Minutes of the: Ecma TC39, ES3.1WG

held in: Phone conference

on: 15 May 2008

1 Roll call and logistics

1.1 Participants

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

2 Agenda

3 Minutes

Targeted additions to the Object object

http://wiki.ecmascript.org/doku.php?id=es3.1:targeted_additions_to_array_string_object_date
dontEnum, readonly and dontDelete not required - same functionality can be achieved through the defineProperty method

'fix' to be renamed to 'seal' - otherwise the semantics remain the same as in the case of 'fix' - need to have 'keys', 'values', 'hasOwnProperty', 'beget', 'isEmpty', and 'freeze'

'keys' returns an array of all the enumerable and owned keys on an object - 'values' does the same for values (returns an array of all the enumerable and owned values on an object) - is the order of enumeration same as used in for-in? - yes, also the 'values' methods take a 'keys' parameter to control the enumeration; if the keys array parameter is supplied, then it provides the names of the members whose values should be included in the result, in the order determined by the 'keys'.

'beget' takes an old object as a parameter and returns an empty new object that inherits from the old one - if the argument is not an object, then the prototype on the new object is set to Object.prototype - two argument beget preferred - it will copy only enumerable and owned properties of the second arg - beget would read like an object literal

'isEmpty' will return true if the object has no own properties.

''freeze' sets all the properties on an object to readonly, and then seals the result.

'hasOwnProperty' could cause a name collision - need a different name; use 'isOwnPropertyOf' - longer, but that is Ok

All these are static properties on Object - none of them shall be readonly - none of the primordial members are “readonly” - as long as frameworks can lock them down later that should be Ok -

Need a few more static methods on Object - need a 'prototypeOf(obj)' that will return the prototype of the obj passed in as an argument - can be used to walk the prototype chain in a standard manner - current Caja has a kludgy approach to walk the prototype chain - this would be the equivalent of reading the __proto__ property available on FireFox - this would be the flip side of beget - beget constructs an object from its reflective representation, and prototypeOf can be used to get access to the prototype.

Also need an isArray static method on Array - if the input parameter is considered an Array in any frame the method should return 'true' - Ok, will add it as part of the Array proposal.

JSON

Like the JSON2 proposal - allows for the serialization engine to be separate - can guarantee that there will be no syntax errors in the serialized representation - the stringify and parse APIs look good especially with the 'replacer' and 'reviver' parameters - similar to Java serialization where an object can control what about it get serialized but the serialization itself is done by a separate serialization engine - we should adopt the JSON2 API - Doug to write up the spec - can provide the JavaScript code that implements the JSON serializer as informative - however the description still needs to be prose based (like the rest of ES3) - what to do about cycles ? - should it blow up or throw an error ? - very costly to detect cycles - can use a 'depth' value beyond which we can throw an error - well, what does ES3 say in general about what happens when it runs out of memory? - that is not covered in the spec - we should call out in ES3.1 that if a frame runs out of memory, computation within that frame is terminated - but ES3 itself does not acknowledge the notion of multiple instances, and frames - ok, lets add it as guidance for implementers in the annex.

Meeting adjourned.