ECMA

ECMA/TC39/97/5

Standardizing—Information—and—Communication—Systems

From: Wiltamuth S.

Sent: Thursday, 16. January 1997 03:06

To: e-tc39

Subject: 1/15 working group meeting notes

Suggestions and comments

- The document needs to be clear about what is a building block and what is exposed to users.
- The document should note deviations from commonly accepted current practice. In particular, when we decide something that is in conflict with all of the implementations, this should be called out in either the document or in an addendum.
- The word "type" is not used consistently in the document submitted by S. Resolution: cleaner terminology will be added when this material is merged into the working draft.
- The standard should describe the fact that the language has automatic memory management.

Items for the extension list

- pre-processing.
- do…while… construct

Resolved:

- We will not do domain-specific work, eg objects for the web browser or web server. It may be interesting to do this in the future but it is beyond our scope today.
- We will get to a complete draft before looking at any extensions.
- The production PrimitiveExpr: (Expr) should perform "get value". The result is not an lvalue. Examples:

```
(a.x)(2)          // Legal.  The this value used for the call is null.
(a.x) = 10        // Illegal
(a.x)++           // Illegal since (a.x) is a value, not an lvalue
```

- The ? operator and the comma operator. Should these operators carry "this" through? Resolution - the result of the ? operator is a value, so the result of an ? operator cannot be used as a left hand side.

```
(a ? b.c : d.e) (1,2)   // Legal.  The this value used for the call is null.
(a ? b.c : d.e) = 2     // Illegal.
```

- When errors are raised. The spec specifies the last possible moment that an error can occur. Implementations may err earlier. This will introduce some differences, due to side effects. If we do try…except in the future then trappable errors must be specified precisely.
- Unicode identifiers. We will not allow unicode identifiers. This is a hornets nest because there are so many characters to look at, e.g., wide digits and symbols. Also, there is a workaround - use square bracket syntax with string literals.
- Octal and hex escape sequences. We will support these.
- Unicode escape sequences and end of source. We should make sure that a user-defined unicode sequence not be misinterpreted as specifying end of source. E.g., \u0000.
- We will allow embedded nulls in strings.
- Add form feed and vertical tab as white space.
- Randy to fix the description of auto-semicolon insertion to handle the eturn and carriage return case. (This is currently listed as an issue in the spec.) A few examples:

x = i

++j

return

a + b

- Keywords and reserved words. Agreed to add to the keyword list: delete. Agreed to remove the from the keyword list: arguments, class, extends, finally, implicit, switch, case, default, catch, try. Agreed to have these (and only these) future reserved words: catch, throw, try, finally, switch, case, default, class, extends, implicit, arguments, do, while; everything else gets whacked.
- Octal syntax. For numeric literals, if there is a leading 0 not followed by a dot, an x or an e. E.g., 001.5 generates two tokens - 001 and .5. This would be an error.
- Octal and ToNumber. Leading 0's in a string are not interpreted as Octal.
- In 5.1, delete the line that says "Errors are never generated as a result of calling ToPrimitive".
- In 5.3.1, this hex issue is settled - hex numbers are allowed in strings being coerced to number with ToNumber.
- The attribute set for user-defined functions and variables explicitly eclared in global code and variables defined implicitly in code is the empty set.
- The attribute set for built-in objects is {DontEnum }. (Note that we don't mean the members of built-in objects; we mean the objects themselves.)
- A function defined with eval does not show up on the global object.
- 7.2.2: remove Step 3.
- 7.2.2: the resolution of the issue listed there is that the type of i++ is always a number.
- 7.3.4: the resolution of the issue is Result(5).
- 7.3.5: the issue just goes away - what is in the spec is fine.
- 7.4 and 7.5.1: will incorporate material from the Netscape doc and then review it.
- Hints for ToPrimitive. The hint should be optional. When the hint is omitted and the value being coerced is an object, the object can do as it pleases.
- 7.5.2: Remove steps 5 and 6.
- 7.6.1: Left shift for a negative number - ToInt32, as spec'd. No <<<.
- 7.7: Remove Boolean@ from table.
- 7.9: ToInt32. This is the same issue that we settled before.
- 7.11: We settled this earlier - the issue can just be removed.
- 7.12.1: We should do this as spec'd. E.g., this is an error: (a = b) = c
- 8.4.2: The for…in for the "var in" case needs to include multiple vars, not just one.
- 8.4.3: Should we restrict Expression1? Not grammatically - this would be hard to do. There are restrictions on what can be used here; they are described algorithmically.
- 8.5.3: For functions with multiple returns, there is no requirement that the return statements have the same number of arguments and types.

Open issues

- Non-1970 dates. We will discuss this as an issue. We will not discuss a date type.
- Versioning.

- Prototype property attributes regarding get and put. Looking at 4.5.2 in v .3. If the prototype for an object has a ReadOnly or ErrorOnWrite member, and code attempts to do a put on the object with the same name as the prototype object member, then what is the result? That is, are the attributes of the prototype member respected? Brendan to define an example that justifies changing what is in the spec.
- Order of evaluation for assignment. Discussed this; Randy needs to think about this more.

```
// Example 0
x = y = z = 1

// Example 1
var x = 1
var o = new Object()
with (o)
x = o.x = 2
print(x)  // Is x 1 or 2?  It's 2 if we do left-to-right. This is what is in the spec today.

// Example 2
o[i] = i++
```

- Max string length. Proposal - implementations must support strings with at least 32000 characters. There was discussion about whether to have a minimum or not.
- Need to more rigorously define "anonymous code".
- Will the style of the document be acceptable to ISO?