Minutes of the: ***Ecma TC39-TG1***

held in: ***Phone Conference***

on: ***3rd February 2006***

Attendees

- Ed Smith, Adobe Systems
- Graydon Hoare, Mozilla Foundation
- Jeff Dyer, Adobe Systems

Agenda

- Discuss schedule, “drop dead dates”.
- Go through clarification issues.
- Go through foundational issues.
- Review recent changes on the wiki.

Notes

Schedule discussion

Brief discussion, mostly agree:

- March 1: no introduction of new proposals after this date.
- April 1: try to reach agreement on status of all introduced proposals.

Old-and-new code mixing discussion

Question: How do old code and new code interact?

We need a technique for defining compatibility in mixing code.

Ed: We should define the old language in terms of the new language constructs. That should be the goal; then, if a situation arises where that can’t be done, we should hone in on that, because that’s where the problems are.

Greydon: We might potentially end up writing two specs, since the old spec might need to be rewritten in terms of new spec.

Ed: A concrete example. In the old language, there is boxing of primitives. In the new language, there is not. So what happens when you pass a boxed object from old code to new code? At the boundary, what do I do with naked String value? What do I do with a String object? We should be able to write the E3 String object in E4 notation.

Is it well-defined if you throw an old object and new code tries to catch it using type matching? Do old objects have types? We need to write down whether it is so or not.

Guideline: define the old language entirely in terms of the new language.

Dispatch rules discussion

How does AS work?

- Dynamic dispatch on `this`.
- No specific way to call supertypes.
- No argument-type dispatch.
- No return-type dispatch.
- In general no name-overloading, only overriding.
- Does *not* do "get slot and call result" in ES3 terms.
    - Based on the XML call/get difference (E4X)
    - Also an important optimization: don't make a closure on each call.
- If no name-matching method on `this` type, walk prototype chain (ES3 style)

```
class C {
  function f()
}
var o:C = new C()
o.f()
```

- AS3 looks for a fixed function (traits) which matches open namespace with f.
- If present, call it with o as this parameter.
- No argument-type dispatch, no overloading. Waldemar's proposal had something like that for binary operators, but that's it.
- No operator overloading in AS3 at present.
- No ability to access supertype methods from outside the class.
- If we don't have a f on the type, then we degrade to same steps used in Edition 3. Walk prototype chain, invoke with same steps on original object.
- Prototype chain in Adobe implementation does a traits lookup and then does a prototype walk.
- The prototype chain is assumed to be vanilla Objects with no traits. But we think we should adjust that.

- If writable proto slots are supported, then you could end up with two legal prototype chains with a shadowing relationship, but that would only be known if the link was bidirectional... if the superclass could look down and see if it had a prototype delegator added to the prototype chain.
- There is some discussion of this on the drop_traits page.

Discussion on combinations of property attributes

- Consider when `DontDelete=false`:
    - Does `ReadOnly` have any effect? Not really: I can delete the property and replace it!
    - Does `DontEnum` have any effect? Not really, same reason.
        - Maybe redefine DontEnum in terms of a namespace?

- Edwin thinks all other combinations of the property attributes make sense.
    - But there's no "script way" to set some attributes.

- Enumerating an array with for...in on a methods attached to the prototype will show the methods.
- People want to add builtins, and should be able to, but there isn't a way to flip on the {DontEnum} bit.
- Edwin proposed that maybe {DontEnum} should be replaced with a namespace.

    - Brendan likes the simplicity of the bit, but he's open to a proposal from Ed on how it will work with namespaces.
- In AS3, declared properties are not enumerable.
    - This is same as the Edition 4 proposal, but E4 proposal had the "enumerable" attribute too.
    - AS3 doesn't currently do "enumerable" attribute.

A problem Adobe/Macromedia has been having

- ES3 style built-in object methods are late-bound, can be deleted from the prototype object, etc.
- Writing built-in objects in ES4 compact profile (early-bound, non-deletable methods) causes incompatibility.
    - Methods are declared with traits and are implemented with vtables like Java/C++. Whole design rests on this.
    - There has been the proposal to move methods to the prototype object.
    - We want to be able to write builtin objects using the language, and get transferrability in the right places.
    - It is proving difficult to reconcile E3 builtins with the traits model.
    - Prototype could be a subclass of the instance type.
- Saying that a prototype has "the same type as an instance" might not work?
    - What type constraint should the prototype slot `__proto__` have?
    - What type constraint should `Foo.prototype` have?
- We should think about this more, these may be reasons that Waldemar made the prototype be of type Object.

Opera wants to join

- Gave them a wiki login.
- At the moment, just wants to observe.

Type parameters

- A new proposal, fairly deep implications.
    - Broad purpose is to have typesafe "generic" collections.
- An important prerequisite: What is the type of a function from integers to integers?
    - Note that functions have subtype relations too, like classes.