Notes of the: **Meeting of Ecma TC39 ad hoc on Internationalization**

held on: **5 October 2012**

Location: Google, Mountain View, CA, USA

Attendees: Richard Gillam (invited expert, Lab126), Nebojša Ćirić (Google), Norbert Lindenberg (Mozilla), Eric Albright (Microsoft), Allen Wirfs-Brock (Mozilla), Jungshik Shin (Google)

Minute taker: Richard Gillam

1 Timeline

We began with a discussion of the timeline for the next version of the internationalization spec. The first version took over two years, and it sounds like it's impossible to get anything through the process in less than a year, so we settled on a year and a half: We think we can produce the second version somewhat more quickly than the first one because we're more familiar with the process now, but we still need to leave time to get feedback. We'll target completion for June 2014, to present to TC39 in September or November.

2 Prioritization

We spent most of the meeting going through the "wish lists" that were compiled before the meeting, briefly discussing each item, and assigning it an approximate priority. We generally tried to give higher priority to things developers couldn't easily write in ECMAScript itself.

2.1 Text segmentation

Most of the discussion here centered on whether this was even a necessary feature in the first place. There are some people writing text editors in JavaScript, and there's apparently a group doing a PDF renderer in JavaScript, but there was still some question of whether the functionality was common enough to include in all browsers, especially considering the data tables (especially for dictionary-based implementations such as Japanese word breaking) can be large. On the other hand, browsers already have to have most of all of this data just to render HTML. Google mentioned they already have a BreakIterator implementation. The general consensus was that this feature was medium priority.

2.2 String transformations

This includes Unicode normalization, language-sensitive case conversion, and possible case folding (i.e., converting to a case-independent form of the string—this is generally equivalent to converting to upper case except for a few characters that get lost in upper case, such as ß).

The general consensus here is that case conversion and normalization both needed to go in the main ECMAScript spec, not into the i18n spec. Norbert has a strawman for a normalization API (http://wiki.ecmascript.org/doku.php?id=strawman:unicode_normalization) that we should push with TC39, and we should simply tighten the definition of toLocaleUpperCase() and toLocaleLowerCase() to have them take a locale parameter. Norbert has also put together a strawman for this: http://wiki.ecmascript.org/doku.php?id=strawman:case_conversion

Getting this stuff into the main ES draft was considered high priority; we'd like to get it into ES6 is that's possible.

There was no stomach for doing either folding or titlecase. Eric and Norbert pointed out that Unicode titlecasing really doesn't match any set of user expectations: rules for this vary widely and many publishers define their own house rules.

2.3 Character properties

The big question is whether we just want to surface some sort of Unicode-property-test idiom in the Regex API, or whether we need a separate, callable API just for doing Unicode property queries. After a lot of discussion, the consensus was to just put this into the Regex API and not add any new functions, although we fear it's too late to do that for ES6. We might do the lower-level API as a fallback if this turns out to be true. The consensus was that this is high priority in either case. Norbert was delegated to develop a more specific proposal.

2.4 Message formatting

The larger ES community seems to think this is being addressed with "templates strings," (formerly "quasi-literals"), although this solution doesn't provide a way to deal with plurals and gender (and no one but Allen really liked it). We agreed this was high priority, and delegated Nebojsa to investigate more thoroughly and put together a strawman.

2.5 Time zones

We agreed to broaden the existing time-zone APIs to allow the full generality of time zones, not just UTC and the local time zone, and that we would use the IANA (formerly Olson) identifiers. [This was made easier by the fact that IANA is now standardizing the Olson names.] We agreed this change is high priority, and this it only involves minor tweaks to the language in the standard.

2.6 Calendars

There was a fair amount of discussion about adding some sort of "calendar" API that would perform calendrical calculations. We identified three use cases: We need a set of functions to support the writing of date-picker widgets, we might need a way of converting from one calendar system to another, and there are often other calls for operations like "add six days" or "subtract three months." But it's not clear which calendar systems other than Gregorian (which ES already supports) are necessary, HTML5 already has a date-picker widget, and it's theoretically possible to write a library for this in JavaScript (there are no large data tables involved). For these reasons, this was categorized as low/medium priority.

2.7 Alphabetic index

This would be an API to provide support for "thumb index" or "fast scroll" widgets that allow a user to navigate directly to a particular section of a long list. We think the functionality would mainly be getting lists of the "buckets" to categorize items into and possibly some support to make grouping lists into those buckets more convenient. There are a lot of use cases for this kind of thing, and we think it needs to be in a library and not left up to application developers, but the necessary data is small, and it can be implemented in ES, so it was given low priority.

2.8 Language detection

There was general consensus that this was big, complicated, specialized, and hard to standardize and shouldn't be in a general-purpose standard. We agreed this was out of scope for us.

2.9 Encoding conversion and detection

Most of the time, text has already been converted to UTF-16 before it surfaces in JavaScript, so the use cases here basically all revolve around reading legacy file formats and communicating with external libraries that use a non-Unicode character encoding. We tended to agree that these use cases will dwindle over time, so this functionality will decline in value over time. The tables and code are also potentially big and complicated (depending on which/how many encodings an implementer chose to support, or we mandated support

for), and we didn't think we wanted all ES implementers to have to carry them around all the time. Despite fairly strong objections from Google, we agreed this was out of scope and shouldn't be in a general-purpose standard.

2.10 Number and date parsing

A lot of discussion here—do we really need date parsing when it's error-prone and most people want date-picker widgets, for example? (Entering dates into spreadsheet cells was cited as a counterexample.) We generally agreed that basic number parsing was high priority, but that currency, percentage, and date parsing were either low priority or out of scope altogether. (For currency and percentages, we're assuming the currency symbol or percent sign would be supplied in a separate UI widget and the user would just be typing the numeral anyway.)

2.11 DateTimeFormat improvements

A lot of discussion here, and my notes aren't good. One part of the discussion had to do with allowing (or is that requiring?) more choices for format types or allowing full generality. Norbert has a strawman out for this. Another part of the discussion had to do with adding API to support date-picker widgets (getMonthName(), getDayName(), etc.). The general consensus on both issues seemed to be to wait until we have user feedback from the current version of the spec.

Norbert has also proposed exposing the ToLocalTime abstract operation used by DateTimeFormatter as an API to aid various third-party date/time-formatting libraries that are popping up. See https://bugs.ecmascript.org/show_bug.cgi?id=698

2.12 Specialized time formatting

We discussed three different entities as candidates for formatting support: Date intervals (e.g., "January 6-15, 2011"), relative dates ("3 days ago", "next Tuesday"), and durations ("3 hours 15 minutes"). The consensus was that we weren't clear on the requirements and use cases and that somebody should put together a strawman before we discuss it further, but I don't record anybody as having volunteered to take this on.

2.13 Display names for languages, countries, and scripts

Straightforward enough. We agreed this is medium priority. Microsoft can only support getting language and country names in English and the user's current locale (or was it English and the native language?), and they don't have script-name support at all, so we might have to restrict the scope.

2.14 Resource bundles

There's a wide variety of solutions to this problem right now, all responding to different sets of requirements and constraints, and most approaches are outside the scope of ECMAScript. Somebody wondered if the ES module system could be brought to bear on this problem somehow. We decided to put this issue aside for the time being; we need more information to decide whether to tackle this and with how high a priority. Norbert has volunteered to do a little more research on this.

2.15 Bugs

There were a number of smaller issues in TC39's bug tracking database. We agreed to remove the normalization property from the Collator operations and support for the "kk" key in language tags, which does the same thing. This would require that the Collator always make sure the text being compared is normalized. We agreed this should be in version 1.0 of the spec, and Norbert has already updated it and sent out a new draft to TC39 for review.

We also agreed that the "kr" tag (specifying script reordering) should be added to Collator as an optional feature, and that the pseudo-numbering systems "native", "traditio", and "finance" can be supported as input with mapping to real numbering systems.

2.16 Conclusions

This left us with string transformations (i.e., normalization and internationalized case conversion), message formatting (including plural and gender), full time zone support, and character-property queries as the high-priority items. Nebojsa has entered these into TC39's bug-tracking database so we could track everything in one place, and Norbert and Nebojsa are putting together strawman proposals for the larger features (see above).