ECMAScript Standards Meeting
November 27th, 2001
Netscape, Mountain View

Attendees

Waldemar Horwart, Netscape
Jeff Dyer, Mountain View Compiler Company
Eric Lippert, Microsoft
Herman Venter, Microsoft
Peter Torr, Microsoft

Agenda

Next Meeting date
General document review
Phases of Evaluation
Numeric Heirarchy

Next Meeting

We agreed that the next meeting should be at 10:00 am on January 29th, 2002, at Microsoft in Redmond. Jeff has a conflict with this and has suggested Thursday 31st instead. A week before the meeting, we will decide whether or not there is enough new content or other discussion topics to warrant the expense of the meeting.

General Document Review

- The tuples **GoBreak** and **GoContinue** have changed to allow a label to be either a string or a default (empty) value.
- **CompoundAttribute** is a new name for Attribute
- Created a new **VariableReference** tuple to store environment information with a variable
- New **Contexts** section to hold static information about the current location in a program
- New **Environments** section, still incomplete. **StaticFrame** stores compile-time information, whilst **DynamicFrame** stores run-time information
- **CombineAttributes** function combines attributes; the function cannot be called with a false first parameter, thus short-circuiting further attribute evaluation. Herman points out that combining attributes should be limited to a compile-time operation, as there is nothing interesting you can do with attribute combination at run time. Waldemar will consider making this change.
- References section updated so that the correct this reference is used. For example, if **ob** is an expando object, **foo** is an old-style function, then in the call **ob.foo(42)** the value of this should be **ob**. Herman expressed a desire for pre-conditions on these methods so that it was clear that, for instance, **writeQualifiedProperty** could never be called with a method reference
- For future meetings, Waldemar will provide "clean" copies of the document (without change bars) to ease reading

Phases of Evaluation

Currently, the draft specification has two phases -- verification and evaluation. There may be more phases required, for example the JScript .NET implementation adds a "partial evaluation" phase.

Once an attribute has been used, it cannot be shadowed. For example, in the following code, the usage of **A** on the third line binds to the declaration of **A** on the first line, and the attempt to create a shadowing definition on the fourth line will generate an error:

```
const A = true
{
    A const B = Something // Binds to the A above
    B const A = AnotherThing // Error; A is already bound
}
```

Waldemar would like to disallow forward-references to classes when used in an **extends** clause. For example, the following code would be illegal:

```
class Derived extends Base // error; base is a forward reference
{
    // implementation
}

class Base
{
    // implementation
}
```

The final specification will make such programs illegal, but Microsoft's implementation may extend the standard by allowing such forward references.

Numeric Hierarchy

We discussed the numeric hierarchy again. There are at least three possible ways to represent this:

```
     Object                   Object                   Object
       |                      / |  \                     |
     Number                  /  |   \                ????????
   /   |    \               /   |    \               /   |    \
int  double  decimal     int  double  decimal     int  double  decimal
```

The idea is that it would be nice for programmers to specify that their methods took "a number", without having to worry about (possibly lossy) conversions from integers to doubles, signed to unsigned, etc. The generic type would be more specific than **Object**, since passing a string or other non-numeric value would be an error, but would be less specific than **int** or **double**.

The basic problem is that the types in the CLR follow the second diagram (they all derive directly from **Object**), and it would be infeasible for Microsoft (or any other implementations on top of the CLR) to create a new numeric hierarchy that was not based on the platform's built-in types. One possibility that was discussed was to invent a new "pseudo-type" or compound type in place of the question marks in the third diagram. It would not be a real type in the type system, but would be a way for the user to specify "any numeric type", and the compiler would have both compile-time and run-time checks to ensure that this condition was met. Nevertheless, this would incur a runtime cost and would not interoperate well with other languages.

It was decided that, for this edition of the standard, we would remove support for Integer and Decimal types, and continue to have Number as the "generic" numeric type. The issue is taking up a lot of the committee's time, and we believe there are suitable workarounds for customers that need them (eg, the **System.Decimal** type in the CLR). We will also remove **long** from the standard (both signed and unsigned), which will enable all the built-in numeric types to be stored in a **double** without loss of precision. For the initial release, overflow checking will be turned on, but we may have a switch in the future to enable truncation. Other numeric types (such as long or decimal) may be added as part of the units work.