| | |
|---|---|
| ***Minutes of the:*** | ***Ecma TC39-TG1*** |
| ***held in:*** | ***Mountain View, CA*** |
| ***on:*** | ***21st-22nd of June 2007*** |

## Logistics

- June 21, 1200-1700 PDT, with dinner at 1830, at Sundance Mining Co. (regrets only)
- June 22, 1000-1700 PDT
- Mozilla building "S"
  - 2121 Landings Drive, Mountain View, CA 94043 (map link)
  - `<script>` conference room

## Attendees

- Francis Cheng, Adobe Systems
- Douglas Crockford, Yahoo!
- Jeff Dyer, Adobe Systems
- Brendan Eich, Mozilla Foundation
- Cormac Flanagan, Univ. California, Santa Cruz
- Lars T Hansen, Adobe Systems
- Graydon Hoare, Mozilla Foundation
- Blake Kaplan, Mozilla Foundation
- Jason Orendorff, Mozilla Foundation
- Chris Pine, Opera Software
- Alex Russell, Dojo
- Allen Wirfs-Brock, Microsoft

## Agenda

- type system issue: bounded length array type, semantics and syntax
- self type: at least return-this self-typing is wanted by iterators and generators
- Open proposals and open proposal issues in the es-lang.org trac.
- Decision on whether to adopt an environment model of type expressions, including type closures.
- Decision on when to resolve type names to type values, and whether to treat "strict mode" as partial evaluation.
- Decision on how much environment to share between compilation units / modules. Reconcile program units and multiple compilation units (if there is any disagreement between them).
- Clean up all wiki pages and source code that disagrees on these decisions, so we have a single story!
- Need another wiki export
- Phone meetings between now and the next f2f?
- Spec writing process

## Minutes

### Open proposals

#### Maintenance of ES3

Ongoing, no update. We will be producing more wiki material as time goes on. The conditional compilation takes precedence.

- JD: Should discuss compat issues like octal.
- BE: We decided to let octal die and place discussion of it in an informative annex. There are a few other non-standard additions, we should decide whether to make these normative.
- AWB: I can promise that the feature list will be ready by the next f2f meeting.
- BE: We can't change Mozilla JS to match JScript because of incompatible control flow forking based on `isIE` tests.
- AWB: We could id those issues and say that they are impl. dependent.
- JD: Octal interesting because there is an ISO desc, but it's buggy.
- BE: May be best to just say octal is impl dep.

#### Enumeration type

- JD: Would like to defer this. Is the IE impl what we want? Don't see a sum type in the proposal, which doesn't seem so useful.
- BE: Useful for ensuring that you don't miss a case statement in a switch.
- LTH: No way right now to check for missing case statement.
- GH: The switch type proposal is supposed to handle that, though it's a long-winded way compared to enum.

**RESOLVED**: No one is opposed to deferring this, so this is now deferred.

#### Self Type

On the agenda elsewhere. We'll save discussion for later.

#### Remove arguments object

Deprecation was discussed. Lars thinks that's worse than doing nothing.

- BE: widespread use.
- LTH: powerful statement that arg object should die. I think it's a good idea, but there are uncertain implications.
- BE: Still in favor of removing. Move it to an informative annex. Would have to still be there for legacy content, but would prefer that it be non-normative.

Implementations that deal with web apps will have to continue supporting it, look for info in informative annex.

- JD: We should discuss whether we want to approve 'this function' and 'this generator'. I originally resisted because it overloads the meaning of 'this', but I may be willing to live with this.

- AWB: It's clear what it means inside a nested function, but it's inconsistent with 'this' generally.
- LH: 'this' really means 'this receiver' in this case.
- JD: this is already a primary expression, so adding function and generator works.
- AWB: Another way is in every activ object, there's an implicit variable.
- LH: Problem is that you could shadow user variables.
- BE: We did something like that, callee, but it had non-trivial implementation costs.
- AWB: Proposal doesn't say what happens to the arguments object. Should explicitly state what will happen.

**RESOLVED**: Accepted, will move to informative annex. Clearly state the better way to do it.

## Eval

- From crunched Google web app (and other sites) scripts, paraphrasing:

```
function f(p) {      var Da = eval    var x = 42     ...     Da(p)      }
```

p cannot refer to local variables (i.e. p cannot see x).

- BE: mandate `EvalError` be thrown for indirection thru a property ref other than a window object.
- IE does a dynamic scoping to activation object for nearest active function from same window! (see resurrected eval)
- LH: Should be statically scoped to global object of the iframe, no dynamic scope at all, for explicitly indirect call.
- AWB: Unlikely that IE would change this.
- BE: I'll write up the proposal so it includes the discussion about IE's behavior. We have multiple global objects now, we'll discuss this later.
- AR: what about the optional second argument?
- BE: Trying to keep eval from being too impl defined.
- LH: Need to provide a stronger sandbox to support second object that specifies object.
- BE: Question is the object arg added to the scope chain, replaces scope chain or isolated?
- DC: 3.1 proposal isolates it. Makes eval also a String.prototype method.

```
String.prototype.eval([extra])
```

- LH: Can still leak if you're not careful.
- DC: If you only pass in strings and numbers to the env, it should not leak.
- BE: We should consider String.prototype.eval() aside from the 3.1 proposal.
- LH: How does it work in a browser env? Does it have access to the window's global object or is it in a new pristine env?
- DC: It's in a new pristine environment.
- BE: What do people think of adding the String.prototype.eval()? Might want a different name.
- AR: So Gears isolates worker thread evaluation like this. Seems like a useful way to sandbox and get data in and out.
- LH: So Doug's point is that you can have communication if you pass in an object that can share data.
- JD: So the String prototype eval is a separate proposal and needs its own proposal.

**Resolved**. resurrected eval in the compatible-with-renamed-eval-that-can't-see-local-vars sense is approved.

### Program configuration

- BE: Not clear to me that you need to have diff versions of code in one script tag.
- AR: It would be useful to allow backward and forward compat code in a single file.
- DC: And lacking this it prevented us from using try/catch for five years.
- AWB: Talking to major web devs, they don't want to deal with new features in an unconditional way.

Comment hiding is one way to allow web devs to shield incompat browsers from unsupported features. Primary issue is do you have a mech that allows dev to write code for diff versions in same file.

- BE: Even with @if, in many cases, you would have to write your code twice, so not sure it is that useful. Good example is try-catch, you'd have to rewrite the try-less version, you can't factor out exception handling syntax using functions:

```
@if ES3     use  try-catch @else     ...
```

- JD: Would like to move program config to ed. 5.

**RESOLVED**: Deferred program configuration to future edition.

### Program units

- BE: We've talked about this idea for a long time. We hear from AJAX folks that packages don't give people a module system. So a lot of developers have to invent a module system.
- AR: It's almost all complexity and performance. I need them and I need them to go fast. I'm conflicted about this because it makes it easy for people to shoot themselves in the foot.
- AWB: I like this but our Windows Live guys would say we won't use it because of existing tools in use that do it server-side.
- DC: Happening on wrong side of the network. Hazard here that it would be abused on the client side.

So we're developing a special server-side language that gets stripped out by the time it gets to the client.

- GH: Don't see a downside to have a canonical way to compress.
- DC: New dev is widget based pages where server won't know what would be loaded later. Not sure this proposal affects this.
- GH: Don't think so.
- GH: We have mult compl units proposal, we need to reconcile that proposal with this one, so we have a unified set of terminology.
- JD: Regardless of the syntax, we need to pin down the semantics.
- BE: JD and GH should work this out.
- GH: term module is fine with me. We also need to figure out when names get resolved.

**COFFEE BREAK**

### Issues related to module loading

```
class C extends D { }     var x:T
```

- GH: Some of these types of dependencies imply a load order. Other cases include:

```
ns::y use namespace ns C.<T> // body may require T
```

Issue is that strict and standard modes have different dependencies. Can think of strict as a partial evaluator.

Example:

```
var x:T = y; // y needs to be bound
```

- CF: Should strict and std mode have same load order?
- GH: This should definitely be true.
- AWB: Is load a runtime or compile time process?
- GH: Preferred approach to static analysis in strict mode is partial evaluation.

```
use module foo "http://..."   module foo {    ...  }
```

- GH: Does a module define a boundary that requires a certain level of static analysis or partial evaluation?

Maybe nothing but ES3 behavior.

- BE: Has to be at least ES3 behavior (section 10 “Execution Contexts” rules for binding functions, binding vars, then evaluating code).
- GH: Cyclic classes are nonsense. Two classes can reference each other but not extend each other. Verifier allows cross references.
- JD: what about across modules? AS3 chose not to allow this.
- AWB: Some would argue that two modules are dependent on each other, you really have one module.
- LH: That’s not necessarily true, it’s all about how you want to decompose your application.

Note: module can be the same as a file.

- GH: Nested modules can mutually recur.
- AWB: Modules can reference each other, but it’s only an issue when the modules are brought together into a mutual scope and the references are bound.
- LH: Modules is not a good name because there’s no interface. Unconvinced that this will reduce bandwidth consumption. Restriction that packages need to be at top of compilation unit should be dumped.

- JD: So what would have to be determined at module/unit boundary?
- GH: Open for discussion, possibly all types in strict mode?

- AWB: Given Server Side expansion, is it possible to have in some scope two uses of a unit both of which have been physically expanded? Any restriction on the binding environment being same for both?
- GH: Currently, the second one would just be ignored.

- LH: No way to guarantee a reasonable number of expansions?
- GH: Can’t make that guarantee.
- GH: One add’l point: we should have a way of saying, use this module but as late as possible.
- AWB: What if we did lazy load on first use, though not sure how to do that.
- GH: example of caching issue:

```
use unit todaystocks "http://finance.yahoo.com/q?...";
```

- AR: is there a way to call ‘use unit’?
- BE: Sure, eval can do it and tack a random string on it. Have to do this unless we do no-cache.

### Dictionary

Need to defer to next month. No work done on it recently.

### Trac issues database

- Set milestones for all issues marked as 'proposals' issues that did have milestone set.
- Issue: Should deprecated features be implemented in the Reference Implementation (RI)? We've already pulled octal out, but should we pull arguments object out of the RI? We would have to rewrite some of the tests in the test suite.

## Friday, June 22

### Trac issues database continued

- Continued reviewing Trac issues marked as 'proposals'.
- Ticket # 73: The meaning of null-testing for 'is', and syntax for its type expression: Resolved that is, to and cast will respect null in rhs type. For example:

```
null is ()  // false null is null // true null is Object // true null instanceof
Object // false because instanceof walks the prototype chain, not type
hierarchy.   type T = int? null is T          // true
```

- Ticket #71: Strucural object/array/function(?) types should be implicitly nullable in annotations: Resolved that types shall be implicitly nullable in annotations. In other words the following three statements are equivalent:

```
[...] [...]? ([...], null)
```

and you need

```
[...]!
```

to get non-nullable structural array type.

- Related discussion about structural types: users should have the option to make structural types dynamic via trailing `*` before right bracket. A ticket will be filed on this.

| | **non-dynamic** | **dynamic** |
|---|---|---|
| Array | [int] | [int,*] |
| Object | {p:int} | {p:int, *} |

- Ticket #69: Support for bounded arrays:

Min bound already implied by type system:

```
x=[1, "x"]:[int,String,*]; x.length = 1; // Problem here because this would
subvert the type system
```

Lars will rewrite ticket to include min bound as well as max bound. Result of Ticket 120 affects this issue, defer until 120 is resolved.

- Ticket #68: Intuitive syntax for the creation of monotyped arrays:

JD: what about providing values to types without default value

```
class Foo! {} new [Foo, Foo] // should be error, no way to know what Foo is (and
no default value) new [Foo, Foo](new Foo(), new Foo()) // this would be allowed
new [int, String](10) // Seems like a problem, you get 0 and 9 empty strings new
[int](10) // get ten 0's
```

So are we allowing arguments on this new syntax?

BE: We should move this issue to the Wiki proposals section and discuss this as part of a new proposal.

- Ticket #83: Adaptation of the Math object to the various numeric types:

LTH: Obvious solution is to make parameter types be `*` and dispatch on the actual type encountered.

GH: Or better yet use type `Numeric`

- Tangent on Numeric, Number and its subtypes. Lars will re-open Ticket #3: Issues around Numeric and subtypes of Number and we will discuss at a later meeting.

Resolved: int, uint, double, decimal should all delegate directly to Number.prototype. For example:

```
int.prototype.foo = 42; print (1.2).foo; // prints 42 because int.prototype.foo
=== double.prototype.foo
```

### Review Agenda items

- Decision on whether to adopt an environment model of type expressions, including type closures.
- Decision on when to resolve type names to type values, and whether to treat “strict mode” as partial evaluation.
- Decision on how much environment to share between compilation units / modules. Reconcile program units and multiple compilation units (if there is any disagreement between them).
- Clean up all wiki pages and source code that disagrees on these decisions, so we have a single story!
    - We should divide responsibility for this
- Need another wiki export

### Phone meetings between now and the next f2f?

Jeff and Lars will be miss next four calls, but calls will go on to discuss clean up work, etc.

### Spec writing process

- Spec writing process should start immediately. We reworked the TOC to mirror the AST.
- If proposal has smiley, should start moving it over to spec namespace. Proposals should include links to any related trac issues.
- Graydon will research Dokuwiki extensions that can merge in other content.

## Self type discussion

```
type T = {f:function(this:S,...) : R, x:int, y:int}  // S is what type, T, this,
*? y = {...}:T   x:{f:function(this:this, ...):R,x:int}   g = x.f // unsafe but
allowed   g({f = ..., x = ...})
```

A trickier case:

```
y = {...}:{f:function(...):this, x:int, y:int} x: {f:function(...):this, x:int}
= y; x.f = function(...):T{...}; x.g();
```

- self types will be updated to include the 'unsafe but allowed' issue, and we'll discuss this proposal further next month.
- Proposal should also include discussion of `this:this`, argument this, and return this.

## Packages and Namespaces

- GH: are packages necessary? Can't you just use `use default namespace` instead?
- JD: packages are not strictly necessary, but they are a useful shorthand.
- JD: packages are like namespace training wheels.
- GH: So the following is not possible?

```
namespace debug; package x.y {   debug var x; }
```

- JD: This is currently not possible, though I am not against making it legal.
- GH: Going the opposite way, what if we removed the namespace definition syntax (but left namespaces under the hood)? So `namespace debug` would not be allowed – you'd have to use a package and allow package ids to be qualifiers.
- LTH: I don't see the point in re-opening discussion on this topic when this was decided over a year ago after lengthy discussion.
- JD: I'm fine with allowing definitions before a package, and also allowing overriding of namespaces inside a package, so that the preceding example would be legal.
- GH: Given units proposal which of the following is preferable?

```
unit x {   com namespace sun;   use default namespace com::sun; }   // or
unit x {   package com.sun {     } }
```

- JD: I would never write the second, so I'm not opposed to this. However, there are a lot of developers with Java backgrounds who strongly prefer packages.
- GH: We could call packages syntactic sugar and leave it at that.
- BE: want to remove unnecessary restrictions (defn before pkg, no namespace qual override in pkg).
- CP: desugar packages to package-less core language.
- BE: desugaring packages means beefing up namespaces.