Locale-sensitive string comparison

Take 3~~2~~, ~~4/29/99~~6/22/99
Richard Gillam

It's looking like we'll have to wait to put out a formal spec for the internationalization library, because it looks like it might have to go into the language (since we have functions in the core language right now that would depend on it). Since it could be a while before we have a standard for internationalization, I'm thinking we should probably add one more function to this spec:

`String.prototype.localeCompare(that, ~~strength~~)`
Performs a locale-sensitive string comparison. *that* is converted into a string, and the two strings are compared in an implementation-dependent fashion. The result is intended to order strings in the sort order specified by ~~be language-specific according to~~ the system default locale, and will be negative, zero, or positive~~either –1, 0, or 1~~, depending on whether *this* comes before *that* in the sort order, the strings are equal, or *this* comes after *that* in the sort order (the actual return values are left undefined to permit implementors to encode additional information in the result value)..

~~The *strength* parameter specifies the level of differences the two strings can have and still be considered equal:~~

~~4~~	~~The two strings must be identical to be counted as equal~~
~~3~~	~~The two strings are counted as equal only if they do not have any "primary, secondary, or tertiary differences" as defined by the underlying environment. (With most locales and implementations, this will be the same as 4.)~~
~~2~~	~~The two strings are counted as equal only if they do not have any "primary or secondary differences" as defined by the underlying environment. (This generally means that case differences and ignorable-character differences don't count, but other differences do.)~~
~~1~~	~~The two strings are counted as equal only if they do not have any "primary differences" as defined by the underlying environment. (This generally means that the strings must have "a different letter," according to the locale, in at least one position.)~~

~~The *strength* parameter is always converted to a number. Values greater than 4 are treated as 4, and values less than 1 are treated as 1.~~

This function is intended to rely on whatever language-sensitive comparison functionality is available to the ECMAScript environment from the underlying operating system, and to compare according to the rules of the host environment's current locale. It is strongly recommended that this function treat strings that are canonically equivalent according to the Unicode standard as identical (in other words, compare the strings as if they had both been converted to Normalized Form C or D first). It is also recommended that this function not honor Unicode compatibility equivalences or decompositions.

If no language-sensitive comparison at all is available from the underlying environment, this function may do bitwise ~~comparison for strengths 2, 3, and 4, and behave as though it performed toUpperCase() on both strings and then did a bitwise comparison for strength 1~~.

~~**NOTE:** The concept of collation strength is highly language-specific: In English, case differences and the presence or absence of most ignorable characters are tertiary differences: “email”, “Email”, “e-mail”, and “E-mail” will all compare as equal with a strength of 2 or 1. Diacritic differences are secondary differences: “résumé” and “resume” will compare as equal with a strength of 1. Differences in actual letters are primary differences: “resume” and “resound” will always compare different. In Swedish, however, a and å are different letters (a primary difference), and will always compare different, while v and w are variants of the same letter (a secondary difference) and will compare equal with a strength of 1. [The concept of collation strength levels in used in Java, the Unicode collation algorithm, the ISO 14651 collation algorithm, and the POSIX library, among others.]~~

~~When sorting a list of strings, pairs of strings should be compared using a strength level of 4. Lower strengths will leave parts of the list in arbitrary order. A strength of 4 will give a well-defined order on all strings in the list (except, of course, for those that are actually identical). The strength parameter is useful when two strings are being tested for equality.~~