Richard Ishida

i18n @ W3C

Richard Ishida
W3C’s goal is a Web for all, regardless of language, script or culture. The Web community has made tremendous progress in internationalizing the Web over recent decades, but as Web penetration in language communities increases, as usage scenarios grow, and as new applications such as digital publishing emerge, there remains more to do.

For the Web to truly work for stakeholders all around the world, there must be a collaboration of language experts, Web site designers, developers, and vendors who are active in moving the Web forward. To ensure a rapid response to the growth of the Web, the W3C wants to marshal the resources of organizations and experts who care about these problems and enlist their help in strengthening internationalization support for the Web.

To accelerate progress in this area, the W3C is also looking to supplement the core funding it receives from W3C Member fees so that it can increase in-house resources dedicated to this work.

The internationalization initiative will provide participants and funding to address three main aspects of the Internationalization continuum:

- **Language enablement** appeals most directly to stakeholders (e.g., governments, publishers, community groups, etc.) who utilize the language.
- **Developer support** appeals most directly to tech companies that are building the infrastructure for a global Web and supporting W3C standards groups.
- **Author support** appeals to people creating Web content in their own language, as well as to companies who build or localize Web sites in many languages.

The Web needs your help

Success in meeting these goals requires participation and funding from language, developer, and author communities, in order to expand the effort over and above what can be achieved with our core funding.
• **Language enablement**
  understand where the gaps are for users of the global Web

• **Developer support**
  build standards & applications that support a global Web

• **Author support**
  help people create content in their own language, or create content that will be localised
language matrix

w3c.github.io/typography/gap-analysis/language-matrix

For the 79 languages currently listed, we see the following problem areas. This page is updated as information becomes available.

- **33** languages need work for advanced publishing
- **25** languages need work for basic features
- **2** languages don’t work well on the Web
- **47%** of cells still need investigation.
The following are the basic elements of a page format. Fig. 11 illustrates an example of a page format in vertical writing mode.

a. Trim size and binding side (vertically set Japanese documents are bound on the right-hand side, and horizontally set documents are bound on the left-hand side. See Fig. 12.)

b. Principal text direction (vertical writing mode or horizontal writing mode).

c. Appearance of the kihon-hanmen and its position relative to the trim size.

d. Appearance of running heads and page numbers, and their positions relative to the trim size and kihon-hanmen.

Fig. 12: Binding-side (bound on the right-hand side and bound on the left-hand side).

NOTE

Establishing a kihon-hanmen may be seen as defining not only a rectangular area on a page, but also within that area an underlying, logical grid, to guide the placement of such things as characters, headings, and illustrations. However, once a kihon-hanmen is established, there is no absolute requirement to align characters with the grid, especially when setting characters inside a line. The only factors that influence the placement of characters are strong gravitational forces that (i) attract the first and last characters on a line to align with the border of the kihon-hanmen, and (ii) attract each line position to the line positions on which the kihon-hanmen is based.

It may help in understanding the basic concepts of Japanese layout and kihon-hanmen to think in terms of a slit-based model, rather than a grid-based model. Each slit is the full length of the lines on which the kihon-hanmen is based.

2.2.4 Elements of Kihon-hanmen

The kihon-hanmen is the hanmen style designed as the basis of a book. The following are the basic elements of the kihon-hanmen (see Fig. 13):

a. Character size and typeface name

b. Text direction (vertical writing mode or horizontal writing mode)
2.3.1 Arrangement of ‘Letter Face Position in Character Frame’ for Full Width Parentheses

In horizontal writing, the letter face of a full-width opening parenthesis is placed on the right end of the character frame, and the left space is considered a user-controlled area. In vertical writing, the letter face of a full-width opening parenthesis is placed on the bottom end of character frame, and the space is considered a user-controlled area.

![Fig. 7: An example of positioning for bilingual annotations](image)

See [https://github.com/wiki/kreq/issues/10](https://github.com/wiki/kreq/issues/10)

2. Interlinear Comments

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.

![Interlinear Comment](image)

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>行间注释</td>
<td>Interlinear comments</td>
</tr>
</tbody>
</table>

**行间注释**

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>行间注释</td>
<td>Interlinear comments</td>
</tr>
</tbody>
</table>

**行间注释**

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>行间注释</td>
<td>Interlinear comments</td>
</tr>
</tbody>
</table>

**行间注释**

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>行间注释</td>
<td>Interlinear comments</td>
</tr>
</tbody>
</table>

**行间注释**

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>行间注释</td>
<td>Interlinear comments</td>
</tr>
</tbody>
</table>

**行间注释**

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>行间注释</td>
<td>Interlinear comments</td>
</tr>
</tbody>
</table>

**行间注释**

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>行间注释</td>
<td>Interlinear comments</td>
</tr>
</tbody>
</table>

**行间注释**

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.

<table>
<thead>
<tr>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>行间注释</td>
<td>Interlinear comments</td>
</tr>
</tbody>
</table>

**行间注释**

Interlinear comments are ways to annotate the meaning of text fragments or a single word, and are so named for their interlinear positioning. They usually lie in the interlinear space and co-exist with the body text. Compared to other annotation methods, i.e. headnotes or footnotes, interlinear comments are more compact and slick better to the body. These kinds of comments are often found in ancient books, such as Rouge Note, an early commentary of the novel Dream of the Red Chamber.
2.6.3 Arabic Script and Typography

Arabic script has some characteristics that are challenging for typographers and font designers. Examples below show some characteristics worth to be considered carefully. How could typography, which came late to the Arabic world, then follow the tradition of the many authors and artists who manually shaped the Arabic writing over decades? Even in it’s simplest Naskh style?

1. Multi-level baselines

Letters may join through a finely inclined line

or two, square-ended lines

Multiple baselines don’t occur in all fonts. The above examples use the Arabic Typesetting font. Compare those examples to to more typical fonts:

2. Multi-context joining

Rendering of letters depends not only on their place in the word (initial, medial, final) but also on their neighboring letters, i.e. the letter they join with. Each letter has a different appearance in each combination.

3. Words as groups of letters

Letters in order to be distinguished in some contexts, such as ا ل, vs. ا ل, or to resolve ambiguity. See also the section about sleep letters below.

3.3.2 Justification When ETHIOPIC WORDSPACE is the Word Delimiter

Since the arrival of the printing press in Ethiopia in 1668 (Pankhurst, 1996), full justification of Ethiopic has been a common typesetting practice in Ethiopian, and later Eritrean, publishing houses. Earlier, Ethiopic justification rules are a feature of Hode Lulud’s Historia Ethiopica, which is noted as the first use of movable type for Ethiopic script (Lutz, 1681). Prior to letterpress typography, calligraphic manuscript rendered on parchment also featured full, or approximately full, justification. Though the latter likely reflects the scribe’s desire not to waste a millimeter of available letter writing space.

The placement of Ethiopic wordspaces presents a complication to the justification of Ethiopic text. Two placement styles developed in typeset literature which will be referred to here as “word bound” and “centered” styles. Additionally, the word spacing following an Ethiopic fullstop may (or may not) be governed by a special rule and in combination with the two wordspacing spacing styles. These spacing rules are discussed in the following sections.

3.3.3 Justification with Word Bound Wordspace and Punctuation

In keeping with the justification for Latin script, the non-printed or “blank space” (space and gaps) between words is treated as stretchable. The width of the space symbol itself will be elongated to some aesthetic width value that may vary from space symbol to space symbol across a printed line. In Ethiopic justification, the blank space between the Ethiopic word separator and the words it separates is likewise allowed to stretch. This stretching of blank space may be either symmetrical (“centered”) or asymmetrical but in the latter case space stretching is always between the right side of the separator and the following word—referred to here as “word bound”.

In “word bound” justification the word separator, which may be either a punctuation symbol or \( \text{U+0135} \) ETHIOPIC WORDSPACE ( ), appears to adhere to the word to the left as if it were its final character. Figures 25, 26, Ethiopic justification in word bound style (Erikson, 1921 [1913 EG]) and Fig. 26, Ethiopic justification in Historia (Ethiopica) (Ludolf, 1681) both illustrate the word bound style.
3.3 Initial letter styling

Drop initial is a typographic effect emphasizing the initial letter(s) of a block element with a presentation similar to a 'flagged' element.

3.3.1 Selecting initial letters

Initial letters in Indic scripts must be selected on the basis of orthographic syllables, rather than individual letter forms (see an example at the end of section 3. Text segmentation). A detailed definition of Indic syllables can be found in section 6. Indic Syllable boundaries in Indian languages the size of the Initial Letter is determined by the number of the lines between top line of the syllable and lowest point in the orthographic Indic syllable cluster where subjacent consonant and other diacritics appears.

3.3.2 Typical drop initial usage in Indic scripts

Most of the Indic drop initial letters in magazines and newspapers use 2 to 4 line drops. Some examples are shown below.

- Fig. 4: Examples of Indic initial letters

The Sunken and raised Initial letter are not preferred in Indian languages. In examples of this kind, reference points on the drop cap must align precisely with reference points in the text. In Indic scripts the top reference point is the hanging baseline for those scripts that have one, and the bottom alignment point is the last letter edge.

Initial letter wrap property is not applicable for Indian languages. No contour filling is required in Indian languages.

Alignment of the top line of the non-highlighted characters is at the top of the thicker top line of the initial letter is commonly used in India. In some examples top lines of the initial letter and the following letters don’t touch. This is due to variable technology/formats used by the publishers. It is preferred that both the top lines of Initial letter and neighboring text should touch. Here are some additional examples of initial highlighted letter and drop letter based on the Indic syllable definition.

2.3 Tibetan Syllables

Word boundaries within a section are not indicated, only ‘syllables’, known as tshag bar ‘/tshag bar/’. Syllable boundaries are usually separated by the slash character, tshag bars TIBETAN MARK INTER SYLLABIC TSHEG / 

The pronunciation of Tibetan words is typically much simpler than the orthography, which involves patterns of consonants. These patterns reduce ambiguity and can affect pronunciation and tone.

The following diagram shows characters in all of the syllabic positions, and lists the characters that can appear in each of the non-root locations. The two-syllable word in the example is नौरूस (nouru = 'towards' pronunciation).

![Tibetan Syllable Diagram](image)

2.3.1 Structural Rules

The primary consonant in a syllable is called the root consonant (or root consonant), and the other consonants in the syllable (usually up to 6) in total are called or. The following rules help identify the root:

1. A consonant with a vowel is always the root, unless it is the phrase connector, and letters with superscripts or subscripts are not root consonants.

2. In a 2-consonant syllable with no vowel, the first consonant is always the root.

Additional cases include syllables with no root consonant, or in a complex syllable, where the root consonant is shared with another syllable.
breaking news: mlreq

w3c.github.io/mlreq/

---

### 2.4.2 Display rules for Mongolian space

Mongolian case supplementary component display rules and Mongolian space (0x202F) processing.

The inputting of Mongolian case supplementary component into the computer requires the Mongolian space 0x202F first, and then the corresponding Mongolian script. For example, the internal code of the word "аы" is: 0x202F 0x1624 0x1688, among which 0x202F is called Mongolian space, inputted before the Mongolian case supplementary component.

---

**Figure 2:** Alignment relationship between punctuation and Mongolian script text.

---

**Figure 3:** The differences between the common space and Mongolian space.

Mongolian space (0x202F) and case supplementary component cannot appear at the beginning of a line, for example. **Figure 4** is the correct processing method and **Figure 5** is the wrong processing method.
Japanese Gap Analysis
W3C Editor's Draft 12 June 2018

This version:
https://dev.github.io/w3c-gap-analysis/
Latest published version:
https://www.w3.org/TR/w3c-gap-
Latest editor’s draft:
https://w3c.github.io/w3c-gap-
Bug tracker:
File a bug (open bugs)
Editor:
Richard Ishida (W3C)
Github:
repository

Copyright © 2016 W3C™ MIT, ERCIM-Keio, ALINtel. W3C liability, trademark and permissive document license rules apply.

Abstract

This document describes and prioritizes gaps for the support of Japanese on the Web and in eBooks. In particular, it is concerned with text layout. It checks that needed features are supported in W3C specifications, in particular HTML and CSS and those relating to digital publications. It also checks whether the features have been implemented in browsers and eReaders. This is a preliminary analysis.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

This document describes and prioritizes gaps for the support of Japanese on the Web and in eBooks. In particular, it is concerned with text layout. It checks that needed features are supported in W3C specifications, in particular HTML and CSS and those relating to digital publications. It also checks whether the features have been implemented in browsers and eReaders. This document complements the document Text Layout Requirements for the Japanese Script, which describes the requirements for areas where gaps appear. It is linked to from the language matrix that tracks Web support for many languages.

This document is currently an individual contribution.

NOTE

Sending comments on this document

If you wish to make comments regarding this document, please raise them as github issues.
Only send comments by email if you are unable to raise issues on github (see links below). All comments are welcome.

To make it easier to track comments, please raise separate issues or emails for each comment,
2.11.2 Double-sided ruby

In order to get double-sided ruby to appear on either side of the base text in HTML, you need the CSS `ruby-position` property to work with the value `under`. It currently works per the standard only with Firefox. It works with the proprietary `webkit` prefix in Chrome and Safari, but doesn’t work with Edge. See test results.

Double-sided ruby doesn’t appear often in Japanese text, but it is used, and should be available to content authors.

2.11.3 Ruby alignment and other styling

For background information see `Aligning annotations and bases`

Fine control in HTML over the placement of ruby annotations relative to the base require the availability of the CSS property `ruby-align`. This only works in Firefox, and sometimes in Edge with proprietary syntax. See results for major browsers.

The impact of this is probably more important for advanced typographic layout.

Needs work for basic styling support.

2.12 Emphasis & highlights

Bold and italics are not always appropriate for expressing emphasis, and some scripts have their own ways of doing it, that are not in the Western tradition at all. Does this script require support for emphasizing or highlighting text that cannot be achieved currently? See available information or check for current needs data.

For requirements, see JLReq `Composition of Emphasis Docs`.

2.12.1 Boten marks

There are tests for boten mark support that show that they are not supported by Chrome 59.0.3229.118 or Edge, but they are supported by Firefox 58.0 and Safari 11.0.3.

The expected behavior for basic support of boten marks is specified in CSS.

Given that there are alternative ways of showing emphasis, and although there are already two implementations, I mark this as an advanced need.

Additional requirements arose while the text was being written for CSS. They relate to text that is annotated with ruby at the same time as annotation marks. The conclusions were:

1. Emphasis marks go outside ruby.
2. If ruby only spans part of the emphasised word, emphasis marks stay as close as possible to the base.
3. Where a ruby annotation is hidden or empty, the emphasis marks should continue at the same height.

JLReq also requires that emphasis marks not appear over commas, full stops and brackets.

There are currently no tests for these behaviors, and no requirements in JLReq for the former (with ruby). Need to take a closer look.

JLReq calls out the usual practice of using sesame shaped boten for vertical text, and bullet for horizontal. Controls for this exist in CSS, and tests show that it is supported by the browsers that support boten.

The default side for boten is to the right of vertical lines and above horizontal lines, according to JLReq. CSS controls allow this positioning to be set by the author, and in fact it is the default in both browsers that support boten for text where a lang tag identifies it as Japanese.

Needs work for advanced level support.
phase 3 – networks

w3c.github.io/sealreq/

Southeast Asian Layout task force

This group exists to allow a network of experts to share information about gaps and requirements for support of the Southeast Asian scripts on the Web and in ebooks.

The main problem we seem to face at the moment is that experts don’t know how to tell the W3C what support of their script is needed on the Web, and the W3C doesn’t know how to contact people who can help when there is a request. This network of experts should help to significantly reduce that problem.

Some experts may go a step further, and contribute to a gap-analysis or requirements document for a given script.

Topics for discussion are suggested by the gap-analysis template. This work supports the development of the Aspen project for indicating hot-spots for language support.

Documents

- Javanese Script Gap Analysis
- Khmer Script Gap Analysis
- Lao Script Gap Analysis

Feedback

Please use the GitHub issue list to report issues for language support, for discussions, and to send feedback.
notifications

w3c/sealreq (+0/-0/3)
2 issues received 3 new comments:
- #17 Is inter-character spacing used in Lao? (2 by leonux, jdark) lao question
- #2 Do Javanese & Balinese lines break at syllable or word boundaries? (1 by adtbayuperdana) balinese javanese question

w3c/csswg-drafts (+1/-0/9)
1 issues created:
- #2975 [css-text-4] hyphenate-character doesn't just put hyphen at end of line (by r12a) i18n-sealreq

3 issues received 9 new comments:
- #2975 [css-text-4] hyphenate-character doesn't just put hyphen at end of line (7 by r12a, Crissvo, kojishi, litterum) i18n-tracking
- #2976 [css-text-4] Dealing with unusual line-break/hyphenation rules (1 by r12a) i18n-sealreq i18n-treq i18n-tracking
- #2809 [css-text-4] hyphenate-character doesn't accept just a character (1 by r12a) css-text-4 i18n-tracking
issue tracker

w3c.github.io/i18n-activity/textlayout/
2.11 Ruby annotation

Ruby is used for phonetic and semantic annotations of East Asian text, including furigana, pinyin and zhuyin hualuo systems. In addition to positioning annotations along the correct side of the base text, there are many fine adjustments of the annotation and base text to support.

Requirements

- Chinese Layout Requirements: Intercolumn annotations
- (Chinese) Bopomofo on the Web
- (Chinese) The Manual of the Phonetic Symbols of the Mandarin Script (English Chinese
- Japanese Layout Requirements: Ruby and Emphasis Dots · Positioning of Jukugoro-ruby
- (Japanese) Use Cases & Exploratory Approaches for Ruby Markup
- Implementing Japanese Subtitles on Netflix: Rubies

GitHub resources

- Requests for information
- Spec issues
- Browser bugs
- Type samples

Spec links

- HTML5: The ruby element
- CSS3 Ruby

Tests

- HTML5, the ruby element and its children
Type samples

This page lists pictures in the type-samples.github repository. There are 62 items.

Select a feature

Filter by script

Filter by medium
breaking news: jlreq

w3c.github.io/jlreq/charter/

- Japanese gap analysis
- JLReq errata & improved usability
- Produce new, informative documents
- Issue list discussions
Language Enablement

- Language matrix
- Expert networks
- Gap-analysis
- Layout requirements
- Text layout index
- Tracker tools & notifications
Developer support
spec reviews
github.com/w3c/i18n-activity/projects/1
Review comments tracker

This page tracks comments made by the I18n WG on the specs of other WGs. It only tracker issues the I18n WG has not closed. (Issues may remain open in the I18n track if closed by the other WG in their repo.)

The mostly recently changed issues appear at the top of each section, and sections are ordered according to where the most recent changes occurred. The date indicates the time there was a change to the tracker issue (not the issue in the other WG’s repo). Each item links to a tracking issue in the I18n-activity github repo. Click on the link in that list to follow the actual discussion.

There are 255 issues.

webauthn

- Truncation to 64-byte upper limit doesn’t mention character boundaries #973
  - aphpillips commented on 27 Jun · 20 comments
  - aphpillips commented on 27 Jun · added

  Hmm... also I think it doesn’t mean to say “64-byte "maximum" length.” I suspect it means to say “maximum” there. PS: Please add the 18n-comment label.

- css-ruby
  - Add over-max-under-last value to ruby-position & text-emphasis-position for captioring
  - aphpillips commented on 27 Jun · edited

  [This issue is related to issue #565 and RR #951]
  - aphpillips wrote:
    - Note that the specification does not require truncation on a Unicode character boundary
    - I was wondering whether you’d bring this up.

  I’ve done some modest research on this topic of “unicode string truncation” (due to the text you cite above) and apparently it is more complex than simply performing truncation on a Unicode character boundary. It apparently ought to properly be done on extended grapheme cluster (EGC) boundaries.

  I found detailed analysis here: https://jaytech.github.io/translate-presentation/... and a library: https://github.com/jaytech/Unicode-Triuncate, but nothing regarding “unicode string truncation”
Internationalization Best Practices for Spec Developers

W3C Editor's Draft 07 August 2018

This version: https://specdev.github.io/specdev/
Latest published version: https://www.w3.org/TR/international-specs/
Latest editor's draft: https://specdev.github.io/specdev/

Bug tracker: File a bug (open bugs)

Editor: Richard Ishida (W3C)
Github: specdev


Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

This document provides advice to specification developers about how to incorporate requirements for international use. What is currently available here is expected to be useful immediately, but is still an early draft and the document is in flux, and will grow over time as knowledge applied in reviews and discussions can be crystallized into guidelines.

4.13 Truncating or Limiting the Length of Strings

Some specifications, formats, or protocols or their implementations need to specify limits for the size of a given data structure or text field. This could be due to many reasons, such as limits on processing, memory, data structure size, and so forth. When selecting or specifying limits on the length of a given string, specifications or implementations need to ensure that they do not cause corruption in the text.

After: turn-0-1624

Specifications SHOULD NOT limit the size of data fields unless there is a specific practical or technical limitation.

There are many reasons why a length limit might be needed in a specification or format. Generally length limits correspond to underlying limits in the implementation, such as the use of fixed-size fields in a database or data store, the desire to fit into practical boundaries such as packet size, or some other implementation detail related to storage allocation or efficiency.

When truncating strings, it’s necessary to decide what units to use when counting the size of the string. In many cases this is beyond the control of the specification, since the truncation is occurring for some preordained reason, such as the size of a fixed-length field in a file format or database. However, when the choice is available, some general guidelines can be applied.

Below you can see the effect of truncating strings in different scripts on an arbitrary code unit boundary. In this case, each string is encoded in UTF-8 and truncated after the 38th byte. There are several things to notice here.

First, as the number of bytes-per-character goes up, the number of characters available inside the byte count limit goes down. ASCII has four times the available characters than emoji, three times as many as languages such as Chinese, and roughly twice as many as the Cyrillic example.

Second, in each of the non-ASCII examples the byte boundary for truncation falls in the middle of a character. The resulting “dangling byte” are rendered as U+FFFD and the byte sequence itself is not valid UTF-8. If the byte sequence were then serialized into a file format, such as JSON or CSV, the surrounding file might not be wholly valid or the file processor might generate errors because the byte sequence itself is invalid. Unlike many legacy character encodings, UTF-8 is highly patterned, so the longest broken character sequence that can result from mid-character truncation is one character. By contrast, in many legacy encodings, a file or document containing a mid-character truncated string can be wholly changed or rendered unintelligible after that point.

- ASCII
- Cyrillic
- Emoji

Specifications that limit the length of a string MUST specify which type of unit (extended grapheme clusters, Unicode code points, or code units) the length limit uses.

Specifications that limit the length of a string SHOULD specify the length in terms of Unicode code points.

If a specification sets a length limit in code units (such as bytes), it MUST specify that truncation can only occur on code point boundaries.

If a specification needs a length limit, it SHOULD provide an algorithm that enforces the limit on any implementation in an indeterminate manner.

Related Entities:
- Other truncate units
- Other truncate byte boundary
- Other truncate code point boundaries
- Other truncate indicators
self-review checklist

w3.org/international/techniques/developing-specs
in development
Developer support

String metadata (bidi & language)

Review radar

Self-review checklist

Guidelines for developers
Articles, best practices & tutorials

You can also find resources using the Technique index and Topic index, which provide more fine-grained access to information.

Getting Started

Getting Started with the W3C 11th site
Introducing Character Sets and Encodings
Language on the Web
Internationalization Quick Tips for the Web

Characters

Handling character encodings in HTML and CSS (tutorial)
Character encodings for beginners
Character encodings: Essential concepts
Choosing & applying a character encoding
Declaring character encodings in HTML
Declaring character encodings in CSS
The byte-order mark (BOM) in HTML
Normalization in HTML and CSS
Characters or markup?
Changing an HTML page to Unicode
Using character escapes in markup and CSS
Document character set
Setting the HTTP charset parameter
Setting charset information in .htaccess
Checking HTTP Headers
Checking the character encoding using the validator
HTML, XHTML, XML and Control Codes
Missing characters and glyphs
Who uses Unicode?
Migrating to Unicode

Language

Working with language in HTML (tutorial)
vertical text guidelines
i18n test suite

w3.org/international/tests

Summarized test results:
CSS3 Writing Modes, vertical text

Intended audience: users, HTML coders, script developers, CSS coders, Web project managers, and anyone who wants to know whether browsers support the CSS Ruby spec.

Updated 2016-12-02 12:07

These tests check whether user agents correctly apply the writing-mode property per the CSS3 spec for the vertical-rl and vertical-rr values. They are just essential tests. More detailed tests for edge cases and finer aspects of rendering can be found in the CSS test suite.

To see the test, click on the link in the left-most column. To see detailed results for a single test, click on a row and look just above the table. The detailed results show the date(s) the test result was recorded, and the version of the browser tested.

Any dependencies are shown in notes above the table, and notes below the table will usually provide any additional useful information, including an explanation of why a result was marked as ‘partially successful’.

Key:
- pass
- fail
- partially successful

The proprietary test results are for either preferred implementations, using -webkit- or -ms-, or for the nightly version of Firefox, or for non-standard writing-mode values in Internet Explorer.

If writing-mode-vrl-001 or writing-mode-vrr-001 fails, or either of the corresponding -prop tests, the remaining tests for the section can be ignored.

vertical-rl

Basics

Assertions:
- writing-mode-vertical-rl will display a line of text vertically.
- writing-mode-vertical-rl will wrap lines from right to left.
- writing-mode-vertical-rl will cause text to display from the right side of the containing box.

Proprietary syntax

Assertions:
- writing-mode-vertical-rl will display a line of text vertically.
- writing-mode-vertical-rl will wrap lines from right to left.
- writing-mode-vertical-rl will cause text to display from the right side of the containing box.

Glyphs

Assertions:
- By default, writing-mode-vertical-rl will display Chinese characters upright.
- By default, writing-mode-vertical-rl will display Arabic characters rotated 90° right by 180°.
- Writing-mode-vertical-rl will display German characters rotated 90° right.
- By default, writing-mode-vertical-rl will display Arabic characters progressing up the page.
Internationalization techniques:
Authoring HTML & CSS

This page lists links to resources on the W3C Internationalization Activity site and elsewhere that help you author HTML and CSS for internationalization. It is one of several techniques pages.

You can see a list of updates to this document. You can also raise an issue about this page.

Collapse all  •  Expand all

Characters
Language

Getting started
Declaring the overall language of a page
Identifying in-document language changes

Choosing language tags

Use subtags as defined by BCP 47 for language attribute values. more
Use the shortest possible language tag values. more
Where possible, use the codes zh-Hans and zh-Hant to refer to Simplified and Traditional Chinese, respectively. more
Use the subtag zxx when the text is known to be not in any language. more
When the language is undetermined and you have to label it, use lang="". more
If you are serving XML and the format you are using supports it, use xml:lang=""; otherwise use xml:lang="und" when the language is undetermined and you have to label it. more

How to's

• Choosing a Language Tag
  Which language tag is right for me? How do I choose language and other subtags? Covers all the subtag types in the latest version of BCP47.
• Language tags in HTML and XML
  A simple overview of the syntax for language tags in BCP 47.
• Tagging text with no language
  How do I use language markup in HTML or XML content when I don't know the language, or the content is non-linguistic?
• Two-letter or three-letter language codes

w3.org/International/techniques/authoring-html
Detailed report

Conflicting character encoding declarations

Explanation

The following character encoding declarations are inconsistent:

a. `<meta charset="iso-8859-1"/>

b. `<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

Browsers will apply precedence rules to determine the character encoding to use for the page, but this may not be the encoding you intended.

What to do

Change the character encoding declarations so that they match. Ensure that your document is actually saved in the encoding you choose.

Further reading

Character encodings explained
Choosing a character encoding
Changing the encoding of a document

Multiple encoding declarations using the `meta` tag

Content-Language `meta` element used

A language attribute value was incorrectly formed

A language subtag is invalid

A `lang` attribute value did not match an `xml:lang` value when they appeared together on the same tag.

Non-UTF-8 character encoding declared

Non-preferred names used for legacy character encoding

Found Unicode code points for directional controls

Unpaired directional controls found

Invalid `b` tags found with no class attribute
Author support

Internationalisation articles
Techniques index
Internationalisation test suite
Internationalization checker
next steps

- Widen participation of expert networks in Ireq groups
- Increase scope and documented output of language enablement work
- Significantly improve specdev guidelines and the self-review checklist for developers
- Investigate ways to extend the i18n test framework to support tests and results for paged media generators
what you can do

• Join a layout network as follower or contributor.

• Contribute to creating gap-analysis docs.

• Learn about strings and internationalization.

• Use the information in our articles and test results.

• Check your pages with the i18n checker.

• Support the sponsorship program.
Thank you

w3.org/International/talks/1809-tokyo/

All full-page photos in this presentation are © Richard Ishida