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

held in: ***Phone conference***

on: ***14 October 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

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

Changes related to the following: 'use strict', lexically scoped consts, blocks-introduce-scope.

3 Minutes

scoping of 'const'

current ES3.1 says that 'const' should hoist to the top of the enclosing block (making it consistent with function hoisting) - but there is an email thread arguing for the scope of 'const' to only be the section of a block that is lexically after the declaration - if vars hoist but 'const' does not that could be an inconsistency - but, var and function already have different hoisting behaviours - then, should we add a third behaviour for consts ? - whether 'const' hoists or not we must still deal with the use-before-init issue - what is the behaviour of consts in implementations that already support it ? - IE rejects const, cross-browser web pages don't currently use it; both the ES3.1 effort and the Harmony effort have not taken compatibility with existing browser const behavior as constraining on the standard; ES3.1 does take on the "parse on 3/4 browsers" constraint, but this constrains only syntax, not scoping or semantics - both the "const hoists to block start" currently in draft ES3.1 and the "const doesn't hoist" proposals would be equally incompatible with existing browser semantics and equally compatible with existing browser syntax - not sure if we can resolve this in a timely manner for ES3.1 - perhaps we should consider cutting this ? - not yet, lets discuss more on the discuss lists.

Should SubStatement be a part of LabelledStatement

This is a proposed unconditional grammatical change - but, it is a breaking change from ES3 - interesting that no one on the discuss lists have pointed it out as a breaking change yet - that could be because it is not a common case - should we make it conditional on strict mode ? - raise this on the discuss lists again.

Security concern around algorithms creating new Object where Object is the standard built-in ctor ...

Yes, the Object must be the original primordial Object, and not what its current binding might be (in the case where programs use the name 'Object' for something else) - isn't that a security concern ? Everything reachable from the primordial object is now vulnerable - yes, but systems like Caja (and others) lock down all white-listed state accessible from the global object - fine, this is Ok then.

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

Stringify specifies the default implementation of toJSON for all objects that don't explicitly provide one - need to specify what do we need to detect, and what do we need to forbid - fundamental restriction: stringify must produce a finite length string - (how) should cycles in the object graph be handled ? - actually, a suitable 'replacer' can allow you to serialize a cycle - it might just return different values for the same object at different points during the traversal - what about getters with side effects; what is the effect on the algorithms in that case ? - if we don't enumerate getters, that would be a case where getters don't emulate a data property - lets stringify a DAG as an experiment - what about the behaviour on detecting duplicate keys ? - the current specification does not conform to the RFC - the RFC intended to allow, but not encourage, duplicate keys - that is because the original JSON implementation was eval based and there is no way eval can detect duplicate keys; since we can detect duplicate keys in native implementations, that should be forbidden - gratuitous inconsistency - lets be explicit of what we do in the case of duplicate strings; keep the last one; that is what eval does - makes sense, since each such key would invoke a "Put", the last "Put" wins.

Changes related to the following: 'use strict', lexically scoped consts, blocks-introduce-scope

Go through the decisions list from pratapL and assign names against them - next draft update due on 20 Oct.

Meeting adjourned.