Minutes for the: Meeting of Ecma TC39-TG1
to be held in: Redmond, WA, USA
on: 22 September 2005

TIME : **10:00 till 17:00 on 22 September 2005**

LOCATION : **Building 41/4585**
Microsoft Corporation
One Microsoft Way
Redmond, WA 98102

CONTACT : **Rok Yu (rokyu@microsoft.com)**
425-703-4297
206-369-0123

1 Attendees

Rok Yu (Microsoft)

Brendan Eich (Mozilla)

Jeff Dyer (Macromedia)

Edwin Smith (Macromedia)

Francis Cheng (Macromedia)

2 Convener Role

Bill Schulze has resigned as convener. We thank him for his excellent leadership.

Brendan Eich has graciously agreed to be convener for the next 6 month term.

3 Schedule for Ecma-262 Edition 4

Straw man for schedule is as follows

Feature agreement in 6 months

Draft in 1 year

11:00 AM Friday 1 hour morning conference call

Face to face meetings for rest of the year:

- Friday October 14th in Mountain View hosted by Brendan
- Friday November 18th in San Francisco by Macromedia
- Friday December 16th in Redmond by Microsoft

Consider evolving ideas JSEP: Secondary collecting process

4 Final review and approval of ISO/IEC (E4X) DIS-22537 and Disposition of Comments

Jeff will send out a draft with the following changes for final review end of week for review.

setName and setNamespace needs to call [[AddInscopeNamespaces]]

Group agrees that this should be done. Make sure to the attribute case correctly updates its parents' [[InscopeNamespaces]]. Back out the current change of throwing an exception.

Can [[InscopeNamespaces]] have prefix == undefined

Group agrees we can do this.

[[InscopeNamespaces]] where ns == undefined seems to only be used in ToXMLString. If we remove these from [[InscopeNamespaces]] then toXMLSTring needs to generate the set namespaceDeclarations (declared toXMLString step 9) by augmenting [[InscopeNamespaces]] by the namespace for the element and any namespaces from its attributes.

The one case that no longer works is where addNamespace is used to move declarations up an hierarchy when the namespace added doesn't have a prefix. Using a prefix will continue to work.

var x = <foo><p:bar xmlns:p="url"/><p:bar xmlns:p="url"/></foo>

x.addNamespace("url");

print(x)

old spec → <foo xmlns:genprefix1="url"/><genprefix1:bar/><genprefix1:bar/></foo>

under new proposal → <foo><p:bar xmlns:p="url"/><p:bar xmlns:p="url"/></foo>

x.addNamespace(new Namespace("myprefix", "url"))

print(x) → <foo xmlns:myprefix="url"/><myprefix:bar/><myprefix:bar/></foo>

ToXMLString Step 10.a

[[*AncestorNamespaces*]] → *AncestorNamespaces*

9.1.1.11 [[Insert]] and 9.1.1.12 [[Replace]]

Text needs to be reverted back. The original comment re: propagating namespaces up ancestor chain is valid and prefix reassignment are fine.

5 Feature Proposals for Edition 4

Design space

	Large	**Small**
Compile time checking	Flex	AS2/JScript.NET
Runtime time checking	AJAX	Browser (Edition 3)

Compile time versus runtime checking

class A { var x; }

class B extends A { var y; }

var a : A = new A, b : B = new B;

a = b; // allowed?

a.y allowed ?

var o : Object = b;

o.toString

o.x

o.y

Type annotations: Yes.

Adaptation: Annotations allow implicitly convert in cases where there is E3 stuff so libs can change without much disruption to clients. Consider allowing internal methods.

Consider variants: Generic non-null type, non-converting type (B)

print(s : String) // pass in null ref value

print(s : String()) // converts using ToString?

Packages: Yes. A way to group items together.

Namespaces: E4X namespaces at minimum. Consider unification to program space in E4 Look at versioning as one use case, but don't block on it.

Classes

- Min bar to drive is to be able to implement E3/E4X built in object model using the class system.
- Subclassing via extends
- Interfaces discuss (P2)
- Overloading (future proof)

If you have class A, class B extends A, your hierarchy ends up looking something like:

Object

Class

Ac

Bc

A

B

Object.prototype = { }

var A : Ac, B : Bc;

A.prototype → { [[prototype]] = Object.prototype }

B.prototype → { [[prototype]] = A.prototype }

function C gives adds into the hierarchy like:

Object

Function

Fc

var C : Fc;

C.prototype → { [[prototype]] = Function.prototoype }

Optional args/rest args

Need to support

Nested classes: Deferred

Method closures: Extracting methods is in. Bound to instance.

Generalizing E4X operators to objects: Deferred (B)

..

filtering predicates

Wild card identifiers

Include directive: Deferred

Generators: B

Metadata Attribute Syntax: Comma separated list in square brackets of expressions or something

Attributes, Attribute Blocks: B

Boolean attributes – conditional compilation support: B

Enums: B

Block Scoping: B

Units: B

Operator overloading: B

Getter/setter: Class level, global?

Native Types: A

Decimal: A

Unboxed primitives: B

Make built in primitives Sealed/Final: A

Typed Arrays: B

Regexp extensions: Bug fixes, Unicode helpers – A, everything else B

XML Schemas: B

.hashcode or something to give objects visible identity: A

Unicode upgrades: A

I18N upgrades: B

Bug fix to allow regexp constructor to copy regexp instance: A (new /re/g)

Strict mode (Macromedia static type semantics): A – a chapter or some normal form on how this behavior changes

E4 Notation

Operational semantic language: Consider base core ECMAScript subset used to represent the next.