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

Phone conference

held on: ***15 April 2008***

1 Roll call and logistics

1.1 Participants

Doug Crockford (Yahoo!), Pratap Lakshman (Microsoft), Mark Miller (Google), Allen Wirfs-Brock (Microsoft) and Kris Zyp (The Dojo Foundation).

2 Agenda

Meta

Setting up an ES31 discuss list

Date proposal

Joining and Equating

Function.caller and Function.arguments

3 Minutes

Need to go through the ES4 strict mode proposal - most relevant to ES3.1 at the moment.

Meta

What is the progress on “meta” - only conceptual progress at present - still debating whether it should be a new global object or static methods on the Object object – prefer to have it as static methods on the Object object - avoids issues related to naming this new thing and any resulting naming conflicts - on the other hand a distinct global object could be better too if we end up with a huge suite of methods - allows us to package the methods better - how about methods on Object.Meta. - that would make use of these methods more verbose - we also want that implementations be able to statically recognize the use of these methods.

Caja would prefer to have the “meta” APIs packaged as an aggregate - easier to disable as an aggregate if required - but why would you want to be restrict the usage of such APIs when they can serve as fundamental building blocks - that would crucially depend on what the semantics of these primitives (methods) are - JavaScript does not distinguish between the provider of an object and the consumer of an object. But only the provider of a constructed object should be able to have unconditional control over the provided object.

When we mention “meta” APIs we should consider all of the methods metioned in §8.6.2 (”Internal Properties and Methods”) from ES3 - yes, we should factor that into our design - agreed, “meta” should be used for more than just property attributes - Caja can use these meta level “hammers” to implement the mirrored operations and internal book keeping information. An example is Caja’s ___.primFreeze – an unconditional meta-level hammer, which therefore must not be directly available to untrusted code. Instead, the Caja runtime wraps it with caja.freeze and Object.prototype.freeze_. The first applies primFreeze only to so-called JSON containers (direct instances of Object or Array). The second is protected, and so can only be used by a method within the object-to-be-frozen.

Meta level “hammers” will be unconditionally available - no meta control over the meta level “hammers”. We leave it to safe subset languages to figure out how they wish to wrap them.

ES3.1-discuss list

Should call it es3.x-discuss so that it can be useful even after we move past ES3.1 - any specific preference on where to host it - would be good if we could host it from the same place as the present es4-discuss list - agreed, pratapL to request Mozilla to host a es3.x-discuss list.

Date proposal

Present proposal looks good - need to run it by the ES4 folk - pratapL to edit the ES3 spec with the date proposal and upload to the wiki for review.

Joining and equating

Both of these concepts will be removed from ES3.1 - ES3 spec to be updated (as a separate step after the update for the Date proposal).

Function.caller and Function.arguments

Not there in the ES3 spec, yet implemented in browsers (at least a few of them) - from Mozilla documentation, Function.caller meant to replace deprecated arguments.caller, and Funtion.arguments itself is deprecated - should we codify it in ES3.1? - no! however we should mention this as informative text in Appendix B of the spec - we cannot stop recognizing them either - they are defacto in ES3 in the absence of strict mode - in ES3.1 “strict mode” these shall throw an error.

Allen and Mark cannot make it for the Thurs meeting - Allen will be attending the ES4 meeting - we shall cancel the meeting and reconvene next Tuesday.

Meeting adjourned.