Minutes of the: ***Ecma TC39, ES3.1WG***

held in: ***Phone conference***

on: ***12 August 2008***

1 Roll call and logistics

1.1 Participants

Doug Crockford (Yahoo!), Pratap Lakshman (Microsoft), Mark Miller (Google), Sam Ruby (IBM) and Allen Wirfs-Brock (Microsoft)

2 Agenda

No agenda circulated ahead of the meeting.

3 Minutes

JSON

What should be the behaviour of JSON serialization for the various new attributes we are adding to properties ? - JSON serialization serializes only enumerable properties - for getter properties it will serialize the value of the property - JSON should not be considered an orthogonal persistence mechanism - need to make the spec language clear on this aspect.

Renaming "usage subset cautious"

Should we rename cautious mode to strict mode ? - just drop "usage" ? - should we use the term 'pragma' ? 'pragma' would be unlikely to appeal to the constituency of programmers who use JavaScript - how about "use strict" ? - do we foresee more than one strict mode in future ? - can imagine a 'use decimal' to make all constants decimal constants - in that case 'strict' 'decimal' can be thought of as the pragmas - and they would be lexical too; "use" followed by a list of pragmas each of which modifies the meaning of the compilation unit.

Decimal

Type testing is an issue since there is no Decimal type yet - also, implicit conversions when infix operators are present with a decimal and a IEEE double (Number) is an issue; dec + 1 will currently cause binary addition and then that can become contagious - this should be the expected - adding a value with undefined precision to a value with a defined precision should result in a value with an undefined precision - is there a significant performance cost to do the addition in decimal ? - what about implementations that have alternate representations for values expressed as integer literals ? SpiderMonkey has them - don't say anything in spec language that precludes a future possibility of optimization.

What is the meaning of '==' and '===' in the context of Decimal? '===' as an identity operator or equality operator ? (-1 === 1m) is false ?! - current behaviour might not be intuitive to the JavaScript programmers - what about (+0 === -0); need to resolve what '==' and '===' mean; don't alter the meaning of '===' - not enough experience programming in multiple math modes in this language - '===' should mean computationally indistinguishable - ok, but note that 1.1m and 1.10m are computationally distinguishable - can we have one NAN one +INF and one –INF ? - why not do all such operations only using APIs that can be made available on Number ? - that can be considered - typeof on primitive decimal values should return 'number'; just widen the range of number values; eases transition path; currently Number has 4 categories (-INF, NAN, finite binary, and +INF); this can be widened to have a 5the category which would be finite decimal - need to think about these some more.

For the next meeting lets make the meaning of ‘===’ and ‘==’ an agenda item.

Meeting adjourned.