Minutes of the: Ecma TC39-TG1

held in: Sunnyvale, CA (Yahoo!)

on: 16th November 2006

Attendees

- Graydon Hoare, Mozilla Foundation
- Steven Johnson, Adobe Systems
- Doug Crockford, Yahoo!
- Cormac Flanagan, UC Santa Cruz
- Dave Herman, Northastern University
- Brendan Eich, Mozilla Foundation
- Chris Pine, Opera Software
- Dick Sweet, Adobe Systems
- Francis Cheng, Adobe Systems
- Dan Smith, Adobe Systems
- Michael O'Brian, mbedthis
- Jeff Dyer, Adobe Systems
- Michael Cowlishaw, IBM

Agenda

- Decimal presentation by Mike Cowlishaw
- General Spec issues
- Review latest proposal changes
- byte array class proposal
- Break into small groups to work on proposals

Notes

Decimal presentation

- history. 1950's: Binary was deemed more efficient, so became standard

- issues with Binary. binary cannot exactly represent most decimal fractions.

Rounding hides inaccuraces, but can lead to error accumulation. Leads to rounding errors. Binary cannot be used for commercial or human-centric applications.

- Why floating point decimal?

fixed point is awkward, error-prone, hard to maintain.

- Why decimal hardware?

Software is slow. e.g. Java 1.4 decimal addition is 1,700 cycles vs. 8 cycles for hardware. This is typical for many operations. Overall application test cases found to spend significant execution time processing decimals, ranging from 45% to over 90%.

- IEEE 754 current draft (’754r’). Decimal floating-point types and arithmetic were agreed upon in 2003. Suitable for commercial, financial, and mathematical applications.
- Integer-based floating point. Designed to match calculations done “on paper”. Compatible with IEEE 754, 854, and 754r. Mixed type arithmetic (12 * 9.99).
- Product plans. IBM will have dec floating-point units in hardware (Power6). IEEE 754r dec floating point is the strategic direction for SAP AG.
- Other standards. ISO C and C++ are jointly adding decimal floating point as first-class primitive types. Java 5 BigDecimal (arithmetic) in 2004 (MathContext for IEEE 754r decimal types). C# and .Net ECMA and ISO standards: arithmetic changed; allows use of 754r format COBOL 2002 has floating-point decimal (754r 128-bit type on COBOL 2008). XML Schema 1.1 draft has precision decimal.
- Benefits of new types. Better interchange, faster processing (esp. with hardware), no conversions (only one type of decimal), well-defined arithmetic and rounding rules; safer and easier-to-write applications; long-term: a single numeric type (no binaries).
- Reviewed ECMAScript decimal proposal and it looks fine. The true test will be implementing it. One suggestion is to add utility functions to provide access to bit patterns of decimal numbers. This would affect the byteArray proposal.

General spec issues

- What types of type checks do we need to do and when do we need to do them? Talked about in Oslo, and have gone through 8 different options. Dave and Cormac are still working on this. Clarification of runtime types and conversions. Dave thinks that it’s better to defer discussion about this until the next meeting. Issue is how many issues we plan to catch and which issues will be programmer’s responsibility to catch.
- Dave: Have we decided how the code and prose will intermingle in the spec.
- Brendan: Have we decided what to do about parts of the spec that needs exposition, such as regular expressions.
- Jeff: Can we carry use es4 to do that? If not, we’ll need to fall back to pseudo-code. Hoping all built-ins will be in es4.
- Brendan: There is also the issue of Unicode. SMLNJ may support Unicode now.
- Dave: Ed Smith has done a lot of work with defining the standard library in es4, but there are still some areas that aren’t covered.
- Jeff on subgroups: 1) writing built-ins in es4. This is a large group, so we may want to further subdivide this group. Should shoot for a rough schedule by the end of the next f2f meeting.

Weekly Phone Conference Change

The weekly phone conference will now occur on Tuesdays at 10:00 AM Pacific Time from now on (was previously on Wednesday).

Overview of current status

- Parser: Jeff has been working on the parser. Getting close to completing the expression grammar. The key is to get the AST stable. Dave: One of the dangers of a reference implementation is that everything is concrete, even things that you would prefer to leave unspecified. No agreement on how to deal with such situations. For example, Brendan brought up issue of toString(), which was not concretely specified in previous editions. Do we specify exactly how to implement toString() in es4? If we don't want to include that as part of the spec, how do we indicate that the reference implementation version of toString() is not part of the actual spec?

- Detour on Logical XOR: Dick proposes that we eliminate logical XOR. Jeff: it doesn't hurt to have it, but maybe we should take it out if we get criticized. **Resolved**: logical XOR to be removed.

- Detour on Type parameter syntax: proposal that x.<y> be changed to x of y. **Resolved**: Syntax will not change.

- Type checker: Cormac shows bare bones code for type checker. Graydon asks do we want the type checker to change the AST? We could have the type checker take an AST as input and output an AST with runtime checks inserted. Jeff thinks that its safer right now for the type checker to not change the AST. Cormac agrees.

- Parser: Graydon received several Java classes from Lars and is converting them to ML.

- Stylistic question: type names were in all caps. Graydon used this in mach.es to differentiate type names from constructors. Resolved ML convention for spec: type names in ALL CAPS (FOOBAR); constructors in CamelCase (FooBar); var/field in lowercase (fooBar).

- We want to write the built-ins using es4 instead of ML. Question is for things like sort(), do we want to use ML or just leave it open? i.e. how do we write Array.prototype.sort()? Jeff: should use es4. Cormac: sometimes we just have to use English. Jeff: Would rather have local descriptions of primitives rather than use Waldemar's approach.

Renaming traits

- Detour on changing name of 'traits'. Brendan says that its use here is not canonical. We'll need to change it. Proposed that it be typeDesc. Graydon doesn't like typeDesc. Jeff: let's describe what we're talking about.

1. [1,2,3]:[int]; 2.

```
class A {
       var i:int  // AS3 trait
}
```

Dave: we're talking about three different things 1. Type annotation 2. Meta-slots in the type descriptor in a class. *AS3 calls this traits 3. Runtime property. Proposed name is "layout". Property = {name, type, attribute, index}* this is a line in the layout, what should it be called?

Another example:

```
 class A {
     var x:int;
     function f(i:int):int {}
}

class B extends A {
    var y:String;
    override function f(i:int):int {}
}
```

Layout of B =

```
  {"x", "int", DE,    }       // slot 0
  {"f", "int->int", DE/DD,    }  // slot 1
  {"y", "int", DE,    }       // slot 2
```

DE = DontEnum DD = DontDelete

- mob: what's the conflict with use of the term 'trait'? It's used in general OOP to mean something like an interface or mix-in.

- T is type descriptor (classname, record type, function type, array type, null, undefined} i.e. any type that is not a union or any.

- Long ago, Graydon suggested adding a new typeconstraint attribute instead of having Layouts, but we discovered that this prevents implementation of vtables.

- Brendan: every value has a type tag.

- So, what has a layout? Classes, Records, Arrays and Functions.

- Dave: array structural type doesn't really have a layout because it's a subtype of the Array class.

- Brendan: I agree, layouts exist to allow vtables, and you wouldn't use a vtable for structural types.

- Brendan: I think we're agreed that the layout is for the nominal systems.

Spec logistics

- Graydon suggests using tags to denote code that will live in the spec.
- mob suggests that we have links in the final spec that takes you directly into the code.
- mob asks will all the source code from the ref implementation be part of the formal spec?
- Graydon's conceptualization is that only parts of the ML implementation will be included.
- mob would prefer to err on the side of including too much, meaning include the entire ref implementation.
- Brendan: we really need to make it clear where something is informative rather than normative.