Minutes of the: Ecma TC39, ES3.1WG

held in: Phone conference

on: 23 October 2008

1 Roll call and logistics

1.1 Participants

Pratap Lakshman (Microsoft), Mark Miller (Google) and Allen Wirfs-Brock (Microsoft)

2 Agenda

Not circulated ahead of time.

3 Minutes

'const'

Two unsettled design issues; static dead zone -vs- dynamic dead zone - (a) "dynamic dead zone": runtime read barrier (b) "static dead zone": set of static rules that prevent the read - before-initialization, and thereby removing the need for a read barrier - dynamic dead zone is practical, but static dead zone would be better - issue with whether a top level const should create a binding on the global object? - regarding the static dead zone: how difficult are they to specify and how hard are they for programmers to understand? - concern that we don't have adequate time left to get the rules right - need to be able to account for how it should behave as a property - top level const and top level 'let' should be scoped to the program unit as vars; top level program unit more like a block - still, top level function declarations do have to add their name to the global object - perhaps we are better of deferring this instead of hurrying up and risk getting it wrong.

Blocks-introducing-scope

If there were no consts, then these would not have an observable effect any more - not really, we still need to deal with 'catch blocks'; might as well introduce block level scope - but the correction could be a local correction - how do you describe it? - interaction with 'eval' and 'with' make it difficult - we need a new kind of record to describe catch blocks - describe the scope chain in terms of a chain of these records; the environment record (ER) - lets see if we can make a local correction; if that is not possible we can think in terms of the ER.

Updates required for the JSON section (refer comments in the 13 Oct draft on the wiki).

How should we handle cycles in the object graph being serialized? - wait, how do we deal with cyclic arrays today? var a = [1, 2]; a[0] = a; for example - all implementations alert (, 2) - should we correct that too? - one option is not to worry about cycles in JSON (similar to toString on an Array) - currently JSON2.js makes no attempt to detect cycles - in case there are cycles the implementation fails - will try and implement cycle handling using JSPON - it could be prohibitively slow; but in practice most object graphs are small enough that the performance hit is not an issue - still a little concerned about unexpected interactions with toJSON and the replacer - and with getters have side effects - agreed; not evaluated that yet.

One general concern regarding providing the ability to turn off extensibility of objects in a language with no hashtables - but the class of applications being currently developed means we have done nothing worse.

pratapL to update work items list and circulate - Doug to review JSON changes, and Mark to review Function.prototype.bind - next draft update is due on 27 Oct.

Meeting adjourned.