>From: Wiltamuth S.
>Sent: mercredi, 19. mars 1997 09:58
>To: e-tc39
>Subject: Notes from the March 18 TC39 technical meeting
>
>
>
>Progress
>------------------------------------
>General agreement from the working group participants that we have made
>excellent progress over the last several months. There are some
>technical issues left to discuss, and there is editing to do. But from
>a technical perspective there appear to be no barriers to completing
>the standard on-schedule for the June ECMA g.a.
>
>
>Next meetings
>------------------------------------
>We discussed whether another full TC39 meeting would be required for
>ECMAScript v1. We agreed that a v1 meeting may be required, e.g., if
>the name issue is not resolved tomorrow. We also agreed that since
>multiple vendors are continuing to innovate in this space, it would
>behoove us to schedule a full TC39 meeting to kick off ECMAScript v2
>work. A scheduled TC39 meeting for v2 would be a good target for
>interested vendors to contribute design proposals. We will pick a date
>for this TC39 meeting tomorrow.
>
>The next working group meetings ahve already been scheduled:
>* 3/24 Teleconference 11 am - 1 pm, organized by Scott
>* 4/1 working group meeting at JavaSoft
>
>
>Next steps for the standard
>------------------------------------
>Discussion of the steps that the ECMAScript v1 standard will take over
>the next six months. (I didn't take notes on this discussion.)

>

>

>Comments on the document -- need for standard standard language

>-----------------------------------

>We need to add the standard "boilerplate" standard language, including:

>1. Scope

>2. Conformance clause

>3. Normative references. E.g., we need to reference the Unicode

>standard and the floating point number standard.

>4. (Optional) Terms and their definitions

>5. Conventions

>

>

>Review of the "Dates" proposal from Shon

>-----------------------------------

>Changes from last time

>* Changed the set* members. We discussed set

>* Added setFullYear.

>* The time value for a date object can be NaN.

>

>

>Comments/changes:

>* "GMT" is the wrong term. We should be using "UTC" instead.

>* Handling daylight savings time when handling setHours and setDate.

>There are probems with crossing the daylight savings time boundary.

>Both MS and NS set the time back to a non-fictional time. E.g., if the

>daylight savings time rule is "leap ahead one hour at 2 am" then 2:30

>am

>doesn't exist. If one tries to set the time/date to such a fictional

>time, then the time is set back to 1:30 am   This logic is not

>reflected

>in the current proposal. Shon will revise the proposal to incorporate

>this logic.

>

>

>General comments on the document

>-----------------------------------

>Grammar notation -- where did we get the grammar notation? Some

>comments on "non-standard" notations, e.g., the use of "but not" in

>productions.
>
>Minimum maximums. Do we need to specify more "minimum maximums"?
>E.g.,
>the length of identifiers is currently unspecified. Should we tighten
>this up by providing a specific minimu maximum. E.g., implementations
>must support identifiers with t least 256 characters. In general we
>have avoided doing this. By not specifying these minimum maximums, we
>require implementors to be limited only by system resources. We
>discussed this and decided not to change anything.
>
>A.4. My summary mail from the last meeting was a bit misleading. NaN
>and Infinity are not literals in the same way that true and false are.
>E.g., this program is legal:

```
>          var NaN = 123
>          alert(NaN)        // 123
```

>and this is not:

```
>          var true = 123
```

>So, the way we will describe this is that NaN and Infinity are members
>of the global object -- they are not literals.
>
>toNumber and Infinity. toNumber("Infinity") will give the infinity
>value
>
>toNumber and NaN. You could say the same thin about NaN --
>toNumber("NaN") gives a NaN. But this is true even if NaN is just
>treated like a regular string.
>
>Versioning. We need a compile-time check and a runtime check.
>
>
>Review of Buffer/Blob/BinaryObject proposal from Nombas
>-----------------------------------
>We discussed this proposal and agreed:
>* We will not pursue this for v1.
>* Something like this is interesting for v2.
>* Nombas will revise the proposal, based on the feedback from today,
>and
>present it at a future meeting.

```
>
>
>Versioning
>------------------------------------
>There are three issues:
>* Runtime check
>* Compile time check
>* External (e.g., browser) verison management.
>
>Versioning and a compilation-time check
>------------------------------------
>A proposal:
>* Make it like a C-preprocessor but very simple and limited, and with
>different syntax so as not to
>interfere with the use of a C pre-processor.
>* No macro substitution.
>* Allow switching on the version and on keywords.  Need to define the
>exact list.
>* Desired set of operators.  The expression language would be limited,
>The productions in 8.3.5 through 8.10 make sense.  We could do a
>smaller
>set, and this would not be a serious restriction.  What a smaller set
>would look like:
>          Predefined identifiers
>                    Version (as an integer, correlated with the ECMA
>standard version)
>                    Keywords (all keywords)
>          Ability for a user to define identifiers
>          Ability to do conditional compilation (if/else)
>          Operators
>                    These operators:
>                              <, >, <=, >=, ==, !=
>                              !
>                              ||, &&
>                              Grouping with ( . . .)
>                    No arithmetic operators
>                    No bitwise operators
>* What types are allowed in the expressions?  Boolean and Number.  No
>Strings.
```

```
>* No function access.
>* Should we make it look more like ECMAScript or more like the C
>preprocessor?  Like ECMAScript. LineTerminators are not significant.
>* Letting old engines work without modification.  There should be a way
>to hide this stuff from old engines which do not support this
>mechanism.
>This hiding mechanism should be orthogonal to the conditional
>compilation mechanism so that the hiding mechanism can eventually go
>away.
>* Language sketch:
>       @var <id> = <constant>
>       //@var <id> = <constant>
>       @if (exp)
>       //@if (exp)
>       @else
>       //@else
>       @elif
>       //@elif
>       /*@<ws>
>       @*/
>       @endif
>       //@endif
>
><constant> is true, false or a numeric constant (hex, octal or
>decimal).
>Undefined symbols are treated as false.
>
>* If an @if statement fails, are conditional compilation expressions
>within the @if block evaluated.
>       @if (test)
>               @var x = true
>       @endif
>       @if (x)         // Is x defined?  No, so it is false.
>               ...
>       @else
>               ...
>       @endif
>
>* Example 1:
```

```
>          /*@
>          @var debug = true
>
>          @if (debug)
>                     // Debug code here.
>          @else @if (profile)
>                     // Profiling code here.
>          @endif
>          @endif
>          @*/
>
>* Example 2 (same as Example #1 but using elif and without hiding):
>          @var debug = true
>
>          @if (debug)
>                     // Debug code here.
>          @elif (profile)
>                     // Profiling code here.
>          @endif
>
>* Example 3
>          /*@
>          @if (version >= 3)
>                     // Use v3 features
>          @elif (version >= 2)
>                     // Use v2 features
>          @else @*/
>                     // Use v1 features.  Note that this code is unhidden.
>          //@endif
>
>* Example 4
>          alert("This message always appears.")
>
>          /*@
>           alert("This message appears if conditional compilation is
>supported.")
>          @*/
>
>          /*@
```

```
>	@if (true)
>		// Do nothing
>	@else @*/
>		alert("This message appears if conditional compilation
>is not supported.")
>	//@endif
>
>* Example 5
>	//@if (noisy) alert('here'); @endif
>
>
```

>Versioning and a run-time check
>------------------------------------
>Agreed to add a member to the global object named
>	ECMAScriptVersion
>It returns an integer value. For ECMAScript 1.0, it returns 100.
>
>As with Infinity and NaN, this member is r/w. Changing the value isn't
>a very smart thing to do, but it is allowed.
>
>Things to add to the extensions list
>------------------------------------
>Randy proposed some version 2 features. Randy will write up proposals
>for these later. Here is a list of the new members:
>
>String extensions:
>* isAlpha, isLower, isUpper
>* leftTrim, rightTrim
>* String.replicate, String.space
>* right, left
>* stuff
>* toProperCase
>
>Math questions/issues
>* Math.int -- chop toward 0 rounding.
>* dtor -- degrees to radians
>* rtod -- radians to degrees
>
>Array extensions:

>* scan(expr, [start, [count]]), returns the element number for the
>match.
>* fill(expr, [start, [count]]), fills the designated array positions
>with expr
>* multi-dimensional arrays
>* other members related to multi-dimensional arrays
>