Minutes of the: Ecma TC39, ES3.1WG

held in: Phone conference

on: 30 October 2008

1 Roll call and logistics

1.1 Participants

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

2 Agenda

'strict' mode

closure on any remaining spec issues

3 Minutes

'closure on any remaining spec issues

JSON

JSON.parse/stringify pseudo-code looks a little scary; overall, seems to be on the right track, though - it is the json2 implementation translated directly to pseudo-code - perhaps we can use the facility provided by the “Function” constructor to pass in the string representing the JavaScript code for the function body; that would make the pseudo-code a little simpler, and allow us to write the behaviour in JavaScript and use that in the specification; that is what we have done for the MakeArgGetter/MakeArgSetter specification in 10.3.2 - can we leverage the object initialize productions to specify the translation of text to objects? say that if the text conforms to the JSON grammar, it is evaluated as if it were an object literal, and the resulting value is used in later steps - can’t do that because we didn’t change the status of the extra line-breaking characters - JSON.parse does not explicitly tell what to do in the presence of getters; it simply delegates that behaviour to Object.keys.

Should SubStatement be a part of LabelledStatement

Prefer LabelledStatement to only contain a SubStatement - but, worried about legacy, and breaking existing code - not sure if this is a common case - need to raise this on the discuss lists.

Function.prototype.bind

bind should only apply to objects whose [[Class]] is “Function” - what about DOM object, then? - RegExp are callable in some cases (!) - isn’t that a deviation from the ES3 spec? Actually, no, chapter 16 allows implementations to define additional properties - does that mean that RegExp.prototype has as its prototype Function.prototype, for those implementations which have callable RegExps? Also, browsers have callable host objects for which typeof does not return “function” - what is a function? and what is a callable? - de-facto standard: only function objects report typeof “function”, but host objects may be callable; or, all function objects are callable but not all callables are function objects - functions seems to be so broken.

Two simplifications for “bind” (1) typeof should be determined using Class and not “callability” (2) curry over [[Call]] and not [[Construct]]; “bind should return a function that is callable but
not constructable - but that latter can be worked around by putting a try-catch around the ‘bind’.

Do we need a script accessible IsCallable? - instead, do we need a way to get at the [[Class]] property?

**for-in loop enumeration order**

mention that the order “is not specified” - and, it is certainly not controlled by the object; we delete that sentence from the spec - what about properties added to the object during enumeration; is the requirement that they are guaranteed not to be visited in the active enumeration required? - can we condition that on strict mode? - is it onerous on any implementation to support? - lets check on the discuss lists - if we retain that requirement then it will need to be listed in the annexes too.

Meeting adjourned.