Case conversion functions are currently limited to ASCII characters. When used with formal languages, that may be appropriate, but it is often not appropriate for applications.

In order to avoid backwards-compatibility problems and to make sure that the various variants of case conversion (e.g. language-dependent) can be selected, we propose to add an optional parameter to the case conversion functions.

Our current design idea is as follows:

ASCII-only if no parameter:
"Türkiye".upcase # => 'TÜRKİYE', note lower-case ü

Parameter triggers (general) Unicode conversion:
"Türkiye".upcase 'en' # => 'TÜRKIYE', note upper-case Ü

The parameter is actually a BCP 47 (http://tools.ietf.org/html/bcp47) language tag. This means that for languages with special case conversion rules, such as Turkish, this works as follows:
"Türkiye".upcase 'tr' # => 'TÜRKİYE', note upper-case İ (with dot!)

In the second example, we used 'en', but most other languages would work, too, because a single case conversion works for most languages. Turkic languages are the biggest exception.

The Unicode standard also defines various cases of "case-folding", which usually is lossy, e.g. mapping German ß to ss and so on. It should be possible to include this functionality in this proposal, e.g. by using symbols or CONSTANTS for the few specific foldings. It may also be possible to define a reversible variant of case conversion in particular for use with swapcase.

In the long term, instead of a direct BCP 47 string, we could create a Locale class that would incorporate language-specific facilities, but this may need more detailed considerations.

The idea of using an additional parameter to indicate language-dependent or other processing variants should be extensible to areas such as number-to-string conversion and date formation. While this proposal is only about case conversion, we should check that there is a good chance to use similar parameter conventions for such extensions.

[This proposal is based on research done together with my student Kimihito Matsui.]

Related issues:
- Related to Ruby master - Bug #3376: russian support added
  - Related to Bug #3376: russian support
  - Closed 06/01/2010
- Related to Ruby master - Feature #2034: Consider the ICU Library for Improving and Expanding Unicode Support added
  - Related to Feature #2034: Consider the ICU Library for Improving and Expanding Unicode Support
  - Rejected
- Related to Ruby master - Feature #10002: String swapcase
  - Related to Feature #10002: String swapcase
  - Closed
- Related to Ruby master - Feature #10152: String#strip doesn't remove non-breaking spaces
  - Related to Feature #10152: String#strip doesn't remove non-breaking spaces
  - Open 08/19/2014
- Related to Ruby master - Bug #10550: Resolv::DNS.getaddresses returns no IPs
  - Related to Bug #10550: Resolv::DNS.getaddresses returns no IPs
  - Closed 11/26/2014
- Has duplicate Ruby master - Bug #11284: String#upcase and String#downcase don't work with non-ASCII
  - Related to Bug #11284: String#upcase and String#downcase don't work with non-ASCII
  - Closed
  - Rejected

History
#1 - 07/23/2014 11:06 AM - duerst (Martin Dürst)
  - Related to Bug #3376: russian support added

#2 - 07/23/2014 11:06 AM - duerst (Martin Dürst)
  - Related to Feature #2034: Consider the ICU Library for Improving and Expanding Unicode Support added

#3 - 07/23/2014 11:07 AM - duerst (Martin Dürst)
I want default case conversion should be Unicode aware (when encoding is Unicode). The previous behavior can be done by str.downcase(:ascii).

Non unicode encoding (e.g. Latin-1) can support non ASCII case conversion, but not mandatory.

Matz.

The related issue, [Bug #10550] Resolv::DNS.getaddresses, needs ASCII-only case conversion. Unicode aware case conversion is not suitable for the issue. See RFC 4343.

Close way overdue, should have happened somewhere around r55281.

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>CaseConversion.pdf</td>
<td>340KB</td>
<td>07/23/2014</td>
<td>duerst (Martin Dürst)</td>
</tr>
</tbody>
</table>