2003 March 26 TC39 TG1 E4 Meeting

Date	March 26, 2003 1:30 PM – 5:00 PM March 27, 2003 9:00 AM – 12:00 PM
Location	Netscape Communications Corp. Miami Vice Conference Room Building 21, 1st Floor 466 Ellis Street Mountain View, CA 94043
Convener	Rok Yu (Microsoft)
Editor	Waldemar Horwat (Netscape)
Participants	Jeff Dyer (Compiler Company) John Schneider (BEA/AgileDelta) Michael Shenfield (RIM) Rok Yu (Microsoft) Waldemar Horwat (Netscape)

Agenda
The agenda was adopted as written.

New Convener
The working group thanks Peter Torr for his outstanding service as convener of TG1 and welcomes Rok Yu into his new position.

Next Meetings
The next E4 meeting will be held in Redmond at Microsoft on May 8, 2003. Location specifics will be mailed out prior to the meeting.

The following E4 meeting will be held in Mountain View at Netscape on June 12, 2003.

Schedule Update
Waldemar has completed algorithms for all statements as well as function calls. Open issues exist in object instance construction. The remaining algorithm definitions to be completed:

- instance construction
- built-in object model
- type coercions

Waldemar has committed to having algorithms for core languages frozen by next meeting. Object model definitions will likely not be completed until June meeting.

The goal that the working group set last November was to be algorithm complete by the March meeting. We are approximately one month behind schedule.

Review Action Items
Waldemar to carve out the subset of E4 for next meeting
Waldemar has cut the following features from E4.

- Instance variable initializers are restricted to be compile-time constant expressions
- For-in statements may not have useless initializers as in "for(var foo=uselessinit in bar)"
- Support for named arguments and fancy enumerability choices is gone
- New kinds of arrays are gone, leaving only the E3 arrays.
- Hooks for packages are present, but packages themselves are not implemented and likely won't be by the time we finish.

These cuts were accepted by working group members.

John, Waldemar will discuss alternative presented by Rok to postpone E4 in favor of E4X with appropriate colleagues and will report back on where they stand.
The members do not have agreement on the relative priorities of E4 and E4X. Both John and Waldemar see E4 and E4X as independent specifications continuing in parallel using separate resources. The E4X specification extends Edition 3 notation and as such is decoupled from the E4 spec.

We have agreed to continue with the current plan to finalize both standards by September for December ratification.

Rok to confirm there no other reasons Microsoft's implementation has wrapper objects.
Rok has confirmed that the only reason Microsoft's implementation uses wrapper objects is to support expando members

Prioritization of Features
John suggested in email that we each bring list of important features and scenarios in order to help drive the future cuts to E4 if they are needed to meet the schedule

John states for BEA , primary theme is consumption of classes defined in other languages. Flexible class and package definition capability is not as important. Features that are important include:
- Interoperability features for working with strongly typed languages
- Ability to import types from e.g. Java packages (including .* option) – do not need include/exclude clause
- Declaring variables with types and casting variables to a specific type
- Using type information to select from overloaded methods
- InstanceOf in the context of working with external hierarchy
- Namespaces for disambiguating different variables/properties with the same name
- Calling property getters and setters defined in other languages
- Adding primitive types in language such as int

Lower priority to BEA but still interesting are:
- defining functions with types
- typed catches
- lightweight class definition mechanism - public/private visibility modification

Rok states Microsoft's priorities are very similar to BEA's in that consumption of objects and types defined in other languages is the primary scenario.

Waldemar states the set of important scenarios for Netscape include the ability to define classes securely.

Section 9.1.6: Add URI field to namespace type

Waldemar has made modification so that ns in ns::id doesn't have to be a compile time constant. Work needed to support namespace constructor will be completed when OM is done.

Intent of adding URI field was not recorded completely in last meeting.

E4X and E4 both have a concept of namespace. John would like to get these unified now. John wants to have one syntax for introducing namespaces – namely the dynamic one.

```
const ns = new Namespace(URI);
```

Problem with constructing namespace is that they are not compile time constants. Compile time constants are useful for use of namespaces in non E4X scenarios.

```
var ns = new Namespace()  is not idempotent and not a compile time constant;
```

Rok thinks language syntax for namespaces and having a new constructor is okay as the same model exists function declarations and creating new Function objects.

Agreement was reached that namespace objects are immutable and that adding a URI is probably a non-breaking change that can occur later. If the delta work to add it is small, Waldemar will add it now. We will continue to support the namespace keyword to create constant namespaces in addition to the namespace constructor for dynamic namespaces.

Action Items

- Waldemar will see if adding URI field can be added with minimal work and if so, add it.
- Waldemar will ensure built-in object model will have a namespace constructor.

9.1.8: Delete super field from class
Done

9.1.9: Rename parent field to super to avoid name clash with E4X parent
Done

10.5.6: Add look up of properties for primitive values
Done

Exit Criteria and Process

Everyone agrees that the working group must complete the following before signing off on the document standard:

- Review pass and verification of all algorithms
- Editorial pass of all prose in the document

Rok states that Microsoft's additional requirements before signing off on the standard are:

- Understand incompatibilities with Edition 3 of the standard
- Understand incompatibilities with JScript .NET
- For features not yet implemented in JScript .NET, ensure reasonable implementation is possible on Microsoft's .NET Runtime.

In order to achieve the first two points, Rok wishes to run a test pass against E4 reference implementation. Group agrees that this is a useful thing to do. Waldemar states that it is reasonable to assume that members will be able to run test suites against the reference implementation and that Netscape will be doing something similar.

John suggested that we make sure we don't leave any residual artifacts in the specification as we make cuts. Waldemar is fairly certain that he has removed all of these. Members have agreed to look for these during the process of review process.

Action Items

- Waldemar and John to determine what additional steps, if any, that respective companies will need to sign off on spec.
- Rok to get set of requirements ECMA requires before signing off on a specification.
- Waldemar will send out list of sections which are stable enough to review.
- Rok to revive spreadsheet to track sign off of sections and calculate throughput rate in needed to complete by September.

Unicode and I18N

This was a discussion of the proposals sent by Markus Scherer to the mailing list.

Edition 3 standard required Unicode 2.1 or higher. The Unicode proposal is to make changes required to make E4 standard require Unicode 3.0 or higher. The changes include

- Using the new Default_Ignorable_Code_Point to define ignorable characters (superset of Cf which we currently use)
- Understanding and processing surrogate pairs
- Supporting new syntax for characters outside the BMP range (i.e. represented by surrogate pairs)
- Adding an additional character to the set recognized as end of line
- Updating set of identifiers

On I18N proposes object model calls to simplify processing in different locales.

The working group is concerned about these issues as we want to ensure the standard we deliver will not be outright rejected by ISO because of Unicode or globalization issues. The bar for adding features at this point is extremely high. The primary deciding factor whether a particular feature will be added is if we expect it to be a showstopper for ISO.

There was general consensus that changing the basic data model for characters from 16 bit code points was not something we will not do in this edition because the slip it would cause and the consensus that the implementation cost on current platforms is too high.

We will postpone talking about this until Marcus is here.

Action Items

- Waldemar to talk with Marcus to come up with proposal for next meeting with list of features that are must have for ISO acceptance along with rationale.

Instance Construction

The open issue is how to call super constructors in a constructor. The design proposed by Waldemar is as follows.

Let *C* be a class and *B* its superclass. Any constructor *C*.*n* must call a constructor *B*.*m* or *C*.*m* before it accesses `this` or `super` or before it returns. The call can be either explicit or implicit; if *C*.*n* does not contain any calls to a constructor *B*.*m* or *C*.*m*, then a call to *B*.*B* with no arguments is automatically inserted as the first statement of *C*.*n*. The constructor *C*.*n* does not have to call another constructor if it exits by throwing an exception. *C*.*n* may not make a constructor call more than once.

A constructor *C*`.`*n* can call another constructor using one of the following statements:

`super(`*args*`)`	Calls *B*'s default constructor *B*`.`*B*.
`super.`*m*`(`*args*`)`	Calls *B*'s constructor *B*`.`*m*. *m* must be a *QualifiedIdentifier* that names one of *B*'s constructors.
`super.`*m*`.invoke(`*args*`)`	Calls *B*'s constructor *B*`.`*m*. *m* must be a *QualifiedIdentifier* that names one of *B*'s constructors.
`this(`*args*`)`	Calls *C*'s default constructor *C*`.`*C*.
`this.`*m*`(`*args*`)`	Calls *C*'s constructor *C*`.`*m*. *m* must be a *QualifiedIdentifier* that names one of *C*'s constructors.
`this.`*m*`.invoke(`*args*`)`	Calls *C*'s constructor *C*`.`*m*. *m* must be a *QualifiedIdentifier* that names one of *C*'s constructors

The invoke method is analagous to the Function.prototype.apply method though is special because it calls the super constructor. For Waldemar, invoke functionality is important for consistency between constructors and functions and to support composition of constructors.

Rok believes that the invoke functionality for calling super is not important to Microsoft's users and it was not immediately obvious if the feature could be reasonably implemented on the .NET Runtime. Rok proposed the feature be cut, or at least be unified with function.prototype.apply.

As a compromise, it was agreed that we would support a calling syntax for making an apply call.
    f(p1, p2, …arr)
where arr is used to supply the remainder of the arguments.

Rok was unaware of that the language supported named constructors and that there was a constructor attribute. Agreement was reached to cut these features.

An open issue remained as to what "C.n may not make a constructor call more than once" as it was not clear what could be supported on various runtimes. Microsoft's current implementation requires calls to super to occur as the first call of the constructor.

Action Items
Waldemar will add algorithms to support calling functions with an array to supply parameters. f(p1, p2, …arr).
Rok and Waldemar will research to see what restrictions must be placed on where calls to super may occur.

Review of E4 Semantics

Waldemar went through first 9 chapters and explained more subtle points and indicated stability of each section. This data will be captured in the updated status worksheet to be sent out later.

Action Items

- Everyone to review algorithms in stable sections of document and send email with comments.