Title: Proposal to add a property for auto inter-script spacing
Authors: Koji Ishii, Yasuo Kida
Date: Dec 12, 2023

This is a proposal to add a property for auto spacing. Initially this property supports inserting inter-script spacing for CJK typography, but it may extend to support other scripts in future, such as French Typographical Rules for Punctuation.

In CJK typography, the insertion of small spaces (usually 1/8em to 1/4em, see the Space section below) at script boundaries helps the readability for many cases as mandated by standards such as JIS X 4051 and JLReq that describe layout requirements for books. CJK text usually consists of multiple scripts; Han and its derived scripts, Latin letters, and ASCII digits. Small spaces between them help the readability.

This feature has been implemented in several applications. iOS 15 and macOS 13 Ventura enabled this feature by default. Japanese TeX and Word processors such as Microsoft Word, OpenOffice, and CJK local applications have enabled this feature by default for decades.

For browsers, the current CSS Working Draft defines the `text-autospace` property. The Chromium team has expressed their interest in implementing this property, and the WebKit team as well.

A character property for auto spacing helps CSS to define this feature, helps interoperability between browsers, and even across non-browser applications and platforms.

The character classes started as a simple derived property of a few existing properties. But the discussion at W3C is going into more detailed rules, by adding or removing specific Unicode blocks, Script Extensions, and even specific code points. When the property is at such a level of complexity, Unicode defining it helps it be consistent with other properties and ensures new code points have correct values.

Property Values

Following is the list of values:

- **W:** Han ideographic characters and its derived characters.
- **N:** Code points that need inter-script spacing with W. It is possible to subdivide N in future.
- **O:** Other values. There is no inter-script spacing before or after O.

The value name W and N are after UAX#11 EAST ASIAN WIDTH. They aren't exactly the same but have similar concepts.

As a character property, it needs to define *some* values for all characters, extrapolating beyond traditional Japanese conventions.

Exact algorithm and code points are to be determined, but the current discussion on W is:

- It should include when the Script property is Han and most of its Han-derived scripts such as Hiragana, Katakana, etc.
- Whether to include all Han-derived and other ideographic scripts such as Hangul, Khitan, Nüshu, Tangut, Jurchen, Classical Yi, Egyptian Hieroglyphs, etc. or not is currently under discussion.
- Some characters whose Script is Common need to be included, such as U+30FC Katakana-Hiragana Prolonged Sound Mark, U+3006 Ideographic Closing Mark, and U+303C Masu Mark. An idea is to use Script_Extensions.
- A discussion to exclude Halfwidth forms is going on.
- An early discussion in Japan suggested (but needs more discussion):
    - Exclude Enclosed CJK Letters and Months. What to do with Enclosing diacritics is to be discussed.
    - Exclude Squared Katakana words.
    - Exclude Kangxi Radicals.
    - Exclude U+16FE3 Old Chinese Iteration Mark, U+16FF0 Vietnamese Alternate Reading Mark Ca, U+16FF1 Vietnamese Alternate Reading Mark Nhay.

The current discussion on N is:

- Not W.
- General Category is L, M, or Nd.
- A discussion to exclude Halfwidth forms is going on.
- A discussion to include some symbols and punctuations is going on, to support words such as “C#”. Punctuations have examples too (“!important”, “:lang()”, etc.) but they are more risky than Symbols, their side effects need to be more carefully examined. For example, the blood type “A-” needs spacing if followed by ideographic characters, but prefixes such as “e-” or “i-” don’t. The specification should probably say the algorithm is heuristic and isn’t perfect for all cases.

All other code points will be O, indicating they don’t insert a space at all.

Subdividing N

The current CSS Working Draft has two switches; `ideograph-alpha` for letters and `ideograph-numeric` for numerals. It also has `normal` to turn on both switches.

To support this distinction, the value N may need to be classified into more sub-values in future. However, this distinction will complicate the algorithm when symbols are involved, such as “C#,” and thus not included in the initial proposal.

Currently there are two future ideas to subdivide:

- Subdivide N to Nl/Nn/No (letters/numeric/others.) What to do with No needs discussions. One idea is to make it a wildcard N, meaning always insert spaces with W regardless which switch is on. Another idea is to make it opaque so that layout algorithms should look for next or previous characters.
- It may be useful to have different values for left and right. Japanese TeX has this capability, but part of the reason is from its use of a specially designed font metrics that isn’t commonly available. The need for this capability needs more discussions.

Algorithm

The CJK inter-script space should be inserted at:

- Between W and N.
- Between N and W.

Space

There are two ways to represent a space: a character space (by the insertion of physical code points, or in a glyph space (like kerning, adjusting the metrics of adjacent glyphs on the device).

It's not necessary for Unicode to define which representation applications should use, but it might be good to clarify that the algorithm should avoid inserting double spaces, such that glyph space shouldn't be inserted if there's character space.

When inserting character space, the CJK inter-script space is usually between 1/8em to 1/4em. Traditionally, JIS X4051 defines it to be 1/4em, while JLReq defines it as 1/8em, and CSS defines it as 1/8em as well. U+2009 Thin Space can be used for this purpose, as it's language dependent.

When inserting glyph space, it's easier to implement in layout engines than in fonts, because:

- The CJK inter-script space is inserted between different scripts, and often between different fonts. Currently OpenType can define features only within a font, and within a script.
- The French Typographical Rules for Punctuation are likely in the same font and script, but they have rules such as "before word" or "at end of sentence," which isn't easy to implement in fonts.