Expr means structural types must be named to be used
        - if we require TyExpr on right of is, we may break AS3 users

- **is as to**
  - in good shape apart from wiki page title
  - dave to update based on recent type system work and previous item

- **nullability**
  - agreement on nullability by default
  - discussion brought up need to:
    - update the spec before re-exporting
    - respond to es4-discuss list with pointers to new export

- **numbers**
- in good shape now (see recent numbers)
- mob is doing int64 as extension; seems to fit

- **strict and standard modes**
  - raised issues of spec language and completeness
  - build on E3 or try to improve it w/ a significantly different metalang?
  - take E3 metalang and clean it up a bit (a la ECMA-357)
  - dave to try writing a few more accessible spec styles for some productions

- **normative grammar**
  - see above

- **intrinsic namespace**
  - in good shape, foundational

- **type refinements**
  - move to deferred

**proposals, continued**

- enumerability
- switch type
- block expression
- proper tail calls
- type definitions
- syntax for type expressions – fix ? to be postfix
- **namespace shadowing**
  - all good

- **iterators and generators**
  - StopIteration is of type StopIterationClass
  - intrinsic::iterator must become iterator::get or some such

- resurrected eval

- expression closures, still reviewing
- multiple compilation units - need to prove the two propositions
- security wrappers - does it do enough to be worth its cost?
  - meaning: does enough? costs a lot?

- issue with intrinsic::global
  - is it bound to the caller’s global in the “sandbox” (sic) object passed to the resurrected eval?
    - lars said yes earlier, brendan said no; revised answer is no.

- catchalls, hashcodes, operators, destructuring
  - All good
  - Note to self:

    ```
    for ([k] in o) => SyntaxError
    for ([k,v,u] in o) => SyntaxError
    for ([k,,] in o) => ok
    for ([k,] in o) => ok
    ```
for ([k,v] in o) => ok

- **bug fixes**
  - brendan: remove eval bug fixes, it has its own page
  - brendan: generic statics for Array and String should be split out
  - jeff: escaped newlines in string literals ok

- **decimal**
  - graydon: pragma syntax update
  - otherwise looks good

- **typeof**
  - update to leave typeof null === ‘object’
  - update to change typeof class === ‘object’
  - BUT: typeof String === ‘function’ for backward compat
  - informative words expressing regret

**day two**

- **syntax for pragmas**
  - ok

- **reserved words**
  - should we do as js1.7 and allow function delete? no
  - should we allow reserved identifiers after ::? yes

- **update unicode**
  - ok (discussion around clarity of implementation choice, how choice is one way or the other for all inputs, depending on input).
  - resolved: format control chars are not stripped from source input, therefore are preserved in string and regexp literals

- **extend regexps**
  - Updated to note per yesterday’s discussion that typeof /re/ === “object”.
  - Also adopting the IE quirk that Opera and Mozilla do: /[1]/ matches “/”.
  - This means that #... line comments in /very-long/x regexps must balance [].

- **slice Syntax**
  - Brendan to clean up, move most to discussion, present minimal proposal
  - Iain: why not define a range generator function?
  - Discussion about +, <, == etc. for Array – put them in a new namespace that new code can use: use namespace operators.

- **triple quotes**
  - still good

- **documentation**
  - move to reflection library
  - use javadoc style comments (precedent: asdoc tool from adobe)

- **globals**
  - singularize intrinsic::globals
• **date and time**
  o all good but nanotime:
    ▪ discussion about accuracy needs – want delta-t for benchmarking, really
    ▪ so don’t need nanoseconds, or want to impose them on all impls
    ▪ ptw’s tick/tickScale proposal from es4-discuss considered too hardware-ish
      ▪ not good if tickScale isn’t constant; if constant, it may have to be too large a
        number of nanoseconds in order for tick to be cheaply computed
  o dave: social psych reaction time research
  o lars/graydon: use nanoseconds since creation of Date object: d.nanoAge()
  o nanoAge to be drafted

• **json encoding and decoding**
  o Array.prototype.toJSONString
  o Object.prototype.toJSONString
  o String.prototype.parseJSON
  o String.prototype.trim (free-riding on JSON here)

• the module perplex – packages don’t solve naming and loading issues

• **stack inspection**
  o good

• **meta objects**
  o lars/dave: namespace() and name() should be in ClassType not Type
    ▪ ditto for supertypes() and subtypes() (rename to super/subClasses())
  o dave: use iterators instead of arrays
  o dave: note to use [T] instead of no-longer-proposed Array.<T>
  o dave: should reflect public methods and fields, structural fields, etc.
  o InterfaceType? sure; InterfaceType.implementedBy()

• **expression closures**
  o good, clean up discussion

• **multiple compilation units**
  o needs exact and complete list of differences between models

• **security wrappers**
  o graydon and brendan: come up with use cases

• iain: version reflection? object detection and try-eval rule
• brendan: dict syntax for null-proto object initialisers

• TODO:
  o brendan: slice
  o iain: json encoding and decoding, trim
  o lars: documentation
  o graydon: date and time
  o graydon and brendan: security wrappers