ECMAScript meeting 12-13th July 1999

Present:

Mike C
Dave
Clayton
Waldemar
Andrew
Herman
Dario
Richard
Bill
Mike M

Next meetings

September 23/24 Meeting

Location is yet to be determined. Somewhere in Italy. Fly to Milan. We hope to sign off the E3 spec at this meeting.

October 15th??

Please check if this ok and get back to Mike.

November 15-16th?

Please check if this ok and get back to Mike.

There won't be a meeting in December, so the next one would be in January.

Meeting Notes

7.1 Unicode Format control characters

For instance, left to right markers for arabic for separating letters and numbers in an identifier

Richard hasn't done anything and asks for a reminder as to what is needed. Waldemar summarizes the options open to us. The spec currently says you should ignore them. This is justified by the idea that if the compiler tried to interpret these codes, it would mess up strings.

Some discussion as to what happens to characters after a backslash. Richard thinks the Unicode format control characters hardly ever appear in program code, so while go to a lot of effort to deal with them. In this view Unicode

escape sequence for the character would be an error unless it appears within a string.

If an escape sequence includes such a character in the middle, it should be ignored. If it appears within an identifier it would be too hard to strip out control codes later, so this might be best treated as an error. Inside a string literal it is just added to the string.

XML doesn't allow these codes within tag names. The idea is that the characters are normalized, so removing the need for the formatting codes.

Dario proposes we simply ignore Unicode formatting codes throughout the code including string literals. You would then use a Unicode escape sequence if you really need to include the characters within a string.

Action Richard to formulate changed needed to the spec.

8.6.2 Internal Properties and Methods

It is not entirely clear what we mean by exposing internal properties of the language. Bill has revised the first paragraph in section 8.6.2 to make the intention clearer. We agree to strike the first sentence on the grounds it doesn't add anything.

15.1.2.8 encode URI() and 15.1.2.9 decodeURI()

Bill restructured Dario's text and changed the section numbers to 15.1.3.*. We discuss what happens when you expand the encoding to a 32 bit value. Waldemar proposes that if it can be expressed as a surrogate pair you use that otherwise you get an error. Discussion on 4 and 5th line of the table (present in earlier drafts, but recently removed) and their corresponding values.

Action: Dario to summarize changes for Bill.

Bill asks about the handling of % and # in relation to the unescaped set, see 15.1.3.3. The notes need to be revised to be consistent with the algoritm (striking the % in the notes).

Dario describes an issue raised by Martin Durst relating to URIs encoded using an earlier algorithm. Bill proposes we add a check on the upper bounds of the character value between lines 29 and 30 of the algorithm.

Mike resolves 15.1.3 is moved to content agreed.

15.4.4.11 Array.prototype.sort

There appears to be a control code between the m and the p in (comparefn) in the heading. This introduces a space in some printouts ...

Bill has to include the new text.

15.5.4.14 String.prototype.split

Some further control code problems. In one printout the bottom two lines of a paragraph were overprinted. Bill passed around the changes to allow us to check the numbering. Moved to content agreed.

15.5.4.9 String.prototype.localeCompare

control code/space between m and p in heading. Waldemar suggests we make it slightly clearer that the result is a number. Bill notes the suggestion. Richard suggests the note be extended to say the comparision is a tertiary match as defined by Unicode. We discuss making it more explicit that for strings which are identical the function returns zero. We spend time worrying whether the algorithm will provide a total ordering for a given set of strings. What risk is there of implementations differing? Perhaps "an implementation dependent total ordering".

Action: Waldemar to propose some wording.

Herman asks whether the term "underlying operating system" appears elsewhere in the spec?

Subject to Waldemar's text, we move this section to content agreed.

Regular Expressions

Changes to 15.10.2.5. Proposal that we strike the test for infinity. If you have an empty match, the suggestion is you only use the minimum number of the range. Bill notes the change.

Change to 15.10.2.8 regarding backtracking. We should match the existing Perl behavior as described by Illya as opposed to Larry Wall. Waldemar's note has some changes to the informative text which clarifies this.

We agree to move this to content agreed.

7.5.3 Future Reserved words

Mike says this is no ok and doesn't need further discussion.

7.9.2 Examples of Automatic Semicolon Insertion

Moved to content agreed.

7.9.1 Rules of Automatic Semicolon Insertion

Out of date in the draft spec.

Number Formatting

Waldemar summarises his proposals for Number.prototype.toFixed, toExponential and toGeneral. Herman asks whether we need to allow as many as 20 digits precision for fraction digits. What should we do if the fractio

digit precision passed is less than zero? Implementation defined would preclude later use of negative precisions for rounding numbers say to the nearest 100.

If the number can't be representing without dropping precision should an exception be generated? An exception would make it practical to wrap the toFixed call and switch to toGeneral if an exception was thrown.

Apart from step 2 and 9 the only other issue is what happens when more than one argument has been passed. We agree that further arguments be reserved for future revisions to the standard, using similar text to that used for toLocaleString.

We then turned to toExponential. Mike proposes we change the interpretation of precision to include the digit before the decimal point. A precision of zero yielding one digit seems strange says Richard, motivating including the digit before the decimal point as then a precision of 1 yields 1 digit. Another way would be to call the parameter "fractionDigits".

We agree to treat toExponential consistently with toFixed, e.g. for negative precision and for more than one argument.

Brief discussion on the names for the functions, e.g. toExponential versus toScientific. Is there anything better than toGeneral?

Proposal to make step 17 in toGeneral use p rather than 20. We return to a discussion of the name of the parameter precision for toExponential and toGeneral. We like the idea of renaming toGeneral with toPrecision. We decide we can live with fractionDigits for toExponential.

We discuss the idea of engineering numbers where the exponent is limited to multiples of 3 (thousands, millions, billions, etc). Should this be another function e.g. toEngineering with toExponential renamed to toScientific, or should we just add a parameter to toExponential? Either way we won't do engineering number formatting in this spec (E3).

We choose to continue to use precision with toPrecision but to include the digit before the decimal point. We decided to generate the same exception for all functions when fractionDigits is less than zero and precision is less than one. We decide to change the Array bounds error to be called range error and to reuse this for when the fraction digits is less than zero.

We go back to discussing step 9 for toFixed. Clayton proposes we don't generate any exceptions for numbers based upon their value e.g. numbers > 10^21 including infinity. We come down in favor of retaining the use of ToString(x) for very large numbers.

With the exception of the paragraph on more than argument, we agree to move the number formatting to content agreed.

Review of the E3 Document

Mike explains his plan to review the document, section by section. He hopes to finish by tomorrow lunchtime.

Section 3 - normative references. We think we can strike the reference to ISO/IEC 646 (ASCII).

Section 4 is out of date. We need to add regular expressions and errors to the list in 4.2 3rd para.

Section 5. We consider the consistent use of roman type for Unicode e.g. in section 7.6. Herman proposes a change to the last paragraph in section 5.2. Waldemar picks up on a few points earlier in section 5.2.

Section 6. Add regular expressions to the list in the 2nd paragraph. Richard proposes we strike the 3rd paragraph. So resolved.

Section 7. Delete last sentences in 1st and 2nd paragraphs in section 7.1 (we agreed to ignore Unicode formatting codes uniformly). We choose to tweak the 2nd paragraph in section 7. Bill proposes we strike the comment after USP in the whitespace production in 7.2.

Richard will draft some text for 5.1.5 to explain that characters such as * and - in grammar rules are in the ASCII range (see 7.4 for examples of such rules). In 7.6 comments in square brackets will be moved out from the grammar rules into separate comments.

In 7.7 === is missing, as is !== from the list of punctuators.

In section 8, first para, the use of bold is perhaps inconsistent with other parts of the spec. In 8.6.2 [[Closure]] is to be renamed to [[Scope]] to avoid confusion with closures.

In section 8.6.2.3 Waldemar is concerned with the description of [[CanPut]] for objects that don't have CanPut methods since this leads to an inconsistency with [[Put]]. Bill will send Herman his proposal regarding [[CanPut]] for offline review.

9.3.1 delete comment on USP. In 9.8.1 we note that some printouts don't show the greater than or equals symbol. This appears to be a font problem.

10.1.1, 2nd bullet point. Delete text from "which is referred" since we don't understand this. In general, people don't understand section 10. Luckily Waldemar volunteers to propose a rewrite to make it more obvious. Note that this section took a lot of time and effort for the first spec for ECMAScript. Herman proposes we simply drop 10.2.4.

Proposal to split productions in 11.1.4 to avoid the use of the dot notation: ArrayLiteral: ElementList, Elision_opt].

Bill needs the proposed changes to the text by August 1st. Please post them to the reflector as well as to Bill, so that he can prioritize his work according to how contentious things are. We may then decide to hold a teleconference in August.

break for dinner

Tuesday, July 13th, 1999

1 Scope

Very short. Perhaps better to say what is out of scope. Mike asks for volunteers to propose new wording, but none of us says yes.

2 Conformance

Herman suggests removing the last paragraph. Dave responds that the paragraph is valuable as it allows vendors to work on extensions while continuing to be a conformant implementation. Waldemar makes a case that the text isn't strong enough and wants to clarify the case of errors:

> "A conforming implementation is allowed to replace a behavior currently defined to throw an error by implementation defined behavior"

The issue arises since extensions make cause something that was error free to be interpreted as in error. Mike proposes we add "support program syntax and semantics" rather than "support program syntax". Andrew notes that we don't provide any standard conformance tests, so the conformance info is not really critical. It does however, warn people that the standard is in practice behind implementations.

We end up deciding that we shouldn't provide further encouragement for divergence, and leave the text as is. We may want to go through all the error cases to mark which ones must throw errors in conforming implementations. This was not an issue in the past, since errors stopped the program.

Richard proposes a change to the first paragraph. Bill suggests its ok as it is. Herman suggests "semantics and syntax" instead of "syntax".

Treatment of preserved arguments on functions

See 15.4.4.3 Array.prototype.toLocaleString. Should we generally adopt the note to protect future use of additional arguments.

Mike argues that we can't do this for old functions since it may break existing programs. The standard said that additional arguments were ignored, leave people free to provide them. If a function is extended to use an extra argument, old code may thus break.

Clayton proposes E3 includes a recommendation that well-formed programs only provide the specified number of arguments. Herman argues that extensions involving adding arguments to existing functions is more likely to cause problems than new funtions.

For user functions it is defined that extra arguments are ignored.

Bill proposes we add a new paragraph after the third paragraph in section 15. he proposes we state that if functions are called with additional parameters the behavior is undefined. We should also add a non-normative note recommending that extensions define new function names rather than define additional arguments to existing functions.

Mike proposes a variation on Bill's text: if functions are called with additional arguments, the behavior is undefined and is permitted to throw an error.

Delete operator

The naming of the reference error. Jeff proposed to rename this to property error, but we backed off on this, because we were concerned that this error might show up with the delete operator. Further study shows that delete doesn't return any errors. The error was originally called assigment error, but could also be thrown when a variable was deleted. We decide to stay with "assignment" error.

Document Review

In section 11, Bill notes that Jeff didn't supply his update on errors, so Bill used his meeting notes instead. References to non-terminals are supposed to be italicized.

Herman proposes changes to the algorithms in 12.1. He will pass the details to Bill. Various places where "NoIn" variants are needed for expressions in the productions. In 12.14 we are only allowed to have one catch, so we need to drop CatchList from the productions. Herman needs to provide exact wording for the revised algorithms to Bill.

In section 13, references to FormalParameterList don't need the opt suffix. This is only needed in the production rules.

Herman talked about the example:

```
eval("function(){}")
```

Some discussion after lunch on the scope the optional function identifier is bound to. Currently its not used. Waldemar's proposed change is not yet fully incorporated into the document. Herman suggests that we make a copy of "the" scope chain in 13.1 step 6.

Dave proposes that sections start on new pages, especially section 15 which is the bulk of the spec.

Herman asks that we replace the text "completion type of Result(X)" by X.type.

15.2.4.5 step 3 needs to be supplemented with a note to the effect that [[hasProperty]] is not used since this step doesn't inherit properties. Mike M will find where it was he copied this text from as both places need to be revised. In 15.3.5 we need to add the [[Scope]] property.

In the Array.prototype.* operations in 15.4.4.* we need to specify length values. This applies generally to functions specified as having a variable number of arguments.

15.5.3.2 Richard proposes we replace "UTF-16 encoding of a character" by "UTF-16 code point value". Similarly for 15.5.4.5. In 15.5.4.9 and 15.5.4.10 we ought to add a note that these functions are generic as per the previous functions.

15.8 we want to strike "merely" and to add [[Scope]].

Last para before 15.9.1.9, replace "underlying operating system" by "host system". Is 15.9.5.21 spelled correctly with a lower case "z" in getTimezoneOffset?

In section 15.11 we need to globally replace "Exception" by "Error". In 15.11.4.4 we decided to make toString generic (currently the document notes its not generic). In 15.11.6.4 ArrayLengthError becomes RangeError.

Section 16 is out of date. Bill to replace it by Waldemar's text.

Final Points

Please send your text to Bill by August 1st. Bill will send out reminders.