ECMA/TC39/97/24

Notes from the 3/14 working group meeting

In attendance

Guy from Sun

Brendan from Netscape

Clayton from Netscape

Shon from Microsoft

Scott from Microsoft

Randy from Borland

Agenda:

- Discussion of changes from last time
- Discuss Shon's Date proposal
- Versioning
- Guy asking questions about proper object model behavior

Next meetings, etc.

3/18 TC39 technical meeting at Microsoft in Cupertino

3/19 TC39 business meeting

3/24 Teleconference 11 am - 1 pm, organized by Scott

4/1 working group meeting at JavaSoft

Discussion of changes from last time and general comments

In several places the document uses zero with saying whether positive zero, negative zero or both are intended. It would be useful to do a scan through the doc looking specifically for these cases.

This expression:

```
x++.toString()
```

is allowed in the current grammar. It means this:

```
(x++).toString()
```

We are going to tweak the grammar to disallow this case. (NS does not currently support this; MS does; no one thinks it is important to make this work.)

4.7.3 Regarding the issue in the text (resolution and numeric literal

>-- how many digits matter) -- 20 are guarentted (rather than 19m as stated in the doc).

Formatting numbers with exponents -- we will always provide a sign for the exponent. We will always use a lower case e. E.g., 1e-30

We need to scan through the functions and note the attributes on each.

E.g., in 12.5.5.1 the doc says the length property is "unchanging". We should use attributes rather than text for describing this

>functionality.

Infinity and NaN literals (A.4). We decided to have these, with minor modifications. (This was Guy's suggestion.)

A.6. We decided to add a fromCharCode function. E.g.,:

```
String.fromCharCode() == ""
String.fromCharCode(65) == "A"
String.fromCharCode(65, 66, 67) == "ABC"
```

and

```
String.charCodeAt(...)
```

Discussion of Shon's Date proposal

Looks like we will adopt this, modulo some changes:

1. Year 2000 issues. Need to add getFullYear, setFullYear.

2. Date range. There was some discussion about the range of dates that are allowed. There was no resolution of this.

>3. Agreed to whack some of the extra parameters -- we'll only keep the ones needed to avoid confusing code, as in this example:

```
var x = new Date("2/28/1997")
x.setDate(30)
x.setMonth(11)
alert(x)          // It's not 12/30!
```

4. Agreed to do getGMTYear, getGMTDate, etc.

5. Should add setMilliseconds, getMilliseconds

6. Agreed to have a "not a date" value.

Shon will revise the proposal and distribute it on Tuesday.

Versioning discussion

Agreed that:

* Most of this versioning issue is beyond the scope of the document. But it does have an indirect effect on the standard -- whether or not we're inventing a new language or codifying existing practice.

* The default is the highest version of ECMAScript that is available.

* We should have runtime support that enables querying the engine to find out what version is supported.

Guy asking questions about proper object model behavior

* Does Object(obj) always/sometimes/never wrap obj?

* Is object.prototype read-only or not.

It is read-write

* What does toString on the prototype object return?

[object] String

[object] Date

[object] String

[object] String

* What does valueOf on the prototype object return?

```
If the object was wrapped
        return the original object
else
        return the object itself
```

* Object instances do not have any built-in member?

No

* What happens when you call the Function constructor with a function?

If a compilation error occurs in a use of Function, a

>non-function (typeof for it is not "Function") is returned.

* We should allow this:

```
x = Function("a, b, c", "d", "return (a+b+c+d);")
```

This is processed as if this were written:

```
function Anon(a, b, c, d)
{
        return (a+b+c+d);
}
```

* Is the length property of the Function constructor == 1?

Yes. In general we return the minimum arity.

* What happens if you call the Function constructor with no arguments

This is legal and equivalent to passing the empty string as the lone arg

* If I create a function instance. foo.prototype = <some object>. Is there any magic to set the Constructor property automatically?

No. The backlink is not fixed up automatically. You can do this manually.

* foo.arguments is transient. When foo is called, it remembers the state of foo.argumenents (including whether it exists) and then whacks it. On exit, it restores the original state.

* We are dropping function.caller from the spec. This is in the NS implementation but function is clearly the wrong place for this.

* Does valueOf for arrays return the array object?

valueOf() on an array does a join with comma.

default value for an array