慶應義塾大学SFC研究所
Advanced Publishing Laboratory

日本語書記技術WG報告書

2019年3月31日

目　次

日本語書記技術WGの議論の概要

小林龍生

1　日本語書記技術WGの構成

WGの名称など

本WGは，当初“日本語組版技術WG”として発足したが，W3Cにおいて“JLreq Task Force”が新規発足したことを受け，混乱を避ける目的もあって“日本語書記技術WG”と改名した．

本WGは，2017年11月1日を第1回に，ほぼ毎月1回のペースで議論を深めてきた．

メンバーと役割分担

WGのメンバーとそれぞれの議論の主軸は，下記の通りである．

小林龍生
本WGのとりまとめを担当
児童向け雑誌編集者，符号化文字集合の国際標準化活動，W3Cにおける Japanese Language Task Force 議長などの経験を踏まえて，議論のとりまとめを行っている．

小林　敏
W3Cの『日本語組版処理の要件』の主執筆者，JIS X 4051（第3次規格）の原案作成委員会では幹事
活字組版，写真植字，コンピュータ組版，DTPによる書籍の製作に携わる．日本語のルビや縦書きなどの要件をEPUB3やCSSに反映させる過程で『日本語組版処理の要件』が果たした役割は看過できないものがあるが，事後的な反省点として，『日本語組版処理の要件』が関係者の間である種の「金科玉条」化し，すべての要件が必須のものと捉えられ，そのことが逆に，その後に，規格化や実装の妨げとなる局面が見られるようになっている．
本WGにおいては，各要件のデジタル通信環境への継承の必要性につき，是々非々で判断する役割を果たした．

村田　真
慶應義塾大学の特任教授，APLの研究員

APL においては，アクセシビリティ WG の主査を務めており，当初は，アクセシビリティ WG と本 WG とのリエイゾンとして参加を求めた．しかし，議論を深める過程で，単なるリエイゾンオフィサーとしての役割を越えて，主体的に議論に係わっている．アクセシビリティへの視点が，一般的な日本語書記技術論の議論の幅を拡げる上でも不可欠であることの発見は，今期の大きな収穫の一つと言えよう．

木田泰夫
元 Apple Inc. シニアソフトウェアマネージャー
国際化ソフトウェア，特に日本語の扱いに長い経験を持つ．本 WG においては，最も，ビジネス，実装に近い位置に立つ．
EPUB3 の開発においては，アップルコンピュータにおいて，WebKit の実装を進める過程で，多くの重要なインプットを行った．
グローバルな視点から日本語書記技術を相対化して見る視点が期待されている．

Florian Rivoal
W3C CSS ワーキンググループ，AC ボードメンバー
過去にシャープの XMDF の開発に係わった経歴も持ち，現在は，Elika らと並び，W3C の CSS ワーキンググループの主要メンバーとして活動している．また，AC ボードメンバーにも選任され，W3C 全体のマネージメント，ポリティクスへの影響力も大きくなっている．
本 WG においては，リエイゾンとして，本 WG の議論の成果を W3C にインプットする役割のみならず，木田同様にグローバルな視点と抽象化された機能要件と W3C の規格を中心とするさまざまな技術規格との対応関係を検討する役割を担っている．

小林潤平
大日本印刷株式会社
読みやすい（誤読が少なく，読み速度が速い）文章の表示方法を，視線トラッカーを含む測定技術を用いて実装的な研究を行っている．
本 WG においては，従来，なんとなく〈読みやすさ〉に寄与するとされてきた日本語の組版要件を，実証研究に基づき，批判する役割が期待されている．

※橋口侯之介氏
古書肆誠心堂店主
氏には，客分として討議に加わっていただいている．
『和本入門』を初めとする複数の著書があり，日本における和本研究の第一人者．特に，氏の書籍の「明治 20 年問題」は，明治初期における書籍業界の大きな変動を，技術面，流通面双方から捉えた卓見．冊子本と巻子本，物之本と草双紙など，本 WG の議論に歴史的な視線を加える上で，氏の果たしてきた役割は大きい．

2 議論の概要

本 WG での議論は，活発かつ多岐にわたったが，大きく分けると，3 層に分けることができよう．

1 〈本〉とはどのようなものか（定義ではなく機能論的属性）
2 読みやすさとはどのようなものか（小林潤平理論による組版機能の評価）
3 リフロー可能文書における具体的な実現要件（ルビと行組版）アクセシビリティからの視座を踏まえて

2.1 〈本〉とはどのようなものか（第 1 層）

〈本〉の概念が揺らいでいる．電子書籍の出現と普及により，従来自明のものとして受けられてきた本の概念が揺らいでいる．

のみならず，さらに広い概念である〈書記〉の概念そのものが，Web 文書等のデジタル通信技術の進展とともに，ゆらいでいる．

〈本〉と〈書記〉という表記

本報告書では，〈本〉という表記と〈書記〉という表記を，一般的な本，書記ということばとは，やや異なった意味合いで用いている．

日本語では，「本」「書物」「書籍」といった類義語が複数存在する．後述するが，これらの類義語の機能論的概念分析そのものが，本 WG の議論の大きな軸となっている．

〈本〉は，英語で言えば，“book”とか印欧語の語幹として多く現れる“biblio”といったことばに近いニュアンスで用いる．日本語の「書籍」「書物」を包含する．

本報告書では，煩瑣を避けるために，原則として，一括して〈本〉という用語をもちいる．山括弧で括っているが，囲われていない場合も，特に言及がない場合は，〈本〉と同一の言葉として読み替えていただきたい．

〈書記〉は，英語の writing の訳語であるが，英語の document に対応する語として用いる場合もある．「書類」「文書」といったことばからこぼれ落ちるニュアンスを慮り，敢えて違和感のあることばを用いている．山括弧で囲われていない場合も，特に言及がない限り，意味的な相違はない．

しかし，振り返ってみると，本の概念も文書・書類の概念も，歴史的にも常に一定であったわけではない．

本の概念は，その出現のときから，時代と文化の変遷に呼応して変貌し続けてきた．未来の本や書記のあり方を想像するためには，その補助線としての過去の書物や書籍，書類や文書の変貌を振り返ることが欠かせない．

書記とは：機能論的視座から

大英図書館でインド文字の専門家だった Albertine Gaur は，その著書“*A History of Writing*”の劈頭で，“All writing is information storage”と記している．日本語学の泰斗，小松英雄は『日本語書記史原論』で，この Gaur の言葉をうけて「書記とは，文字で書かれた情報の記録である」と記している．

小林（龍）が提唱している日本語書記技術論も，この二人の先人の言葉から出発する．

記録された情報は読まれなければならない．読まれることのない情報の記録に意味はない．

読まれることが伴って初めて，情報の記録は完結する．そして，読まれた情報は，なにがしかの相貌の変化を伴って記録される．この書くことと読むことの連鎖にこそ，〈書記〉〈本〉のかけがえのない本質がある．

そして，〈本〉の持つべき機能性とは，すなわち，読むという行為を支援する機能でなければならない．

では，読む行為を支援する機能とは，どのようなものか．本 WG が解き明かそうとする問題圏は，まさにここにある．

本 WG は，第 1 期において，読む行為の対象を従来の本に限定することはしなかった．第 1 期においては，議論を深めることが出来なかったが，第 2 期においては，本が担ってきた機能的役割を書記との機能的差分で相対化して議論を深めたい．

本・書物・書籍

現在（平成時代）の日本の社会において，この三つのことばは，同義性を持ちながら，微妙な差異を内包している．それらのことばの差異に注目することで，〈本〉が持つ機能的属性の外縁と内核が見えてくるかもしれない．以下，断片的な言説（ディスクール）を並べる．

- 電子書籍，電子本とは言うが，電子書物とは言わない
- 日本書籍出版協会（書協）であり，日本書物出版協会ではない
- 本好き，書物好きは言うが，書籍好きには違和感がある（？）
- 書物の価値と書籍の価格

これらの言説から以下のようなことが透けて見えるように思われる．

- 書物という言葉には，装幀や手触りといった物理的な属性がつきまとっている
- 流通や販売などの商取引の場には，書物よりも書籍という言葉の方が馴染みやすい
- 本という言葉は，書物や書籍という言葉に比して，その指し示す範囲が広く，また，コノテーション的にニュートラルである

物之本と草紙

橋口侯之介氏によると，江戸時代「仏書・漢籍など教養書は物之本といい，娯楽性の高いものは草紙あるいは双紙（合わせて草双紙とも

A History of Writing　Albertine Gaur “*A History of Writing*”, Abbeville Pr, 1992

日本語書記史原論　小松英雄著“日本語書記史原論”，笠間書院，2006 年

日本語書記技術論の勧め　「日本語書記技術論の勧め」（コンピュータソフトウェア）　https://www.jstage.jst.go.jp/article/jssst/33/3/33_3_66/_article/-char/ja/

いう）とよばれていた」（“和本入門” p. 67）

物之本という言葉が，現代の書物という言葉と深い関わりがあることをうかがわせる．

和本入門　橋口侯之介著“和本入門　千年生きる書物の世界”，平凡社，2005 年（平凡社ライブラリー，2011 年）

本の価値：今後の議論

一般に書物とは呼ばれない（書物という概念からは排除される）書籍の位置づけの必要性．

マンガの単行本は，流通上，雑誌として分類される．例えば，手塚治虫の『火の鳥』の豪華愛蔵版（ハードカバー）は，書物とは呼ばないか．マンガ一般を草紙の末裔と捉えることが出来るし，現在の和書流通市場においては，草紙にも高価格で取引されるものが多くある．「物之本の文化」「草双紙の文化」という言葉を措定すると，イーグルトン的な意味での文化相対主義的視点（大文字の Culture ではなく，互いに価値的軽重を伴わない複数の cultures）を措定した議論が可能となる．

冊子本と巻子本

この区別は，洋の東西に共にある．

本の機能性を議論する上で，単に冊子本のみならず巻子本も視野に入れることは必須のことのように思われる．

巻子本を視野に入れることにより，以下のような問題圏が浮かび上がってくる．

- 一般の Web 文書は，巻子本の嫡流と捉えられる
- 冊子本の巻子本に対する優位性を検討することで，一般の Web 文書の課題が見えてくる
- 大きな問題として，ページと行数指定による安定した参照・引用の問題がある

巻子本　巻子本（特に日本の絵巻物）のグラマトロジーの解明による電子漫画技法への適用という問題もある．（小林（龍））

装幀・判型の問題

野村悠理が『書物と製本術　ルリユール／綴じの文化史』で出版学会奨励賞を受賞した．このような地味ではあるが貴重な研究が脚光を浴びることは，御同慶の至りである．このような研究の蓄積により，従来の物之本（グローバルな意味で）が持ち，電子書籍に欠落する機能属性がどのようなものであるかが，解き明かされることを期待したい．

書物と製本術　野村悠里著“書物と製本術　ルリユール／綴じの文化史”，みすず書房，2017 年

グーテンベルクの 42 行聖書は，二つ折本で，非常に大型だった（京都外大にある一葉の大きさは 38.9× 28.3 cm）．

一方，アルダスは，多くの八つ折本を制作した．聖書を一般民衆が読む習慣が拡がることには，グーテンベルクというよりもアルダスの八つ折本の普及の寄与が大きかったのではないかと思われる．ルターによる宗教改革の源流をグーテンベルクに求めることができるのであれば，アルダスは，イエスに対するパウロのような役回りを演じたと言えよう．

判型の相違は，可搬性の多寡という物理的特性からも論じることが出来るが，例えば，ハードカバー A5 判の学術書，四六判の文芸書，

判型　読者は判型を見て，ある程度の内容を判断できる慣習が定着している．そこには，これまでの累積してきた慣行，本の扱いやすさ，材料の制約，書店店頭での棚のサイズなどに規制された，内容と判型についてのある種のゆるいコード（規則）があるといえよう．そうしたことを作る側は意識し，内容に応じた判型が選ばれている．したがって，判型は，ある種の選択された表現ともいえるかもしれない．こうした問題は，判型だけではなく，ハードカバーとソフトカバーあるいは，糸かがりと無線綴じの選択などにもあり，それらは機能性の問題だけでなく，ある種の選択された表現であると，考えられないか．議論する必要があろう．（小林（敏））

この指摘は，重要である．情報（ビット）とは，複数の可能性から 1 つを選び取ることと考えることが出来る．「判型を選ぶ」「糸かがりと無線綴じ選択」などの言葉は，その選択そのものが本が担う情報の一部であることを雄弁に示している．（小林（龍））

新書判２段組の娯楽小説というように，内容と判型の関係についても忘れてはならない．

本と書類の機能論的な相違（本にあって書類にない機能）については，次項以降で詳述する．

安定した参照引用の可能性と書記群の線形化としての本

上記の断片的な議論から，本が持つべき普遍的な２つの機能が透けて見えてくる．

1 つは，安定した参照引用可能性の機能である．

さらに，この機能は，

・本が持つ情報（コンテンツ）の安定性
・本が持つ情報（コンテンツ）の特定部分の指示可能性（ページ＋行／章＋節など）

の細分化することが可能である．

もう 1 つは，製本や EPUB における spine 情報などに象徴される，断片的な書記（フラグメント）をある特定の視点／意図から排列する機能である．

後者についてシリアライズの主体を書き手側に求めるか，読み手側に求めるか，というかつてハイパーテキスト論として盛んに議論された問題圏が係わる．

ハイパーテキストとしての Web ドキュメント

以下は，小林（龍）の『EPUB 戦記』の一部である．

中野さんと「電子書籍」の編集・制作に没頭していた 1990 年代初頭，パーソナルコンピュータ普及の初期段階にあって，ハイパーテキスト論が世上を賑わした．ハイパーテキストという概念の歴史をどのように捉えるかについてもさまざまな考え方がある．ヴァネヴァー・ブッシュの Memex，テッド・ネルソンのザナドゥ等等．この時期，技術分野からも人文科学分野からも夢と共にネットワーク化された電子テキストのさまざまな可能性が議論された．ハイパーテキストの技術的基盤は，ティム・バーナーズ・リーの HTML によって，いとも簡単に実現した．しかし，その爆発的な普及と共に，ハイパーテキストについての本質的な議論は，潮が引くように議論の前線から消えていった．

EPUB を初めとする，電子書籍のフォーマットに係わる議論は，改めて，本とは何か，という普段は意識することすらない問いを改めて自覚させることになった．製本と spine 情報との関係，冊子本と巻子本の関係，という二つの軸で，改めて，これからも考え続けていかなくてはならない，本の本質に係わる問題圏を，改めて整理しておこう．

［ページ概念からの解放と課題］

「本」には，冊子形態のものだけではなく，巻子形態のものも存在する．冊子形態の巻子形態に対する大きなメリットの 1 つは，頁概念が明確である点にある．なによりも，ページを単位とするランダム

製本の機能論的な問題圏　この問題圏については，小林（龍）が EPUB における spine 情報との関係で，すでに論じたことがある．

“Web 文書と電子書籍　製本機能としての spine 情報”

https://www.jstage.jst.go.jp/article/johokanri/55/11/55_802/_html/-char/ja

小林（龍）のこの論考は，特に，本と書類の決定的な相違として，製本によるシリアライズの問題を取り上げている．

製本の機能と spine 情報　spine が Web に存在せず，EPUB で導入されていることには村田真などからの強い批判がある．本にとって決定的に重要なものなら Web に入れるべきであると彼らは主張する．EPUB 制定開始時点では EPUB に spine を入れる以外の選択肢はなかっただろうが，spine が EPUB にしかないことは現実にさまざまの問題を引き起こしてきた．本報告の“EPUB は Web ではない”および“Is EPUB part of the web?”を参照．（小林（龍））

安定した参照・引用　ページや行数で参照が可能（or 便利）であることよりも前に，それが安定して永続的であることがまず重要だと思っている．安定して存在する，アクセス可能でありさえすれば、本一冊丸のまま参照することで参照の問題を解決することができる．（木田）

EPUB 戦記　小林龍生著“EPUB 戦記　電子書籍の国際標準化バトル”，慶應義塾大学出版会，2016 年

アクセスが可能であると同時に，ページ，行を指定することにより，特定個所を正確に明示することが可能になる．この機能が，特に，学術書における引用を軸とするエコシステムの形成に大いに寄与したことは疑いを得ない．一方，スクロールを主体とするWebドキュメントは，巻子本形式による書記閲覧形式の正統的な嫡子であると考えられる．しかし，画像とテキストの高度な融合や，閲覧姿勢に対する配慮といった，日本の絵巻物が育んできた高度で洗練された技法，文化が，現在のWebドキュメントの制作方法に十全に生かされているとは考えがたい．縦組横スクロールへの対応を初めとして，Webドキュメントに対する機能要件として検討すべき課題は多々残されている．一方，スクロール可能なドキュメントの特定個所を指定する技術的な方法は，特に紙の冊子形式におけるページ行指定方式と比して，はなはだ貧弱である点も否めない．EPUBを初めとする，電子書籍フォーマットは，あたかも所与の条件として，ページ概念を前提とした設計がなされてきたが，電子書籍フォーマットにおいても，ページ概念を相対化し，巻子形式に対応する概念の導入も積極的に検討すべきではないか．

［線形化過程としての読文行為］

読文行為という言葉は一般にはあまり使われないように思われる．ぼく自身は，読文行為を，Readingすなわち，「読むこと」の意味に使っている．読書という言葉には，書籍や書物を読むという含意があるが，読むこと（Reading）の対象は，必ずしも，書籍や書物だけではない．読文行為という言葉に違和感を覚えるようであれば，単に「読むこと」と読み替えてもいっこうにかまわない．

「電子聖書」では，付録に５インチのフロッピーディスクを付け，荒井献の東京大学における最終講義録「プロローグとしてのエピローグ」をハイパーテキスト化した電子情報を，収録した．このハイパーテキストでは，荒井献が引用した文章と，引用された元著者とのハイパーリンクを張り，元著者別のインデックスを設けて一覧できる機能を実装した．その結果，例えば，荒井が荒井自身の文脈の中で引用しているショットロフやシュスラー・フィロレンツァなどの文章から，元著者のフェミニスト的視点を鮮明に読み取ることが可能となった．紙の本においても，索引の制作が本文の著述とは独立して評価しうる優れて知的な創造行為であることは夙に論じられている．ハイパーテキスト構造によるリンク付けの作業が，多様な文脈による読文行為の可能性を拡げることは，将来の電子書籍の在り方を考える上で，もっと重視されてもいいように思われる．その際，かつてのハイパーテキスト論ブームの折に盛んに論じられ，ぼく自身もさまざまに思考を巡らせたことではあるが，ハイパーテキストのまさに蜘蛛の巣のようなリンク構造を選択的に辿ることは，読者の側から見ると，読文行為の時間軸に沿った線形化と捉えることが出来る．この線形化という行為は，シャノン的な情報エントロピー論に立てば，エントロピーを減少させる行為と言い換えることが出来るだろう．それを，読者が自分自身の物語を紡ぎ出す行為だとさらに言い換えてもよい．すなわち，ハイパーテキストをある意図の元に順を追って読むという行為は，読者

の側が，自分自身の手で，新たな spine 情報を構成し，一冊の新たな本を編む，という行為に他ならない．

［創造的営為としての読文行為］

EPUB の spine 情報をヒントとして，紙の本と電子書籍と Web ドキュメントの相違について考えてきた．もう一度，要約しよう．本を本たらしめている基本的な機能は，ある要素文書を特定の意図で選択し，一定の順序に排列することにより，ひとまとまりのメッセージを表明することにある．紙の本は，この機能を物理的な製本という行為によって著者の側に固定化する．EPUB における spine 情報は，紙の本における製本機能に相当する．一方，電子化ドキュメント，なかでも，Web ドキュメントは，その多様なリンク付け機能により，読者の側に，多様な可能性の中から，ある特定の意図に基づく読みの順序を選び取る自由を残している．さまざまなリンク構造の中から，ある一定の意図で選び取られた読みの順序は，それ自体として，すぐれて創造的な成果物であり，思想の表明であり得る．将来の電子書籍とは，多様でしばしば混沌とした Web ドキュメントの連鎖の中から，ある意図で切り取られ選び取られたドキュメントの，線形化された読みの軌跡となるのではないか．

2.2 版面設計に関する議論（第 2 層）

本 WG の出発点となった JLreq は，“第 2 章　日本語組版の基本”の多くを版面設計に係わる議論に割いている．この版面設計の議論は，欧米のページネーションの流儀と異なる点が多くあり，実装上もさまざまな困難な点が指摘されている．以下，「版面設計の呪縛からの解放」をキーワードとして，本 WG での議論をまとめる．

版面設計の呪縛からの解放

日本語組版における版面設計は，組版の大枠を決めるだけではなく，各ページに文字を配置する場合にも大きく影響してしている．

また，版面サイズの決め方も欧文組版と異なり，字詰め方向の版面サイズは，1 行に配置する文字サイズと字数で決定し，行送り方向のサイズは，文字サイズ，行間及び行数で決定する．これは，配置する文字は正方形の文字の外枠を持っており，この文字をベタ組で配置し，段落の最終行を除外し，各行の行頭及び行末をそろえる（行長をそろえる），さらには，行の位置はページの中で固定し，行送り方向の各ページの版面サイズもそろえる，という考え方からきている．（特に活字組版では，版面サイズを一定にすることは，作業性からも，強く要求されていた事項でもあった．）

字詰め方向や行送り方向への文字の移動距離がそろっていることで，視線の移動が常に一定となり《読みやすい》，さらに，行長や版面サイズがそろっていることで，ページあるいは見開きページにおいて，整然とし，バランスがよいと考えられていたことによろう．

しかし，実際に文字を読んでいく場合，1 字単位ではなく，ある程度のまとまり（単語あるいは文節単位）で読んでいく．となれば，段

落内で行を折り返す場合，文字単位で折り返す位置を決めるのではなく，単語あるいは文節単位で折り返す方が読みやすいと考えられるし，小林潤平などの実証研究でも，そのことが示されている．（なお，単語あるいは文節単位で折り返す方法でも，行頭・行末の位置をそろえることは可能であるが，字間のバラツキが大きくなり，行頭はそろえるが，行末はなりゆきとする“行頭そろえ”を選択するしかないであろう．

また，段落間の行間を少し拡げることで，段落の認識がより明確になる，ということもいわれている．この場合，段落間を 1 行アキとすれば行送り方向のサイズはそろえられるが，空き過ぎとなり，1 行アキでないアキにすれば，行送り方向の版面サイズはそろわなくなる．そろわなくても，その方が《読みやすい》のであれば，版面サイズをそろえることよりは，《読みやすさ》が優先され，ある程度の版面サイズが不ぞろいになることは許容される．

となると，前述した行長や版面サイズをそろえるという点で，《読みやすさ》への機能的な貢献度が有意に見られないならば，特に，見開きを考慮しなくてよいモニタでの表示であれば，前述した日本語組版の版面設計の方法は，再検討される必要があるといえよう．

つまり，版面サイズは，字詰め方向及び行送り方向の最大サイズを決めればよいこととなり，欧文組版の版面サイズの決め方と同じ方法も選択できることになる．

今後こうした点での議論をしていく必要があるといえよう．（この項，小林（敏））

文節と読みやすさ　日本語文章における視点移動や認知の単位は，文節とされている．したがって，文節のまとまりを最大限に保持する設計が，読みやすいレイアウトの実現に寄与する．例えば，折り返しの位置を文節間に設定したレイアウトによって，視点移動が効率化され，読み速度の向上をもたらした結果が報告されている．（小林（潤））

段落間のアキ　最近の実験結果では，段落のまとまりや段落先頭センテンスを意識した読み方が，文章内容の理解につながる可能性が示唆されている．その結果をふまえると，段落間の行間を広げる方式や，段落先頭センテンスを字下げする方式は，段落の認識や段落先頭のセンテンスを他と識別しやすくする点で，有効な設計方針であると言える．（小林（潤））

行　長

ひとつの目安として，以下のことがいえる．

行長に関しては，その最適な長さについて古くから議論されてきた．様々な研究の結果，行長については，長過ぎると「行末から行頭に至る正確な視線移動が困難になる」一方で，短過ぎると「一目で受容可能な情報量に満たず充分な読み能力を発揮できない」というトレードオフの関係が存在し，それらの妥協点が最適とされてきた．最近の日本語文章に関する研究では，行長は長いほど速く読めるが一行 20 文字付近でほぼ上限に至るとともに，読者がもっとも読みやすいと感じた行長は一行 25 文字前後であったことから，最適な行長は 20〜30 文字程度と結論付けられている．（この項，小林（潤））

縦書きと横書き　縦書きと横書きの読みについては，慣れの影響が大きいことが指摘されている．古くより，縦書きと横書きどちらにも慣れている場合には，読み速度や理解度は同等との結果が報告されてきた．

半世紀前には，縦書きの文章が多かった．1960 年に報告された永野らの実験では，縦書きの方が横書きよりも速く読まれており，その結果について「現状（1960 年当時）における読書環境では，縦書きのものが横書きよりもはるかに多いことから，慣れの度合いが反映されたもの」と推察されている．

一方，現代は横書きの文章が多い．2018 年に実施された大学生 12 名の小規模な予備実験でも，縦書きの方が速い学生はゼロであり，横書きの方が縦書きよりも平均 10％程度速く読まれていた．（小林（潤））

縦組機能の維持

EPUB3 における日本語組版機能実現の大きな目標が，縦組の実現にあったことは言を俟たない．

EPUB に準拠した電子書籍の多くが，EPUB3 で実現された縦組機能を用いている．一方，一般の Web 文書での縦組機能の利用頻度は必ずしも高くはない．

小林（潤）の研究においても，縦組が横組に比して，《読みやすさ》と言う点では明確に劣っていることが自明のこととなっている．

電子化文書において将来的にも縦組機能を維持していくべきか否か，という議論も，その（グローバルな）維持コストとのトレードオフを踏まえた上で，議論すべき時期が早晩訪れることと思われる．

縦書きと横書きの読みやすさ　横組文章の読みに慣れた人が多い状況では，横組の方が読みやすい人の割合が多いことを意味する．一方で，縦組文章のみに慣れた人は，縦組の方が読みやすいのも事実である．

人体構造の点から検討すると，横組の方が読みやすい可能性が高いと思われる．人間の場合，頭を動かさずに目だけ動かして見える範囲は，上下よりも左右に広い．すなわち，眼球運動的に無理なく読める範囲は，横組の方が広いことを意味する．（小林（潤））

版面設計の見直しの必要性

日本語組版における版面設計の要諦が，行位置の固定化にあり，行位置の固定化の《読みやすさ》への機能的な貢献度が有意に見られないならば，日本的版面設計の原則も，先に挙げた縦組と同様，そのグローバルな維持コストとの連関で見直すことが必要になる時期が来るかもしれない．

柱とノンブル

柱とノンブルは，機能論的には，少しくその役割が異なっている．

インデキシング機能①　辞書などで用いられるサムインデックスも含め，ある比較的大部な書物の中から，目的の個所を見つけ出し，該当個所に到達することを支援する機能．

この機能は，パラパラとページをめくれる，という冊子体書物（コーデックス型）特有の優れた物理特性に拠るものと考えることが出来よう．

インデキシング機能②：参照と引用の安定性　参照や引用をする際，従来の書物では，ページ数と行数によって特定の個所を指示することが一般的に行われてきた．

この機能が有効であるためには，版面が固定されており，安定していることが必須である．

書物が書物たる所以の一つとして，引用や参照の安定性が挙げられる．ある書物のページ数が固定化されており，特定のページと行数を指示することで，目指す個所に確実に到達できることは，書物の持つ最も重要な機能の一つである．

また，聖書の章節による指示も同じ機能を持つが，この点については，改めて論じる．

2.3　行組版に関する議論（第３層）

今期本 WG では，主として，電子書籍や Web 上の書記を〈リフロー可能〉という観点で捉え直して，議論を深めてきた．特に，W3C に対するアウトプットとしては，約物の詰め処理と改行時の禁則文字処理，及び，ルビの処理に絞って議論した．

リフロー可能な文書処理

第 1 層，第 2 層の議論とも関連するが，電子書籍や Web 上の書記においては，ページやウィンドウ（ペイン）のサイズが可変であることが前提となる．本 WG では，このようなページサイズ可能な書記を〈リフロー可能な書記〉という観点で捉え直す作業を行っている．

例えば，従来の JLreq に対する批判の一つに，行頭行末の約物処理（いわゆる禁則処理）についてのものがある．JLreq の要件を厳密に

行の調整処理の問題点　特にアキ処理を行った場合，字間を空けることになるが，少ない字間で処理すると，字間がかなり空いてしまう場合もでてくる．これは，ベタ組に慣れた読者にとっては，あまり読みやすいとはいえない．（小林（敏））

適用すると，特に行長が短い場合などに，行の調整処理で無理が生じるとの指摘がある．

リフロー可能な文書における行組版処理

第 1 層の議論とも関連するが，従来の本の有意な特徴の一つに，ページが固定化されていることがある．そのことによる機能的優位性の一つとして，引用，参照の安定性があることは，すでに述べた．

また，《読みやすさ》という観点からは，行長の変化に伴い，適用すべき禁則処理に変化が伴うべきである，との議論についても言及した．

こうした議論の中で，ページレイアウトが固定された書物や電子書籍と，いわゆるリフロー可能な電子書籍の，機能論的相違についても，議論が及んだ．

書物とはどのようなものか，という問題意識とリフロー可能な書籍との関係については，すでに述べた（まだ書いていない）．

リフロー可能なドキュメントに関する組版要件案については，本報告の“リフロー可能なドキュメント環境とは”で述べる．

具体的なアウトプットとしては，従来の JLreq における文字クラスを簡略化し，実装をより一層容易にすることを目指した，簡易版組版規則の要件をまとめた（本報告書に添付した“簡便な行組版ルール(案)”）．

また，具体的な要件定義の作業と雁行して，小林（潤）が，その実証研究の成果を基に，それぞれの要件の《読みやすさ》への寄与の多寡について検討を加えた．（本報告書の“読み効率を高める日本語電子リーダー設計の試み”参照）．

さらに，活版印刷以前のいわゆる和本の歴史の中にも，文節区切りを視覚的に明らかにし，《読みやすさ》に資すると思われる試みが複数なされてきたことも指摘されている．

なお，リフロー可能な文書における行組版規則（特に禁則処理）は，行長の長短によって変化せざるを得ない．リフロー可能な文書における可変的な版面設計のあり方については，第 2 期 APL における本 WG の主要なテーマとして，議論を深めていきたい．

分かち書き

ディスレクシアの議論から，分かち書きの有効性が指摘されている（本報告の“組版についてのアクセシビリティ要件”参照）．

分かち書きの方法，単位については，年齢，リテラシー（本来の意味での読み書き能力）の習熟度によっても，差異が認められる．

分かち書きの処理方法については，本 WG としては，W3C のエキスパート（Richard Ishida，Elika，Florian）らとの議論を踏まえて，一定の方向性を打ち出した．

なお，分かち書きした場合には，段落の処理方法としては，上揃え

読みの視覚心理　日本語書記法を検討する上では，文節単位での視点移動のしやすさが，読みやすさのひとつの指標になる．

人間の視野は，中心部分でもっとも解像力に優れ，周辺部では低下するという特徴をもつ．そして，解像力の高い視野中心部はとても狭く，視野中心から約 2.5 度離れると，解像力は半分に低下する．細かな文字を識別できるのは，視野中心部に限られる．日本語の場合，その広さは約 10〜12 文字と報告されている．したがって，視野中心部に収まらない長さの文字列を読むためには，次々と視点を移動していく必要がある．

視野中心部で文字を認識している注視状態は「停留」と呼ばれ，次の停留点への視点移動は「サッカード」と呼ばれる．また，もっとも短時間で単語を認知できる停留場所は「最適停留位置」と呼ばれる．英語やフランス語などのスペースを有する書記言語では，最適停留位置は各単語の中心付近であり，最適停留位置から外れた場所に停留すると，同一単語内で再び停留が発生しやすくなる．したがって，読み効率の向上で重要なのは，最適停留位置への的確な視点移動である．

読書中は視点移動が繰り返される．停留中には，視野中心部で文字を識別すると同時に，視野周辺部で次の停留先の選定を行う．英語のような単語をスペースで区切りながら表記する言語では，スペースによる単語間の視覚的な境界情報が，視点移動に対して重要な役割を担うとされる．そして，単語間のスペースを除いた場合には，読み速度は 30〜50％低下することが報告されている．

日本語文章は，表意文字である漢字に，音節文字である仮名字を混ぜて表記する．漢字は視覚的に目立つ傾向があり，仮名字よりも，視線を誘引する傾向が報告されている．一方で，日本語文章の意味的な最小単位は「文節」と呼ばれ，熟練した日本語の読者は，文節単位で停留しながら読む傾向が報告されている．ここで重要な点は，日本語文章では「視線を誘引する漢字の出現場所」と「文節のまとまり」が一致しないことである．古くより，文節の間にスペースを挿入した場合の効果が検証されてきたが，文節単位の視線移動を誘引する効果をもたないことが報告されてきた．現代においても，日本語文章はスペースを用いずに表記される．

しかし，現代の日本語書記法が完璧なわけではない．日本語文章における平均停留時間および平均サッカード距離は，それぞれ約 0.25 秒および約 5 文字である．もし

にする方法と，行頭行末そろえにする方法がある（例は，本報告書の“簡便な行組版ルール（案）”に例を掲げてある）．

行組版処理：文節区切り

ここでいう文節区切りとは，段落内での行を折り返す際に，文節（または単語）を単位して折り返す処理法である．この文節区切りの内部表現についての議論に多くの時間を費やした（本報告の“CSS space expansion”参照）．

ルビ：アクセシビリティとの連関

本 WG は，その活動の初期から，アクセシビリティ WG 主査の村田真に議論への参画を求め，常にアクセシビリティとの関連を視野に入れて議論してきた．

ルビについては，特にディスレクシアへの対応を主軸に，議論を行った（本報告書の“組版についてのアクセシビリティ要件”参照）．

W3C 活動との連関

JLreq 第 3 章の議論は，多岐にわたる．また，電子書籍，Web 文書双方における機能論的観点からの必要性，実装の難易度についての言及がなく，標準規則化についても実装についても，そのプライオリティ付けの議論が進んでいない．

この点については，折に触れて，Richard Ishida が言及しており，本 WG の全身に当たる JLreq WG 発足時に設定された問題圏でもあった．

一方，W3C における行組版の議論は，主として，CSS WG を中心に行われており，日本語などの多言語における要件は，i10n WG における Richard Ishida を経由して持ち込まれる状況にある．

こうした中で，Florian により，標準化活動及び実装に向けた要件定義の議論は，ルビと行組版における約物処理に絞ることとした．

議論の成果は，小林（敏）により以下のドキュメントとして纏められている（本報告書に添付してある）．

・ルビの簡便な配置ルール（案）
・簡便な行組版ルール（案）

“ルビの簡便な配置ルール（案）”は，Florian によって英訳され（本報告書に添付してある“Rules for Simple Placement of Ruby”），W3C へのインプット素材として用いられている．

また，Florian により，CSS で実現すべきページネーション関連機能（本報告書の“CSS で実現すべき機能”参照）や，段組の新しい処理方法（本報告書の“CSS 段組の新処理方法の提案”参照）が議論された．

3 課題：第 2 期の議論に向けて

以下のような事項が主な課題となろう．

- リフロー可能書記の組版要件

スムーズに停留とサッカードを繰り返すことができれば，計算上は 1 分あたり 1200 文字の速さで読むことができる．一方，大学生による日本語の平均読み速度は，1 分あたり 650 文字であることが報告されている．計算上の速さと実際の速さには，大きな差がある．

この差が生じる要因は，最適な停留場所に的確にサッカードできないことに起因する，非効率な視点移動にあるとされる．すなわち，日本語文章においては，視覚的に目立つ漢字への視線誘引を前提としながらも，各文節への的確な目の動きを促すための新たな表示方式が必要とされていた．

例えば，小林（潤）らが提案した表示方式では，標準的な横書きレイアウトの文字列に対して，改行場所を文節の間に調節するとともに，文節単位で文字のベースラインを階段状に順次下げていく．この表示方式で読むと，読者の目の動きは自然と文節単位となり，効率よくスムーズに視点が移動するために読み速度が向上する．読み速度が向上しても，すべての文字を読んでおり，文章内容の理解度や読み心地は維持している．

わざとひらがなを使う，わざと漢字を使うなど，いままでも多くの方々が工夫してきた書記法も，濃い漢字に視線が誘引される性質をふまえたものであったと捉えることができる．（小林（潤））

分かち書きの有効性　日本語の読みにおいて，読者の視線は文節ごとに停留する傾向が報告されている［神部 94］．また，中條が提案している日本語文章の読みモデルでは，意味処理や視点移動は文節単位であるとされている［中條 99］．

古くより，文節の間にスペースを挿入した場合の効果が検証されてきた．仮名字のみで書かれた文章の場合，文節単位でのスペース挿入は，読みを向上させる効果をもつ．一方で，漢字と仮名字の混合文では，視覚的に目立つ漢字が 次の停留場所を決める手掛かりとなるために，文節単位でのスペース挿入は文節単位の視線移動を誘引する効果をもたないことが報告されてきた［御領 87，松田 01，藤木 02，Sai 07］．

分かち書きの処理方法　これを書いたドキュメントは，この報告書のどこかにありますか？

→村田さん，Florian さん

JLreq TF の Gap Analysis からのフィードバック

- Web 文書におけるページネーション機能の要件
- 和書➡活版書籍➡電子書籍：失われた情報／新たな付加情報の可能性

EPUB は Web ではない

村 田　真

1 はじめに

EPUB は，Web 技術に基づきながらも，Web ではない．Web 技術にない独自仕様（参考資料［1, 2］を参照）――とくに表示に関わる拡張――を EPUB ではいくつも導入している．いくつかの独自仕様は普及せずに消えていったが，広く使われているものもある．後述する spine は，広く使われている独自仕様の最たるものである．

W3C が出版を軽視していたころは，出版に対処するには EPUB 独自仕様しか方法はなかった．しかし，EPUB 独自仕様は大きな弊害をもたらしている．相互運用性の問題，コンテンツ制作コストの増加という問題である．

1.1 相互運用性の問題

EPUB の相互運用性が十分でない大きな理由は，表示に関わる独自仕様である．こうした独自仕様がなければ，Web 技術の相互運用性さえ確保すれば EPUB の相互運用性は保証されたはずであった．すなわち，どの Web ブラウでも表示できるものは，どの EPUB リーダでも表示されるに決まっていた．表示以外の独自機能は流通にかかわるものであり，表示に関する相互運用性には影響しない．

しかし，表示に関わる独自仕様の存在により，Web 技術の相互運用性を確保しても EPUB の相互運用性は保証されない．保証するためには，EPUB 出版物を用いて EPUB リーダをテストすることが必要である．このテストは，EPUB 3 が出来てから 7 年も経過したのにもかかわらず十分な成果を上げていない．EPUB 3 の相互運用性について海外出版社からの苦情が絶えないのがその証拠である（参考資料［3］を参照）．

備考：日本は電書協プロファイルによって EPUB3 の機能の多くを使わないことによって相互運用性問題を回避している．イノベーションを諦めて縦書き日本語文芸書とマンガに限定し，商業的には成功している．

1.2 コンテンツ制作コストの増加という問題

制作にとってもっとも望ましいシナリオは，Web コンテンツをパッケージ化すればそのまま EPUB になり，逆に EPUB パッケージをほどけばそのまま Web になることである．ここから外れれば外れるほど制作コストは増加する．

［1］EPUB features missing in OWP
https://twitter.com/i/moments/1057874066445545472

[2] EPUB features hampering unification with OWP
https://docs.google.com/document/d/1mrFNnGUDDSjTHkTlvSBtC2DMGSYGZA5THkDVE31UHE8/edit#heading=h.9i4o28k4goyb

［3］EPUB の嫌いなところ
https://twitter.com/i/moments/1057874066445545472

しかし，現状はそれとは反対であることがPublishing@W3Cの電話会議では指摘されている．長大なWebコンテンツとして作りこまれたものをEPUB化することは容易ではないらしい．その大変さを理由にEPUB化は出来ないと言うコンテンツ作成者もいる．

ここでも独自仕様が問題になっていることが疑われる．たとえば，Service WorkerをもちいてHTMLをまとめたWebコンテンツ（例えば［4］，［5］）を，spineを用いてEPUBコンテンツとしてまとめ直すのは再構成に近い大仕事ではないかと推定する．

2 SPINE

EPUBがもっともWeb技術と乖離したのはどこだろうか？　私はspineの導入だと思う．複数のHTML文書を並べる，見開きを二つのHTML文書で構成する，HTML文書の最後まで来たら次のHTML文書に移る，HTML文書に対して見開きのどちらに入るかを指定するなどは，すべてspineを導入したことから発生している．これらは，すべて前述のテストを必要とする．

しかし，Spineが電子書籍では本質的に重要だという指摘をする人は多い（例えば［6］）．商業的にも，複数の記事を集めて書籍として出版したいという要望は強い．

抜本的に問題を解決するためには，spineをHTMLに入れるべきだと思う．EPUBにspineを入れるのは弥縫策に過ぎない．制定中のWeb Publicationsも，default reading orderと呼び名を変えてspineを存続させているが，見直す必要がある（［7］を参照）．

備考：Publishing Working Groupはなんら検討を行うことなくdefault reading orderの見直し要求［7］を却下した．Publishing Working Groupが機能不全に陥っていることの分かりやすい証左である．

どうすればspineをHTMLに導入できるだろうか．HTMLの実権はもはやW3Cにはないので，WHATWGで議論しなければならない．幸い，WHATWGのissueとしてこの件は上がっている（［8］を参照）．

備考：EPUBマンガは，各ページを別のHTML文書にしており，それをspineでまとめている．しかし，これではページめくりを高速に行うことができないので，国内で広く用いられているマンガリーダはEPUBを別の形式に変換してから提供している．つまり，配布のときはEPUBでもWeb技術でもない．

3 CSSのページネーション機能の強化

EPUBがspineを入れ，各HTML文書に対して見開き中での位置指定などのpackage rendering properties（考資料［9］）を入れているのは理由がある．それは，CSSのページネーション機能が不十分だからである．

参考資料［10］は，CSSとHTMLが日本のページ区切り文書のレイ

［4］Resilient Web Design
https://resilientwebdesign.com/

［5］High Performance Browser Networking
https://hpbn.co/

［6］Web文書と電子書籍　製本機能としてのspine情報，小林龍生
https://doi.org/10.1241/johokanri.55.802

［7］Drop the default reading order from WP
https://github.com/w3c/wpub/issues/302

［8］Client side include feature for HTML
https://github.com/whatwg/html/issues/27912

［9］package rendering properties
https://www.w3.org/Submission/2017/SUBM-epub-packages-20170125/#sec-package-metadata-rendering

［10］となりのヤングジャンプでPortalを利用したスムーズなチャプター遷移
https://twitter.com/uskay/status/1062480777655476224

アウトに十分であるかどうかについて検討したものである．とくに，JLReq（W3C）と EPUB 3.0 日本語組版要望表（電書協）にある要件を検討し，どの要件がどの CSS 仕様でカバーされ，主要なブラウザでサポートされているかを示している．結論として，ページネーションに関係する機能が不足していることが指摘されている．

参考資料［3］の第四項も，CSS にないページネーションを EPUB リーダが作り出していることの問題を指摘している．

今後どのように CSS のページネーション機能を強化していくかについては，すでに Fantasai と Florian が検討中である．

4 その他の本質的な問題

他にも本質的な問題がいくつか存在する．これらは必ずしも spine のせいというわけではないが，まったく無関係というわけでもない．

4.1 URL がない

すべての資源を URL で参照可能なことは，Web アーキテクチャにおいてもっとも重要である．URL によるアクセスがどんなプログラムからも可能なことにより，Web はいろいろなサービスが連携して発展してきた．

しかし，EPUB にはそのような URL はない．もちろん，Web サーバに格納された EPUB に URL でアクセスすることは出来るが，電子書店から購入してローカル環境にある EPUB を URL で参照することはできない．したがって，EPUB 出版物の間にリンクを張ることもできない．参考資料［3］の第二項を参照．

4.2 オリジンがない

Web のセキュリティモデルではオリジンが非常に重要である．しかし，EPUB にオリジンはない．したがって，EPUB 中のスクリプトにはセキュリティついての問題がある．資料［3］の第三項を参照．

5 まとめと今後の課題

EPUB3 がいまの形になったのは，CSS のページネーションの弱さ，HTML にトランスクルージョン（spine を自然に実現できる機構）がないことから，仕方のないことであったかも知れない．しかし，問題を抱えていることは事実である．

EPUB 3 を多少手直ししても問題は解決しない（Publishing WG のやっていることはこれではないかと考えている）．EPUB 3 とは非互換になるが，以下の機能を備えたフォーマットが必要である．

1. HTML に spine を入れる．
2. CSS のページネーション機能を大幅に改善する．
3. EPUB のための URL を作る．この URL を dereference すると，ローカル環境に対応する電子書籍があればそれを参照する．
4. オリジンを提供する．これについては Google から TPAC の席

上で提案があった．［10］，［11］，［12］を参照．

これらすべてを達成するには５年以上の時間がかかるだろう．このような長期戦に APL がどう関わるべきだろうか？

［11］Signed HTTP Exchanges
https://developers.google.com/web/updates/2018/11/signed-exchanges

［12］Web Packaging
https://docs.google.com/presentation/d/1FMzlFX5NEYr0It7P_QqxdOgkRgNL4rY8nIoo53MV9GY/edit#slide=id.p

Is EPUB part of the web?
EPUB は Web の一部ではない？

Florian Rivoal
小林龍生訳

Web sites are made of HTML documents, JPEG or PNG images, SVG graphics, CSS stylesheets. . . JavaScript can be used for further functionality. All this can be said about eBooks in the EPUB format. At first, it seems that EPUB documents are one and the same as web documents, with the extra convenience of packaging, and some superficial differences.

I believe that the opposite is in fact true. It is the similarities that are superficial, and the differences profound.

Web サイトは，HTML 文書，JPEG または PNG 画像，SVG グラフィック，CSS スタイルシートなどで構成されている．また，JavaScript を用いて機能を拡張することもできる．これらの機能は，すべて EPUB フォーマットの電子書籍でも用いることができる．用いられる機能という意味では，EPUB 文書は Web 文書とまったく同じものであるように見える．違いは，パッケージ化によって利便性が向上したこと，いくつかの表層的な相違点があることでしかない．

私はその逆が事実であると確信している．類似点は表層的なものでしかなく，相違点こそが本質的なものである．

Divergent evolution
異なる方向への進化
Different compatibility models
互換性の考え方の相違

The way the Web approach compatibility and interoperability is very different from what is practiced in the EPUB world. The web, as its core, is built to be resilient. The concept of responsive design, where one document can be used to different reading environments by adapting to them is shows one aspect of this, but the web's resilience goes beyond being cross-device. Document authors accept the fact that their content will at times be displayed in browsers that lack some of the features they need. Browser developers in turn accept the fact that their browsers will have to render future documents that use features they do not have. Old documents work in new browsers, and old

browsers can display new documents, regardless of how many years of technical innovation and new features have happened in between. The web just adapts to the presence, or absence, of various features, but works regardless, without any versioning.

EPUB の世界での実際に採られている互換性と相互運用性へのアプローチは，Web のそれとは非常に異なっている．Web は（さまざまな使い方が可能な）コアとして，弾力的運用が可能なように構築されている．レスポンシブデザインのコンセプトは，1 つのドキュメントをさまざまな閲覧環境に適応させることで，どのような閲覧環境の下でも閲覧可能にすることにある．しかし，このコンセプトを支える Web 技術の弾力性はデバイス間の違いへの対応以上のものがある．（Web）文書の作者は彼らのコンテンツが時には彼らが求めている機能のいくつかを欠いているブラウザで表示されるという事実を受け入れる．一方，ブラウザ開発者は，自分のブラウザが持っていない機能を使用する将来のドキュメントを（たとえその機能が表現できないとしても）レンダリングしなければならなくなるという事実を受け入れる．その間に何年にもわたる技術革新と新機能が発生したかに関係なく，古い文書も新しいブラウザで動作し，古いブラウザでも新しいドキュメントも動作する．Web は単にさまざまな機能の有無に適応するだけで，バージョン管理なしで機能する．

This is a big constraint on the evolution of the web. Nothing that has worked can ever be broken, and even invalid content must be processed predictably, as what one old browser considers invalid is another browser’s new feature.

このことは Web の進化に対する大きな制約となる．古いブラウザの 1 つが無効と見なす機能が別のブラウザに新機能として実装されても，それまでの機能は何も壊れることはなく，そのブラウザでは無効な機能を含むコンテンツも予測どおりに処理する必要がある． このことは，統合された成果物としての Web が破綻なく継続的に成長してきたという非常に大きな強みの源となっている．これはまた，何らかのニーズに応えるために追加された機能が，追加された以降は，他の機能とともに，全体として，すべての人が使用できることを意味する．

This is also a huge source of strength, as this allows the web as a single artifact to continuously evolve without ever fracturing. This also means that every feature ever added to serve the needs of any constituency is added to the whole, and can from there on be used by everybody, in conjunction with all other features.

このことは，同時に，単一の成果物としての Web が破綻なく継続的に成長してきたという非常に大きな強みの源となっている．これはまた，何らかのニーズに応えるために追加された機能がそれ以降は，他の機能とともに，全体として，すべての人が使用できることを意味する．

Short-term solutions and long-term divergence
短期的な解決策と長期的な多様化

Because making all the actors of the web agree on a robust solution takes so long, people often find themselves wanting to do things that the platform does not allow yet. When that is the case on the web, authors use the features of the platform to create all manners of shims and polyfills within the documents to placate their needs. The results are often convoluted, but because they are built to work within existing browsers, they continue to obey the broader rules of the web. When such things turn out to be good ideas, browser vendors can “pave the cow paths”, and taking inspiration from what people have built, enhance the capabilities of the platform.

Web のすべての関係者が破綻を来さないような（ロバストな）ソリューションに同意するまでには長い時間がかかるため，多くの場合，プラットフォームではまだ許可されていないことをしたいと考える人がいる． そのような場合，作者はプラットフォームの既存の機能を組み合わせて，ドキュメント内にさまざまな方法で shim や plyfill を作成し自分のしたいことを実現する．その結果はしばしば複雑になるが，それらは既存のブラウザ内で機能するように構築されているため，Web のより広範な規則には従っている．
そのような方法が良いアイデアであることが分かると，ブラウザのベンダーは「牛の道を切り開く」（処理手順［処理方法・システム］を自動化する）ことができ，人々が構築したものからインスピレーションを得ることでプラットフォームの機能を強化することができる．

In contrast, because the publication world can exert some control both over the content and the reading software, when features that are not supported by the web platform are needed, they are added to specifications and eBook readers are tasked with implementing them. To the extent that the eBook readers cooperate, this leads to much faster time to market, but with a major downside: since this is done without the involvement of the broader web platform stakeholders, there is no guarantee that what has been added is something the web could adopt. Indeed, as Murata Makoto argues in “EPUBはWebではない” [1] a feature that is often considered essential to EPUB is something that the web lacks entirely: the spine, or the ability to build a single document out of several parts. As this feature was added without consideration for how it related to essential features of the web that EPUB does not have （such as the URL, or the origin）, it seems likely that it cannot be adopted by the web platform, or some related mechanism, if adopted, would have to be significantly different.

それとは対照的に，出版業界はコンテンツと閲覧ソフトウェアの両方をある程度制御できるため，Web プラットフォームでサポートされていない機能が必要な場合はそれらが仕様に追加され，その実装は eBook リーダーの役割となる． eBook リーダーの開発者が協力する限り，この方法は市場投入までの時間を大幅に短縮するが，

[1] EPUB は Web ではない　本報告書，村田真“EPUB は Web ではない”.

大きな欠点がある．この方法は，より広範な Web プラットフォームの利害関係者の関与なしに行われるため，追加されたものが Web に採用されない可能性がある．実際，村田真が「EPUB は Web ではない」[1] で主張しているように，EPUB に不可欠であると考えられる機能のいくつかは，Web には全く欠けているものである．例えば，スパイン（背表紙），または複数の部分から単一の文書を作成する機能．この機能は EPUB が持っていない Web の本質的な機能（URL や由来など）とどのように関連しているかを考慮せずに追加されたため，Web プラットフォームでは採用できないか，何らかの機能が類似するメカニズムが採用されたとしても，それは（EPUB のものとは）大きく異なるはずである．

[1] EPUB は Web ではない　本報告書，村田真“EPUB は Web ではない”．

EPUB and hypertext
EPUB とハイパーテキスト

This extra feature that EPUB has also leads us to consider what it lacks. And what it lacks is profound: again, as highlighted by Murata Makoto argues in “EPUBはWebではない”[1], EPUBs do not have a URL. Of course, they can contain links, and of course, they can be placed on a web server and downloaded when that URL is fetched. Any kind of file can be distributed via the web, and EPUB documents are not exception. However, web pages are not merely distributed via the web. They are part of the web. You can link from a web page to another web page, and the URL it exists at is an inherent part of how its identify and of how works, with deep implications over the security and execution model of the web.

If we take a step back from the technological details, and look at the deeper implications of hypertext, as Kobayashi Tatsuo does in “ハーパーテキスト論再考”[2], we can consider that the whole interconnected hypertext medium-the web-as a single entity. The word “web page”, thought it may seem unremarkable, is in fact quite interesting. Pages are constituent parts of books. Of what book are web pages constituent of? It seems as if the web as a whole is a single hypertext book. The reader carves out a particular view of it as they follow their path along hyperlinks, but no part of the web is truly separate from the rest.

[2] ハーパーテキスト論再考　小林龍生著『EPUB 戦記』（慶應義塾大学出版会，2016）の第 4 章「5　ハーパーテキスト論再考」（213 ページ）．

EPUB が持つこの特別な機能は，何が欠けているかを検討することにもつながる．そして，この欠如したものは膨大だ．ここでもまた，村田真が「EPUB は Web ではない」[1] で述べているように，EPUB には URL がない．当然ながら，EPUB にはリンクを含めることもできるし，もちろん Web サーバーに配置して，その URL が取得されたときにダウンロードすることもできる．どんな種類のファイルでも Web 経由で配布でき，EPUB 文書も例外ではない．しかし，Web ページは単に Web 経由で配布されるだけではない．これらも Web の一部なのだ．ある Web ページを他の Web ページに繋ぐことが出来る．そこに存在する URL は Web ページの本質的な構成要素であり，いかに振る舞うかを決定するとともに，Web のセキュリティと実行モデルに大きな影響を与える本質的な部分なので

ある．技術的な詳細から一歩後退してハイパーテキストのより深い意味を見れば，小林龍生が「ハイパーテキスト論再考」[2] で行っているように，私たちは全体として相互接続されたハイパーテキスト媒体 - ウェブ - を単一のエンティティであると見做すことが可能である．「Web ページ」という言葉は，当たり前のように使われているが，実際にはかなり興味深いものである．ページは本の本質的な構成要素である．では，Web ページはどのような本の構成要素なのだろう．まるでウェブ全体が一つのハイパーテキストであるかのように思われる．たとえ読者がある特定の視点でハイパーリンクを辿っていったとしても，実際には Web 全体から切り離された部分など一つもないのだ．

EPUB documents do not partake in that. They are standalone pieces of content, with well-defined boundaries. As they use hyperlinks, they can each be their own hyperbook. But they are distinct from the word wide hyperbook known as the web.

EPUB 文書は，この議論とは関わりがない．EPUB 文書は，明確な境界を持った独立したコンテンツである．ハイパーリンクを用いることにより，EPUB 文書もそれ自身がハイパーテキストであることが出来る．しかし web として知られる世界を覆い尽くす巨大なハイパーテキストとは異なる．

This too ties back into how the web evolves and what it evolves towards. As a single entity trying to cover all human communication, no feature that is relevant to any medium is ever really out of scope. And as no feature is ever retired either, gradually, the web grows more ever more capable.

この観点もまた，Web がどのように進化していくのか，そしてそれが何に向かって進化するのかに結びつく．すべての人間のコミュニケーションを網羅しようとしている単一のエンティティとして，何らかのメディアに関連する機能はすべて検討の対象となる．そして，どの機能も廃止されることなく，Web はますます機能的になっていく．

Strategic implications for the Publishing industry
出版業界の戦略への影響

In the short term, defining and rolling out a new version of EPUB is much lower cost than evolving the web platform as a whole. In earlier days, this made EPUB a great choice to kickstart the eBook business. The web was not able to satisfy the needs of the publishing industry, and would not adapt soon enough. However, the short-term need has now been satisfied, and the emergency is mostly behind us. While the initial costs of deploying EPUB were relatively low, the ongoing costs of evolving it separately from the Web are not. Meanwhile, the web has continued to evolve, and now covers more of the needs of the publishing industry. Some requirements are still unaddressed, such as packaging, but they are nonetheless being worked on [3]. Unless there is a crisis where time to market is critical, it seems

[3] worked on
https://github.com/WICG/webpackage

that the publishing industry has more to gain in joining forces with those who are trying to improve the feature set of the web to cover its needs than in trying to maintain a separate format such as a new iteration of EPUB or WPUB, and having to separately evolve all the useful features that the web ecosystem grows.

短期的には，新しいバージョンの EPUB を定義して展開することは，Web プラットフォーム全体を進化させるよりもはるかに低コストですむ．初期には，このことにより，EPUB は eBook ビジネスをキックスタートさせるための素晴らしい選択となった．Web は出版業界のニーズを満たすことができず，すぐには適応出来なかった．しかし，短期的な必要性は今や満たされており，緊急避難的対応は，すでに過去のものとなった．EPUB を導入するための初期コストは比較的低かったが，Web とは別に進化させるための進行中のコストはそうではない．この間も，ウェブは進化し続けており，そして今や出版業界のより多くのニーズをカバーしている．パッケージングのようないくつかの要件はまだ対処されていないとはいえ，作業は鋭意進められている [3]．市場投入までの時間が重要であるという危機がない限り，出版業界は，web のエコシステムの進化を横目に見て，EPUB や WPUB のような独自のフォーマットのメインテナンスを繰り返すよりも，Web の機能セットを改善してそのニーズをカバーしようとしている人々と力を合わせるほうがよいように思われる．

[3] 作業は鋭意進められている
https://github.com/WICG/webpackage

リフロー可能なドキュメント環境とは

木田泰夫

昨今人々が日々経験する日本語の多くはデジタルデバイスの画面にあって，それはメールやウェブページに見られるように横組み，ラグ組，パラグラフが改行で区切られた組版であるように思う．この謂わばデジタルネイティブな組版の姿に想いを馳せ，その日常に接する日本語の読みやすさ，美しさを底上げし，未来の姿に繋いで行きたいたいと思うとき，それが従うべき組版の規則は，印刷におけるそれとは自ずから異なっているであろう．デジタルデバイスで起こる組版が，印刷の組版とは根本的に異なる成り立ちを持っているからだ．

デジタルネイティブな組版

印刷における組版とデジタルデバイスにおける組版の根本的な違いは，誰が組版を決定するのかにある．

印刷においては送り手が組版を決定する．組版結果を人の目でチェックすることが可能で，不足があれば人手で調整が行われる．結果は印刷というプロセスを用いて画像として固定され，単一のコピーが受け手の元に届けられる．

対してデジタルデバイスにおいては受け手が組版を決定する．組版の決定は各々の受け手の環境を反映してソフトウェアが動的に，かつ完全に自動的に行う．受け手の環境が単一でないため，例えば行の折り返し位置は受け手ごとに異なる可能性がある．結果膨大な数の異なる組版結果が生成されることになる．

この違いは組版のあり方に根本的な影響を与えるのだが，それは半ば見過ごされてきたように思う．従来の努力の多くは印刷をそのままデジタルデバイスで再現することに払われてきた．例えばPDFやEPUBはそのような方向の素晴らしい成果だ．従来の形態の再現は新しい技術の受容の過程でしばしば必要となる．例えば印刷技術がこの世に登場した初期にはそれまでの技術である写本の体裁を模倣することに努力が払われた[1]．自動車も登場当初は馬車に似た形をしていた．これは技術が受け入れられるために必要な回り道だが，自動車はいつまでも馬車の速度で走るわけには行かない[2]．

この一文ではデジタルデバイスの特質を正面に捉えて考察することで，その土壌からどのような組版が立ち現れるのかを垣間見てみたいと思う．そのことを考えることは，メールやウェブページなど日常に

1　Andrew Pettegree (2010). *The Book in the Renaissance*. Yale University Press（桑木野幸司訳 (2015). 印刷という革命．p. 64．白水社）

2　脇道に逸れるが，米国において速度違反の初めての逮捕者が出たのはニューヨーク，猛スピードの時速19キロで走っていたところを自転車で追いかけてきた警察官に捕まったのだそうだ．

接する日本語の組版の質を上げる一助になるとともに，印刷を前提としない，デジタルネイティブな書籍を未来につなげてゆくことにも資するであろうと思う．

デジタルデバイスにおける組版の性質

デジタルデバイスにおける組版の性質を少し掘り下げてみよう．

完全な自動化

印刷の組版規則は人手で組版を行うための原則として生まれた．時代を経て緻密化したが，典型的ではない場合，つまり高度な判断が必要な場合に対して人によるケースバイケースの処理を期待するという性質を受け継いでいる．対して，デジタルデバイスにおける組版はソフトウェアにより実行されるので，例外的な場合に対してでも破綻せず合理的な組版を完全に自動的に決定できる必要がある．

世界の中の日本語

デジタルデバイスにおける組版は，国際化ソフトウェアのアーキテクチャ，例えばUnicode環境や国際化された基本APIの上で実装されるので，その環境で実装のしやすいものである必要がある．また，国際化されたソフトウェアは複数の言語を同時に扱うことを想定しており，例えば他の言語の中に日本語が埋め込まれる場合やその反対の場合を想定する必要がある．そのため，基本的な文字や行の配置方法に整合性が取れている必要がある．別の言語，特に英語で採用されている組版方法が日本語にも流れ込んでくる．例えばイタリックはその例である．さらに，目的の同じ組版が言語ごとに異なる場合，既に実装されている組版方法を採用する方向に圧力が働く．例えば注の組版方法として脚注が既に実装されている場合，割注の優先順位は下がる．

組版のコスト

デジタルデバイスにおいて新しい組版の機能を追加することは非常に高価につく．全世界に広がる規格専門家の間の合意と，数十人単位のエンジニアによる開発が必要だからだ．対して印刷のための組版では比較的安価に新しい試みを行うことができる．一つの部門内での合意と，好奇心旺盛な組版担当者が一人いればそれで足りるのだ．

逆に，組版をテキストに適用するコストはデジタルデバイスが劇的に低い．そこに人や高価な製造機械は介在せず，安価なデジタルデバイスがソフトウェアを一瞬の間実行するだけである．

紙面の制約

歴史的に書写や印刷において，紙自体，つまり紙面のコストが大きく，それが組版に大きな影響を与えてきた．例えばパラグラフの示し方の変遷はその一例である．デジタルデバイスにおいては紙面のコストは無視することができ，その代わりに画面の大きさというユーザー

インターフェースからの制約が生まれている．

説明可能性

国際化されたソフトウェアの中で，日本語は多数の言語の中の一つであり，関与する専門家は必ずしも日本語に精通しているわけではなく，また日本語のみに対して多くの開発コストをかけることもできない．デジタルデバイスにおける組版規則は，その各々の規則の必要性を非日本語話者に対して合理的に説明できる必要がある．これに対し，印刷組版規則は組版の専門家集団の間で継承されるものであったため，複雑な規則を多くの説明なしに継承してゆくことが可能であった．合意の形成や開発に必要な時間とコストを考えると，組版機能は自ずからよりシンプルなものになろう．

実 行 速 度

デジタルデバイスにおいては大量のテキストの組版を一瞬で完了させる必要がある．特に例えば文字の大きさをインタラクティブに変える動的な UI のある場合，その操作に追従して滑らかな動きとして見せるためには，アニメーション動画において動きをスムースに見せるために必要な速度と同じ，毎秒数十コマの速度で組版を完了して画面を更新する必要がある．これはアニメーションにおいて動きをスムースに見せるために必要な毎秒のコマ数と同じことである．ゆえに組版の規則は高速な実行を可能にするものである必要がある．

リフロー可能

デジタルデバイスにおいては送り手が組版結果を厳密に指定することができない．送り手はスタイルシートを用いて組版の大枠を指定するが，最終的な結果はそれぞれの受け手の環境や好みに従う．特に行長はデバイスの画面サイズやウィンドウのサイズに従って決定されるため，あらかじめ予見したり，指定したりすることができない．その結果，どこで行が折り返されるのか，長いグループルビが行の途中に来るのか折り返しにぶつかってしまうのか，ページに区切られたテキストの場合，どこでページが区切れるのか，そもそも全体が何ページなのか，どれも送り手側では知ることができない．ページが一定でないため，引用，参照の方法としてページ番号を使うことができない．これは重要な点だ．

内容がページに分かれた本，冊子本の発明は歴史的に見て印刷の発明に勝るとも劣らない大きなインパクトがあったと言われている[3]．それは冊子本がそれ以前から存在した巻子本に対して大きな優位性を持っていたからだ．収納・管理の容易性，ランダムアクセス性などに加えてページ番号による参照の安定性も重要な点に思われる．しかしデジタルデバイスにおける冊子本は，これら優位点を失っている．収納・管理の容易性，ランダムアクセス性は冊子本の形式に頼らずに達成されており，冊子本はリフローによりページ番号による参照の安定性を失っている．それならばデジタルデバイス上の冊子形式の意味はどこにあるのであろうか？

3　Andrew Pettegree (2010). *The Book in the Renaissance*. Yale University Press（桑木野幸司訳 (2015). 印刷という革命．p. 18–21．白水社）

このように見て行くとデジタルデバイスの組版にはアーキテクチャの違いから来る制約が存在することがわかるが，そのコインの裏には大きな可能性がある．それはデジタルネイティブな組版について考えるモティベーションとなる．

新たな可能性

アクセシビリティ，ユニバーサルデザイン

印刷における一つの大きな制約は印刷のコストの大きさである．前に述べたようにデジタルデバイスにおいてはこれが劇的に安価かつ高速なため，受け手側で組版を行うことができる．冒頭で述べた，誰が組版を決定するのか，という点である．デジタルデバイスでは受け手が組版を決定する，言い換えるとリフロー可能なため，受け手の必要に応じて表示を最適化することができる．これはアクセシビリティを考慮する際に特に有利な特徴である．

文字の大きさを自在に変えることができる機能は視力に衰えのある場合などに便利であるのはよく知られている．また例えばディスクレシア，読字障害はより複雑な障害だ．その内容は多様で，文字を目で追い，文字のまとまりを単語として認識し，読みに変換するという文を読む過程のどこに障害があるかによって，それを支援するために必要な表示方法に違いがあると言われている．デジタルデバイスなら，個々の障害の種類や程度に合わせて多様で最適な組版と表示を生成することができる．

同様に，ルビの表示の程度を変える，総ルビにするなどの表示の最適化を行うことが可能なら，子供や外国人などの日本語学習者に最適な表示を動的に生成することができる．

きめ細やかな表示の最適化が可能になることにより，アクセシビリティの枠を超え，万人にとってのユニバーサルデザインを追求することができるであろう．

ラグ組の可能性

メールやウェブにおける文字揃えのデフォルトはラグ組である．おそらく，初期のコンピューターに両端揃えを高速に行えるだけの力がなかった故であろう．

日本語の印刷においては両端揃えが標準であるので，コンピューターのデフォルトがラグ組であることを，是正しなくてはならない欠点として捉えることもできよう．しかし，同様に両端揃えが主流であった欧文の印刷組版においてラグ組の利点が認識され選択肢として広まったように，日本語においてもラグ組を積極的に捉えることもできるだろう．

日本語のラグ組にはいくつかの利点を見ることができる．小さなデジタルデバイスなどで行長が短い場合，外国語やグループルビなどが含まれる場合などに，行の調整によって字間の空きが大きくなってしまって読みにくくなるという問題を回避することができる．手書きで

は文節など意味単位で行を折り返すことが多い．文節単位で折り返すことによって，読字障害を持つ人々のみならず一般の読者においても視点移動が効率化され，読み速度が向上することを示す研究結果がある[4]．対して両端揃えでは，行の調整の必要性を減らすために，拗音や撥音といった発音単位でさえ分割を許す場合が多い．ラグ組にすることで，人が読みやすい折り返し方法を無理なく取り込むことができる．

もちろん，美しく組版された両端揃えにはラグ組にはない幾何学的な美しさがある．複数の文字揃え方法を並立させることによって組版の可能性がより豊かになるであろう．

インタラクティブな表示の活用

ユーザーの操作に応じたインタラクティブな動作が可能であるので，例えば注釈をオーバーレイとして表示させるなど，新たな組版の可能性を考えることができる．

最 後 に

ここまでデジタルデバイスの持つ特質から現れる組版の性質について考察してきた．

印刷における組版規則は明治当時の最新技術であった金属活字の技術のもとで，組版作業者が美しく読みやすい組版を経済的効率性を以って作成できるような指針として生まれた．それは金属活字や印刷が持っている可能性と同時にその制約を反映したものとなっている．その構造はデジタルデバイス上で根底から変わる．

それでは，デジタルデバイスの土壌から立ち現れる組版とは具体的にどのようなものになるであろうか．その要件を定義するにはかなりの研究と作業が必要になろうが，それはここで見たようにアクセシビリティなどに新たな可能性を内包しており，書籍やドキュメントを将来に繋げてゆくために必要な仕事だと思われる．

ここではそのためのアプローチの提案をいくつかメモ的に示しておこう．

- 印刷で出版されている書籍をその組版のルールを保ったまま電子化するような用途はスコープ外とする．
- シンプルにすることにより提案・実装コストを下げ，それにより，より早期に，より多くの，またよりベンダ間で統一の取れた実装が得られることを期待する．
- JLReq においては組版の現在の姿を表すために実装とは独立の記述方法をとっている．デジタルネイティブな組版においては技術との整合性が重要なポイントの一つであるため，実装を考慮した記述となろう．エンジニアが読んで実装に移れるような記述にすることが重要である．
- 複数の組版方法のある場合，可能な限りデフォルトの選択を示し，または選択のための基準を示す．例えば行長に対する最適な

4　小林潤平，関口隆，新堀英二，川嶋稔夫：文節間改行レイアウトを有する日本語リーダーの読み効率評価，人工知能学会論文誌，Vol. 30, No. 2, pp. 479–484 (2015)

行間や禁則の選び方など．

・広く受け入れられている規格や実装との整合性をつける．特に文字の定義は Unicode に基づいたものにする．例えば UAX #44 Unicode Character Database 特に General Category，UAX #11 East Asian Width，UAX #50 Unicode Vertical Text Layout など．

・欧文レイアウトでカバーされている機能には踏み込まず，欧文での requirements を参照する．例えば欧文間隔の振る舞い，数式，添字，単位記号など．

・既存の実装で既に同等な目的の機能が達成されている場合，日本語組版に独特な表現を保存する実質的な利点があるかを考える．

簡便な行組版ルール（案）

小林　敏

1　簡便な行組版ルールの必要性

1）ラテン文字を主にした行組版処理

ラテン文字を主にした組版では，原稿段階で，アキを確保する位置には，ワードスペース（欧文間隔，UCS: 0020)）などが挿入され，また，ワードスペースの挿入された位置で分割される（行末のワードスペースはアキを確保しない)．“行頭行末そろえ（justification)”が採用され，行の調整処理が必要になった場合にも，このワードスペースの空き量を変更して調整する．

つまり，ラテン文字を主にした組版では，文字間にアキを確保する，あるいは分割，行の調整処理の自動処理が必要になるが，その箇所は原稿に明示的に示されているといえよう．

2）日本語の行組版処理の基本

日本語の行組版処理においては，配置する文字の多くの字間はベタ組であるが，幾つかの文字の組合せ，あるいは行頭に配置する場合において一定のアキを確保する，あるいはアキを調整する処理が必要になる．

例えば，以下のような例がある．

(1) 行頭に配置する括弧類（詳細は後述）
(2) 句読点と括弧類が重なった場合（詳細は後述）
(3) 漢字や仮名とアラビア数字の間（詳細は後述）

これらの処理箇所は，原稿の文字列（つまりテキスト）において，特に明示的に示されているわけではなく，組版処理の段階で自動的な処理が期待され，実際にも多くの DTP などでは，組版段階で自動処理が行われている．

また，文字を行に配置していく場合，指定された行長に達したときは適当な文字間で２行に分割しないといけない（以下，単に分割という)．この分割してよい箇所（字間)，あるいは分割してはいけない箇所（字間）も，原稿の文字列（テキスト）に明示的に示されているわけではない．こうしたことも DTP などでは，組版段階で自動処理が行われている．

さらに，日本語組版の段落の配置方法として，多くは“行頭行末そろえ（justification)”が採用されている．この場合，行長に過不足が発生した行では，“行の調整処理”が必要になる（“行の調整処理”が必要となる原因と基本的な考え方は JLReq の“3.8　行の調整処理”

ラテン文字と同じ処理の日本語組版　本文で述べたように日本語組版には処理上で解決しないといけない問題がある．この問題を回避すればラテン文字を主にした組版と同様な処理が可能になる．例えば，以下のような方法が考えられる．

句読点や括弧などアキの処理については，まず句読点や括弧などのフォントデータの字幅としてアキを保持しないで，字面に応じた字幅（例えば半角）にする．そのうえで，アキを確保したい箇所には何らかのアキを確保する符号（例えば二分スペース）を挿入しておく．このアキは，行中にある場合は確保されるが，行頭又は行末では，そのアキを確保しないとすれば，行中と行頭（行末）で違う配置法がとれる．

行末での文字間を分割する問題については，文字単位の場合でも，単語（又は文節）とする場合でも，分割可能箇所（又は分割不可能箇所）を示す何らかの符号を挿入しておく必要がある．

さらに，文字単位で分割を可とし，行頭行末そろえを選択する場合，行の調整処理の問題を解決しないといけないが，これは分割可能箇所を均等に空ける処理を行えば，とりあえず問題は解決できる．

これとは別に，以下のような制御記号を必要箇所に挿入する方法も考えられる（以前のコンピュータ組版では，このような制御文字を持っていた例もある)．

二分スペース
四分スペース
マイナス二分スペース（挿入箇所を二分詰める）
マイナス四分スペース（挿入箇所を四分詰める）
分割禁止の制御文字
分離禁止（挿入箇所を行の調整処理で字間の調整に使用しない）の制御文字

あらかじめ，組版処理以前のテキストの該当箇所に，このような制御文字を入力しておけば，問題は解決できる．

参照）．

この行の調整処理も自動処理が必要になるが，自動処理の方法や箇所は原稿に明示的に示されているわけではなく，あらかじめルールとして処理系で準備しておき，そのルールに従って自動処理が行われることになる．

3）日本語の行組版処理の詳細

こうした日本語の行組版処理の内容は，JLReq の以下の節で解説されている．

3.1　約物などの組版処理
3.2　和欧文混植処理（縦中横処理を含む）
3.5　段落整形，そろえ及び段落末尾処理
3.8　行の調整処理
3.9　文字クラスについて

そして，その詳細は，“3.9　文字クラスについて”で説明されている文字クラスを用いて，次の附属書に示されている．

B　文字間の空き量
C　文字間での分割の可否
D　行の調整処理で詰める処理が可能な箇所
E　行の調整処理で空ける処理が可能な箇所

これと比較すると，例えば，前述したように，英語の組版では，行の分割箇所には，欧文間隔（UCS: 0020）が挿入されており，行の調整処理も，主に欧文間隔の調整で行えばよいので，日本語組版に比べると，それほど複雑ではない．

4）活字組版とコンピュータでの組版処理

活字組版では，上記のルールの基本的事項は，出版社と印刷所の慣行として確立していた．必要があれば原稿の冒頭や，原稿指定票で総括的に示される，あるいは原稿の個々の箇所に指示を入れていた．

いってみれば，こうした慣行としてのルールや原稿への指示を前提に，組版担当者の判断により，それぞれの箇所で臨機応変の処理をしていた（校正段階でも，組版処理の内容は点検され，ルール違反があれば赤字で修正が加えられていた）．したがって，例外的な文字の組合せにも対応できた．ある意味で，ルールの適用は組版担当者の能力に頼っていたともいえよう．

これに対し，コンピュータでの組版処理では，機械的な処理（自動処理）が必要である．

5）組版処理ルールの単純化

JLReq で述べた方法は，DTP などでは．ほぼ実現しているが，特別にアキを確保する箇所に組版段階でスペースを挿入する，あるいはアキ量を特別に指示するなど，個別箇所での特別な処理も必要になる．

Web，特にリフローを許す環境では完全な自動処理を考える必要があり，こうした環境では個別処理は避けなければならない．そこで，読みやすさの面での品質を保ちながら，ある程度の組版処理ルールの

行頭禁則と行末禁則　日本語組版では，句読点や終わり括弧類など行頭に配置する，あるいは始め括弧類を行末に配置することを禁止するルールがある．こうしたルールはこうした約物の前又は後ろの文字間での分割を禁止することで回避できる．したがって，分割のルールを定めれば，行頭禁則と行末禁則は実現できることになる．

“行頭行末そろえ”の採用　日本語組版では“行頭行末そろえ”が多い．これは，多くの人がそれに慣れ，慣習とされていることにもよるが，日本語組版での分割は，語単位ではなく文字単位で行うことにある．そのために多くの行で指定された行長と一致することになり，一部の行のみで過不足が発生する．そこで，行頭そろえ，行末そろえ又は中央そろえを選択すると，一部の行にのみ全角又は二分程度の余白がでるだけなので，ある種の乱れととられ，見た目のバランスがよくないと考えられることによる．

ただし，行頭に比べ，行末の不ぞろいはそれほど気にならないと考えれば，行頭そろえとする方法が採用できる．この方法にすれば，行の調整処理の問題は回避できる．なお，語又は文節を単位として分割する場合，行頭行末そろえとするときは，行の調整処理は字間で調整するので，この調整量が大きくなりすぎる行が出てくる．そのため，字間が不ぞろいとなる行が多く出てくるので，行頭そろえ（行末そろえ又は中央そろえ）を選択することになろう．

出版社による行組版ルール　書籍などにおいて，多くの出版社で共通に採用されているルールが多いのであるが，細部についてみると，出版社により，あるいは出版物の種類によって，その適用されるルールが異なる場合もある．例えば，後述する行頭に配置する括弧類の配置方法には，主に 3 つの配置方法があり，出版社ごとに採用されるルールは，ほぼ決まっている．

単純化をはかるが必要も出てくる．

以下は，基本的な要素に限った，そのひとつの処理案である．

なお，ここで解説する組版方法については，このように簡便化された処理法も可能であることを示したものであり，この処理法に限定することを主張するものではない．各処理系での追加機能の工夫，例えば，行の調整処理で詰める調整ができるようにする，あるいはぶら下げ組をオプションで採用できるようにする，といったことを否定するものではない．

2　約物の字幅とアキ

1）句読点と括弧類の基本的な配置方法

字幅が全角の仮名と漢字を用い，これらの字間をベタ組とした場合における句読点と括弧類の原則的な配置方法を図 1 に示す．なお，日本語組版においては，通常は行長を文字サイズの整数倍として設定する．図 1 の右側に示す正方形の枠は，行に全角の文字が整数個並ぶ場合の文字の外枠を示す（以下，同様）．

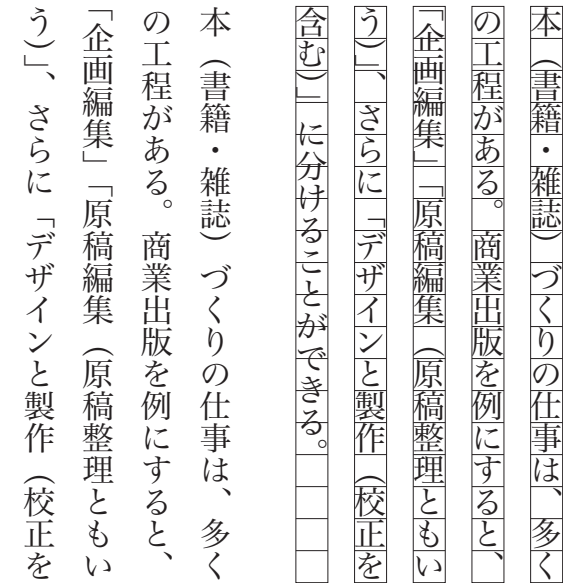


図 1　句読点や括弧類の原則的な配置方法

図 1 に示すように，句読点と括弧類が単独で漢字や仮名の間に配置される場合，重なって配置される場合，行頭や行末に配置される場合で，句読点と括弧類の占める字詰め方向の領域が異なっている．

2）句読点と括弧類の字幅は半角と考える

JLReq では，句読点，括弧類や中点類の字幅を半角（二分）として説明している．これは JIS X 4051 にならったものである．この考え方は，実際のフォントデータの字幅がどのようになっているかとは無関係であり，あくまで説明の前提としての考え方である．

図 1 に示した配置例について，字幅を半角とする考え方による説明例を前ページの図 2 に示す．なお，図 2 における約物をくくった長方形（正方形でない）は，半角と考える句読点，括弧類及び中点類の文

ぶら下げ組　ぶら下げ組は，字間による行の調整処理を少なくする方法であり，JLReq の“3.8.2　詰める処理と空ける処理”に解説がある．

なお，ぶら下げ組は，岩波書店で採用されたのが最初であり，その後，徐々に普及していった．このぶら下げ組を考案した岩波書店の初代校正課長の西島九州男は，その著書の“校正夜話”（日本エディタースクール出版部，1982 年）で，次のように述べている．

“［句読点が行末ではみ出す場合］倍数の関係でどこかで字間を割ったり句読点の下をつめたりして調整しなければならない．どうしても無理がくるし手数がかかる．そこで邪魔な読点などはとってしまえなどという無茶なことになるわけです．そういうことから，私の方で行末にはみ出す句読点を行末の文字の下にブラ下げるということを“発明した”っていいますか，初めてやりました．ブラ下げた方が活字の倍数の関係から合理的だという合理主義的な考えから始めたのですが，亡くなられた木下杢太郎氏や幸田露伴先生などはこの方法があんまり好きではなかった．”

活字組版の句読点や括弧類の字幅　活字組版における活字は，一定の高さのある台（ボディ）の上に字面がある．この台の字詰め方向のサイズが字幅になる．活字組版における句読点と括弧類の台の字詰め方向のサイズとしては，半角と全角のものが準備されており，必要に応じて使い分けていた．

なお，補足しておけば，日本語の活字組版では，植字作業に入る前に，原稿に従って活字を準備しておく“文選（ぶんせん）”という作業を行っていた．この文選作業では，仮名や漢字のみを拾い集め（採字という），句読点や括弧類，さらに字幅が全角でないアラビア数字やラテン文字などは採字しないで，植字作業で採字していた．

終わり括弧類など　句点類（cl-06）と読点類（cl-7）は，ブラ下ゲ組を考慮すると，終わり括弧類（cl-01）と振る舞いが異なる．また，句点類（cl-06）と読点類（cl-7）は詰める調整を考えると，その扱いは異なる．この 2 つを考えなければ統合できる．

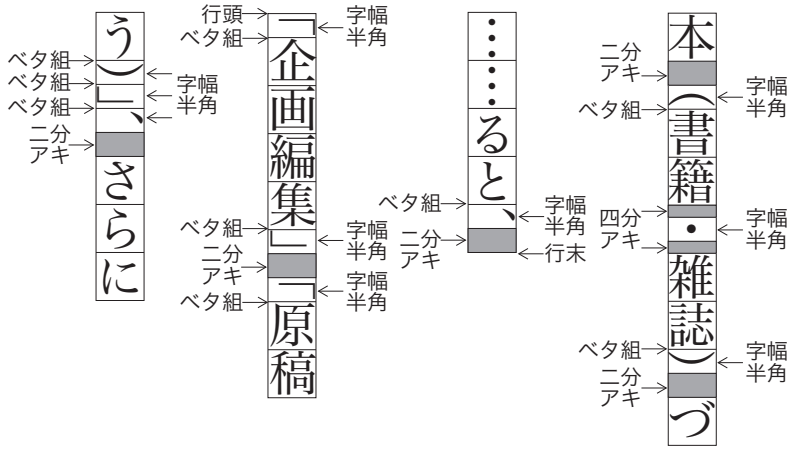


図２　字幅を半角と考えた場合の説明

字の外枠，グレーで示した枠はアキ量を示す．

3）句読点と括弧類の字幅を全角と考える

実際のフォントデータの字幅としては，句読点，括弧類及び中点類の字幅を全角としている例がある．句読点，括弧類及び中点類の字幅を全角とすれば，図１の説明は，図３のようになる．

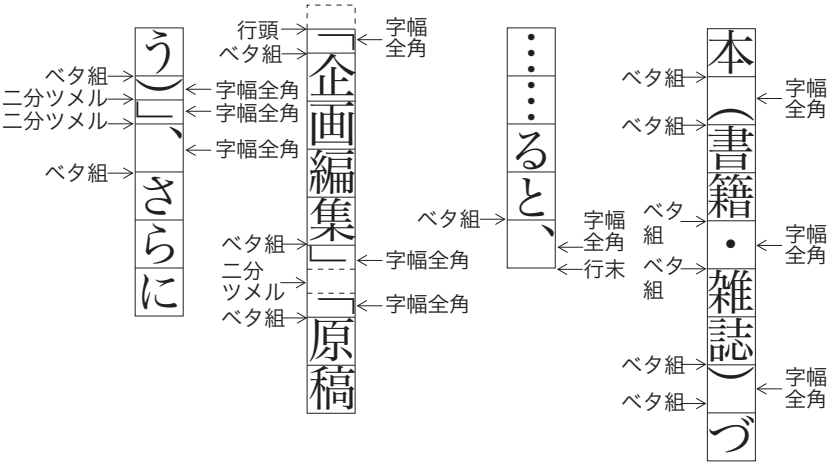


図３　字幅を全角と考えた場合の説明

字幅を半角とする，あるいは全角とする場合であっても，どちらも結果として図１になるのが，日本語組版としては望ましい形である．そこで，以下の説明では，括弧類や句読点の字幅として，実際のフォントデータでは全角としていることなどから，括弧類，句読点及び中点類の字幅を全角とし，それを前提に解説する．

3　文字クラスの統合とその配置方法

1）文字クラスの数を少なくする

JIS X 4051 や JLReq では，文字の配置方法の違いに注目し，文字や記号を文字クラスに分け，その配置方法を規定している．JLReq では，“始め括弧類”の cl-1 から“縦中横中の文字”の cl-30 まで，30 の文字クラスに分けている．

組版処理の複雑さを少なくするためには，この文字クラスの数を少

行頭禁則約物類　字幅で異なる例もあるが，行頭禁則とする約物を合併した．

区切り約物（cl-04）は，文末と文中で使用するという２つの用法がある．文末に配置し，句点としての用法も兼ねる場合，一般に区切り約物（cl-04）の後ろを全角アキにする．この処理は，通常は全角の和文間隔を挿入することで処理している．文中で使用する場合は，区切り約物（cl-04）の前後はベタ組とするが，その前後を二分アキなどにする例もある．空ける場合は，スペースなどを入れるなど，別の特別の処理が必要になる．

後置省略記号（cl-13）も行頭禁則文字として，同じ扱いが可能である．なお，“%”は，行頭配置を許容する配置法もあるが，前の数字と一体で扱った方が望ましく，その処理法は，ここでは考えなかった．

また，これらをアラビア数字の前後に配置する場合，漢字や仮名とは配置処理が異なるが，ここにまとめた約物類はアラビア数字との字間をベタ組とするので，同じ扱いが可能である．

中点類（cl-05）の字幅を半角と考えれば，区切り約物（cl-04）とアキが異なり，詰める調整でも，その扱いは異なる．字幅を全角と考えれば，同じ扱いが可能となる．ただし，行頭禁則文字として扱う必要があり，漢字等とは別の扱いになる．また，ルビの処理で異なるので，別にした．

行頭禁則和字　ここに含まれる小書きの仮名などは，今日，書籍などでは行頭禁則とする例は多くないので，仮名や漢字等に含ませることが可能である．しかし，Web では行頭禁則としている例もあり，また，音読を考慮すると，これらを行頭禁則とす必要もあるので，文字クラスとして残した．

分離禁止文字　２倍ダーシや２倍の３点リーダーなどは分割禁止とすることから１つの文字クラスになっている．単独で使用する全角ダーシや３点リーダーなどは，行頭禁則にも行末禁則にもしない．２倍ダーシは，それを１つの文字として扱えば分割できない．また，２倍３点リーダーは，活字組版では分割も許容されていたので，ここでは分割禁止としないで，一般の和字に含ませることにした．

なお，ダーシとリーダーで扱いが異なっているのは，活字組版において，２倍ダーシは１つの台（ボディ）に鋳込まれており，２倍３点リーダーは，全角の３点リーダーを２つ並べていたことによる．

なくすることも必要になる．

2）統合あるいは採用しない文字クラス

ここでは，次のように文字クラスを整理する．新しい文字クラスは cl-i から cl-xii までの 12 個とする．なお，ここでの新しい文字クラスの番号は，JLReq と区別するためにローマ数字の小文字で示す．

(1) 始め括弧類（cl-i） 始め括弧類（cl-1）と同じ
(2) 終わり括弧類等 (cl-ii) 終わり括弧類 (cl-01)，句点類 (cl-06)，読点類（cl-7）を統合
(3) 行頭禁則約物類（cl-iii） ハイフン類（cl-03），区切り約物（cl-04）及び後置省略記号（cl-13）を統合
(4) 中点類（cl-iv）：中点類（cl-05）と同じ
(5) 行頭禁則和字 (cl-v) 繰返し記号 (cl-09)，長音記号 (cl-10)，及び小書きの仮名（cl-11）を統合
(6) 前置省略記号（cl-vi） 前置省略記号（cl-12）と同じ
(7) 和字間隔（cl-vii） 和字間隔（cl-14）と同じ
(8) 仮名・漢字等（cl-viii） 平仮名（cl-15），片仮名（cl-16）及び漢字等（cl-19）を統合
(9) 親文字群中の文字（ルビ付き）(cl-ix) 親文字群中の文字（熟語ルビ以外のルビ付き）(cl-22) 及び親文字群中の文字（熟語ルビ付き）(cl-23) を統合
(10) 欧文間隔（cl-x） 欧文間隔（cl-26）と同じ
(11) 欧文用文字 (cl-xi) 親文字群中の文字 (添え字付き) (cl-21)，単位記号中の文字（cl-25）及び欧文用文字（cl-27）を統合
(12) 縦中横中の文字（xii） 縦中横中の文字（cl-30）と同じ

なお，連数字中の文字 (cl-24) は，削除した．また，等号類 (cl-17) と演算記号（cl-18），合印中の文字（cl-20）及び割注始め括弧類 (cl-28) と割注終わり括弧類（cl-29）も削除した．ここでは基本的事項に限定すること，あるいは，それらは各要素で配置処理は解説されていることによる．

3）各文字クラスの配置方法 1—文字間の空き量

それぞれの文字クラスの文字間の空き量を表 1 に示す．

表 1 においては，縦列の先頭列に，前に配置する文字クラス名を示し，横行の先頭行に，後ろに配置する文字クラス名（文字クラス名は略して表示）を示す．それぞれの交点のセル（こま）において，空き量を以下の記号で示す．

空白：前に配置する文字クラスの文字と，後ろに配置する文字クラスの文字の字間をベタ組とする，又は行頭若しくは行末との間をベタ組にする．
1/4：前に配置する文字クラスの文字と，後ろに配置する文字クラスの文字の字間を四分アキとする．ただし，配置される前後に親文字群中の文字（ルビ付き）(cl-ix) でルビのはみ出しがない場合，又は四分以下のはみ出しがある場合であり，はみ出しが四分

仮名・漢字等 仮名と漢字では，親文字からルビがはみ出した場合に，その扱いが異なることから別の文字クラスになっている．しかし，親文字からのルビのはみ出しを，仮名にも掛けないとすれば，同じ扱いとなる．

親文字群中の文字 親文字群とは，親文字及びそれに付随するルビ，添え字又は圏点を含めた文字群のことである．

ここに含まれる文字群は，細部にわたれば処理方法は異なる．しかし，これらは添え字又はルビ処理の問題でもあり，そこで解決すればよいことであり，その前後に配置する漢字や仮名との関係だけを考えればよいので，親文字群中の文字（ルビ付き）とした．

なお，親文字群中の文字（添え字付き）(cl-21) は，主に欧文文字で使用されるので，欧文用文字（cl-xi）に含めた．

連数字 連数字とは，コンピュータ組版が開発された当時，和文と組み合わせて使用された半角のアラビア数字などであり，今日では，ほとんど使用されていないことから削除した．

割注の括弧 割注の括弧は，その前後のアキなどについては，一般の括弧類とやや異なる処理が必要になるが，それは，割注の処理として考えればよいことなので，ここでの文字クラスからは削除した．

インライングラフックの文字クラス 行の文字と文字との間に，本文の一部として画像を挿入する例がある．インライングラフィックと呼ばれている．これと似たものとして JIS X 4051 では具体字形の処理法を規定している．“具体的な形態をもって実現した文字の個々の形”と定義しているが，画像で示した文字のことである．JIS X 4051 では“具体字形の文字クラスは，漢字と同じ文字クラスとする．これ以外の文字クラスとすることは，処理系定義とする”と規定しており，漢字と同じ文字クラスとしている（処理系定義で別の文字クラスとすることも認めている）．インライングラフィックも同様の扱いでよいだろう．

和文と欧文のアキ 和文の漢字や仮名とアラビア数字又は欧字との間は，原則として四分アキにする．これは，和文とアラビア数字又は欧文文字とで設計が異なることによる．和文の文字の外枠のアキとアラビア数字又は欧文文字の字詰め方向の外枠とのアキの違いがあるので，バランスを取るための処理である．また，漢字と仮名は，あ

以上の場合は，ルビを掛けないように空ける．

－1/2 後：後ろに配置する文字クラスの文字の前を二分詰める（図 4 参照）．

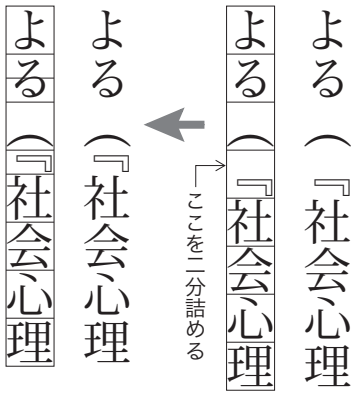


図 4 “－1/2 後”の配置処理例（左側が処理後）

－1/2 前：前に配置する文字クラスの文字の後ろを二分詰める（図 5 参照）．

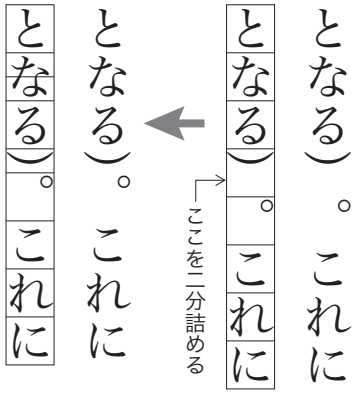


図 5 “－1/2 前”の配置処理例（左側が処理後）

－1/2 先：行頭（又は段落先頭行の行頭）に配置する文字を図 6 のように配置する．ただし，段落先頭行の場合，段落先頭行の字下ガリは全角下ガリとする．

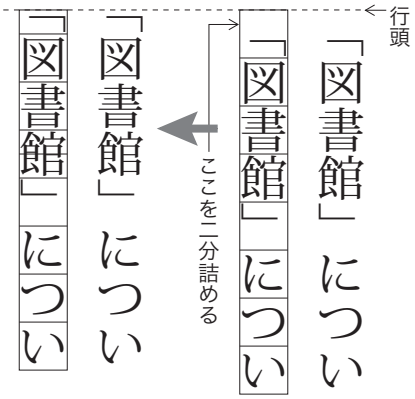


図 6 “－1/2 先”の配置処理例（左側が処理後）

－1 先：段落先頭行の行頭に配置する文字を図 7 のように配置する．ただし，段落先頭行の場合，段落先頭行の字下ガリは全角下ガリとする．

－1/2 末：行末に配置する文字を図 8 のように配置する．

る程度のまとまりを連続して読んでいくことはあるとしても，基本は 1 字 1 字読んでいくのに対し，欧文文字は，どちらからといえば，単語ごとにまとめて読んでいく傾向が強いという違いもある．こうした異質のものを混ぜて配置していく際の工夫といえよう．

ただし，四分アキとするのは活字組版の慣習にならったものであり，フォントにもよるが，四分アキが望ましいアキとは限らない．一般的にいえば，四分よりいくらか詰めた方がバランスがよい．その意味では，和文とアラビア数字又は欧文文字とのアキは，定義により変更できることが望ましい．

括弧類や句読点が連続する場合　始め括弧類（cl-i，ここでは“始め”と略す）と終わり括弧類等（cl-ii，ここでは“終わり”と略す）が連続するケースとしては，以下の 4 パターンである．

(1) 始めと始めが連続
(2) 終わりと終わりが連続
(3) 終わりと始めが連続
(4) 始めと終わりが連続

この場合，(1) は前に配置する始めは全角のまま，その前に二分の空白を保持している．これに対し，後ろに配置する文字の前を二分詰めることになる（図 4 参照）．

(2) の場合，後ろに配置する終わりは全角のままで，その後ろに二分の空白を保持している．これに対し，前に配置する文字の後ろを二分詰めることになる（図 5 参照）．

(3) の場合，JIS X 4051 や JLReq では，後ろに配置する文字クラスの文字の前を二分詰める形を示している．これは，一般に配置する文字クラスの文字サイズが同じか，又は後ろに配置する文字クラスの文字が小さいことによる．ここでは，一般化して，小さな文字サイズの文字を調整するように改めた（又は，それぞれを平均して，四分詰めるとしてもよい）（図 9 参照）．

(4) は通常は発生しない．配置例があった場合，その字間をベタ組にすればよい．

こうした処理が必要になるのは，単独で出現する括弧類や句読点とのアキをそろえ，余分なアキを削除するためである．

例えば，始め括弧類（cl-i）の後ろ及び終わり括弧類等（cl-ii）の前は，特別な場合を除き，ベタ組であり（図 4 や図 5 の処理以前は，ここが二分アキになっている），始め括弧類（cl-i）の前及び終わり括弧類等（cl-ii）の後ろは，その字形の前又は後ろに二分程度の空白を持っている（図 9 の処理以前は，ここの空白は全角に近いアキになっている）．

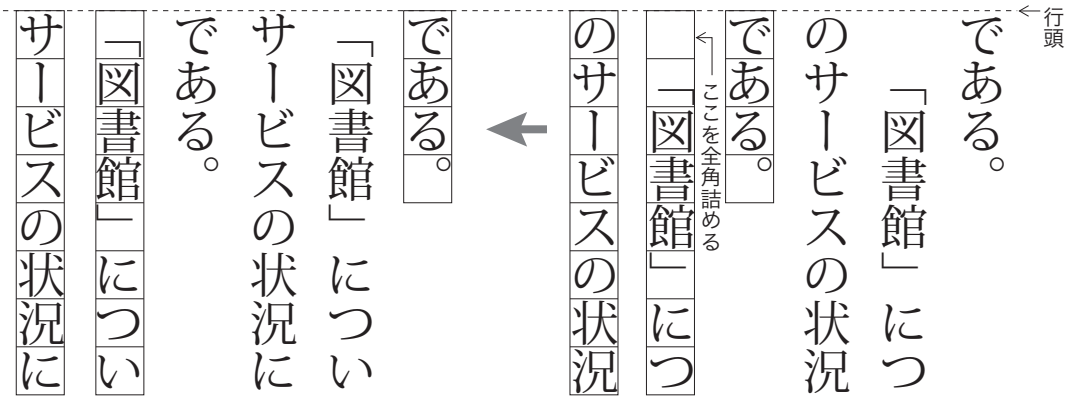


図７　“－1 先”の配置処理例（左側が処理後）

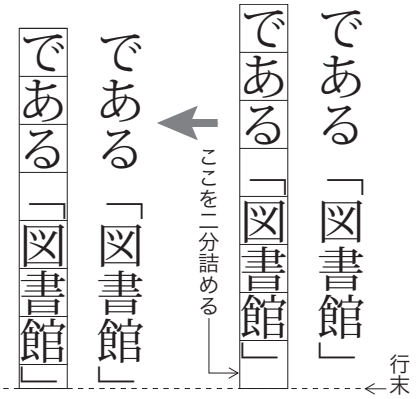


図８　“－1/2 末”の配置処理例（左側が処理後）

－1/2 連：連続する文字クラスの文字の内，小さなサイズの文字の後ろ又は前を二分詰める（図９参照），又は前に配置する文字クラスの文字の後ろを四分詰め，後ろに配置する文字クラスの文字の前を四分詰める（図９参照）．

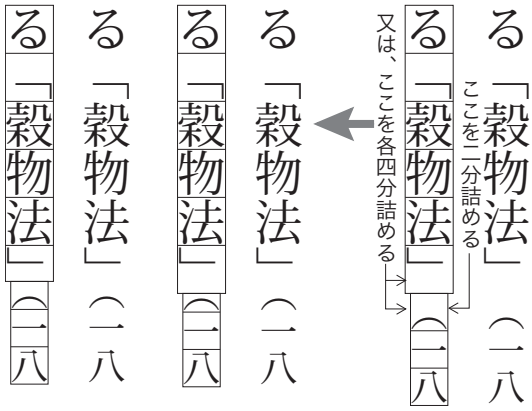


図９　“－1/2 連”の配置処理例（左側が処理後）

1/2 掛：親文字からルビのはみ出しがあった場合，ルビ文字を空白に二分まで掛けてよい．

1/4 掛：親文字からルビのはみ出しがあった場合，ルビ文字を空白に四分まで掛けてよい．

×：このような配置は分割禁止により禁止される．

無：このような配置は，通常は発生しないが，配置する場合は，ベタ組になる．

段落先頭行の行頭処理　段落の１行目の行頭に括弧類を配置する例がある．この括弧類の配置については，いくつかの方法がある．段落の２行目以下の行頭との組み合わせでは，以下の３つが代表的な配置方法である（次ページの図 10 参照）．

①の方式は，見た目で段落の１行目行頭に全角二分の空白があり，２行目以下の行頭では二分の空白がある．②では，段落の１行目行頭に全角の空白があり，２行目以下の行頭では空白がない．③では，段落の１行目行頭に二分の空白があり，２行目以下の行頭では空白がない．

活字組版では，行長に半端のでない①の方法が多かったが，今日では，この方式は少なくなり，これに代わり②の方法が多くなっている．

③の方法は，活字組版時代から小説の出版物の多い出版社で採用されていた方式で，今日でも，この方式を採用している出版物も多い．小説の場合，会話が多く，特に①の方式では，括弧でくくる会話文が下がり過ぎになることを避けるためといわれていた．したがって，この慣例を変えることは簡単ではなく，この方式を望む編集者（出版社）は多いと考えられる．

なお，③の方式は，段落の開始がやや分かりにくい場合があるという問題点がある．

なお，表 1 において，前に配置する文字クラスの最後に“行頭”及び“段落先頭行頭”とあるのは，行頭に配置した文字クラスの配置処理法を示し，“空白”のセル（ベタ組）の場合，行頭に配置する文字の先頭を行頭（又は段落先頭行の行頭）の位置にそろえ，“−1/2 先”とあった場合は図 6，“−1 先”とあった場合は，段落先頭行の行頭を図 7 のように配置する（傍注で説明している③の方式である）．なお，段落先頭行の字下ガリは，全角とする．

また，後ろに配置する文字クラスの最後に“行末”とあるのは，行末に配置する文字クラスの配置処理法を示し，“空白”のセル（ベタ組）の場合，行末に配置する文字の末尾を行末の位置にそろえ，“−1/2 末”とあった場合は，図 8 のように配置する．

4）各文字クラスの配置方法 2—文字間での分割の可否

それぞれの文字クラスの文字間での分割の可否については表 2 に示す．

表 2 においては，縦列の先頭列に，前に配置する文字クラス名を示し，横行の先頭行に，後ろに配置する文字クラス名（文字クラス名は略して表示）を示す．それぞれの交点のセル（こま）において，分割の可否を以下の記号で示す．

無印：分割が可能，ただし，複数の欧文文字が連続する場合は，その文字間では，原則として分割はできない．

否：分割はできない．

可 or 否：分割できるか，できないかは指示による．

5）各文字クラスの配置方法 3—空ける処理が可能な箇所

行の調整処理で空ける処理が可能な箇所と，その優先順位を表 3 に示す．

表 3 においては，縦列の先頭列に，前に配置する文字クラス名を示し，横行の先頭行に，後ろに配置する文字クラス名（文字クラス名は略して表示）を示す．それぞれの交点のセル（こま）において，空ける処理方法について以下の記号で示す．なお，この表では，欧文の語間の処理は考慮していない．この処理を行う場合は，“1/2 まで”をまず行い，次に欧文の語間の処理を行い，最後に，この表で示された箇所を空ける

可　：行の調整処理で，行末に配置する文字クラスの調整で処理できない場合に，空ける処理が可能な箇所を示す．

1/2 まで：行末に配置する文字クラスの後ろを二分まで空けることができ，この処理を優先する．

×：このような配置は分割禁止により禁止される．

4　分かち書きとアクセシビリティ

1）読書と語句の区切り

前述したように多くの日本語の文章では，文字の多くはベタ組で配

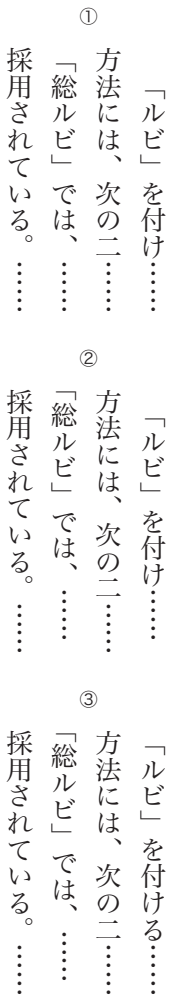


図 10　行頭の括弧類の配置例

行末の終わり括弧類等　行末に配置する句読点や括弧類の配置は，図 8 の右側のようにするのが原則であり，行の調整処理が必要な場合は，ここを優先的に二分詰める処理を行っていた．これは活字組版でも，コンピュータ組版でも共通であった．

行頭の位置をそろえるという必要性に比べ，行末の位置をそろえることは，ある程度はごまかせる，という理由からである．行末句読点や括弧類の後ろのアキは，それほど気にならないということである．

そこで，ここでは，図 8 の左側の配置法を前提に，この箇所を調整で優先的に使用した．特にその行に行長の調整が必要なければ，通常は図 8 の右側のようになる．

なお，従来からここの配置方法は図 8 の右側か，又は左側の配置のいずれかであり，その中間のアキをとる配置は行われていなかったが，これは活字組版の技術的な問題からきたもので，ここでは中間のアキをとることも認めた．

DTP などでは，行末の終わり括弧類等をすべて図 8 の左側のように配置する方法がある．ここで述べる調整法を採用しなければ，その配置方法になる．

置していく．ラテン文字を主にした文章のように単語の区切りを示すアキ又は記号は挿入されない．多くの人が，こうした文章で読みなれているのが現状である．

しかし，単語の区切りを示していない文章を読むのに困難を感じている人たちもおり，こうした人たちは，語句の区切りが示されていない印刷物の文章に，あらかじ語句ごとに斜線で区切りを入れて読んでいる例もある．

アクセシビリティを高めるためには，語句の区切りを示した組版処理も必要といえる．

漢字と仮名の使用と語句の区切り　日本語では，主に概念を表す部分（名詞・動詞・形容詞・形容動詞など）には漢字を使用し，補足的に付く部分（活用する語の語尾，助詞・助動詞，形式名詞など）には平仮名を使用している．これが語句の区切りを示す役割を果たしている．

そこで，例えば，別の語句である漢字や仮名が連続する場合は，語句の区切りがあいまいになるので，読点を挿入するなどの工夫が必要になる場合もある．

2）文節で区切る“分かち書き”

語句の区切りを示す方法として，子ども向けの本で“分かち書き”が採用されている．例えば，小学校の国語と道徳（1 年生及び 2 年生）の教科書，子ども向けの平仮名を主にした本の一部，DAISY 教科書の一部で分かち書きが採用されている．この方法は，文について文節を単位にして区切って表示する方法であり．文節の区切りを示す方法としては，一般に全角の和字間隔 (cl-14，この文書の文字クラスでは cl-vii）を用いている（後ろに掲げる図 11 及び図 12 参照）．

形態素解析による分かち書き　形態素解析による分かち書きは，正確性に欠ける部分があるので，人為的な方法が必要になる．

3）文 節 と は

文節という用語は，橋本進吉が最初に使用した用語であり，橋本進吉著 “国語法研究”（岩波書店，1947 年，参照したものは 1976 年の第 22 刷）では，“文を実際の言語として出来るだけ多く区切つた最短い一区切”を文節と名づけ，次のような例を掲げている（文節の区切りを縦線で示す）．

私は｜昨日｜友人と｜二人で｜丸善へ｜本を｜買ひに｜行きました．

そして，同書では，その形の上からみれば，次のような特徴があると述べている．

一　一定の音節（これは無論一つ又は二つ以上の単音から出来たものである）が一定の順序に並んで，それだけはいつも続けて発音せられる（その中間に音の断止が無い）．

二　文節を構成する各音節の音の高低の関係（即ちアクセント）が定まつてゐる．例へば，東京語では，“今日も”は“キョオモ”のキョの部分を高く，オとモの部分を低くいつも発音し，“いい(好)”は初のイを高く，次のイを低く発音するなど．

三　実際の言語に於ては，その前と後とに音の切目をおく事が出来る．

四　最初に来る音とその他の音，又は最後に来る音とその他の音との間には，それに用ゐる音にそれぞれ違つた制限がある事がある．（例は省略）

さらに，文節は，一つ又は二つ以上の語（単語）から成り立っており，語について，それだけで単独で一文節をなす独立できる語と，それだけで文節を作ることができず，独立できる語と共に文節を作る語（独立できない語）とに分けている．

3）分かち書き文書の要件

分かち書き文書では，次のような要件が必要である．

―分かち書きの区切りとしては，従来から行われている文節による分かち書きが，慣れていることから望ましいといえよう．

―文節区切りは，教科書などとそろえる必要があり，義務教育段階では正確な分かち書きが求められるている．

―分かち書きの区切りとしては，全角の和字間隔（cl-vii）を用いる．

―句読点や括弧類の配置方法は，一般の場合と同じである．

―分かち書きの段落処理としては，行頭そろえ又は行頭行末そろえ（justification）が多く採用されている．

―行頭そろえの場合，２行にわたる分割については，文節の途中では分割しない．分割は，分かち書きの区切り又は句読点や括弧類の後ろとする．なお，分かち書きの区切りの和字間隔（cl-vii）が行末に配置される場合は，全角アキを確保し，行頭に配置される場合は，全角アキを確保しないで，文節の語句の先頭を行頭に配置する．宮沢賢治作“注文の多い料理店”の冒頭の文を使用した例を図 11 に掲げる．

―行頭行末そろえ（justification）の場合は，一般の行処理と同じで，文節の途中で２行にわたる分割は可能である．なお，分かち書きの区切りの和字間隔（cl-vii）は，行頭そろえの場合と同じである．図 11 と同文の例を図 12 に掲げる．（図 12 の右から２行目の行頭に和字間隔（cl-vii）が配置されることになるが，全角アキにしていない．逆に左から２行目の行末では全角アキとしている．）

―読むことにある程度慣れてくると，分かち書きしない文章で十分読める，あるいは，そうした文章に慣れていくためにも，分かち書き無しの表示も必要である．つまり，同一コンテンツを分かち書き／分かち書き無しと表示し分ける必要がある．

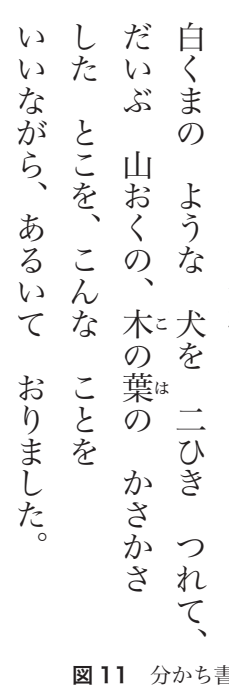


図 11　分かち書きの例 1

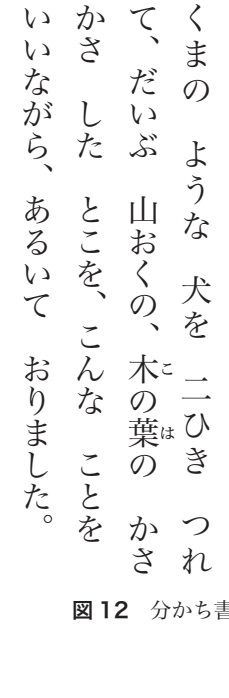


図 12　分かち書きの例 2

4）文節又は語を単位にする分割

語を認識することに困難を抱える児童にとって，文節又は語の途中での２行にわたる分割は理解の妨げとなる．こうしたことから，分かち書きをしない組版であっても，文節又は語を単位とした分割処理も必要になる．

また，一般の読者にとっても，文節又は語の途中での分割に慣れているが，実際に文字を読んでいく場合，１字単位ではなく，ある程度のまとまり（単語あるいは文節単位）で読んでいく．となれば，段落内で行を折り返す場合，文字単位で折り返す位置を決めるのではなく，単語あるいは文節単位で折り返す方が読みやすいと考えられるし，小林潤平などの実証研究でも，そのことが示されている．

こうしたことから，分かち書きをしない一般の文章でもあっても，文節又は語を単位とした分割処理が選択できるようになることが望ましい．この場合の分割位置は，教科書で採用されている分かち書きほ

どの厳密さは要求されないので，形態素解析などの前処理で分割位置を決めるといった方法が採用できる．

なお，文節又は語を単位として分割する場合の段落処理は，行頭行末そろえ (justification) にすると，行の調整処理での字間の処理量が大きくなる行も出てくるので，行頭そろえなどにするのが望ましいといえよう．

単語又は文節単位で折り返す場合の段落整形処理　行頭行末そろえの行の調整処理で空ける処理を行う場合，調整量が全角程度であれば，読者にとっては，ほとんで気にならないが，調整量が２倍，３倍ともなれば，字間のアキがそれなりに目立つ．これは，ベタ組に慣れた読者にとっては，あまり読みやすいとはいえない．単語又は文節単位で折り返す場合に行頭行末そろえを選ぶと，空ける調整では，かなりの調整量となる例もでるので，現実的な処理としては行頭そろえとなるであろう．

表 1　文字間の空き量

		後ろに配置												
		始め (cl-i)	終わり (cl-ii)	禁則約 (cl-iii)	中点類 (cl-iv)	禁則和 (cl-v)	前置省 (cl-vi)	和間隔 (cl-vii)	仮名等 (cl-viii)	ルビ付 (cl-ix)	欧文間 (cl-x)	欧文用 (cl-xi)	縦中横 (xii)	行末
前に配置	始め括弧類（cl-i）	−1/2 後												×
	終わり括弧類等（cl-ii）	−1/2 連	−1/2 前		−1/2 前			−1/2 前		1/2 掛				−1/2 末
	行頭禁則約物類（cl-iii）													
	中点類（cl-iv）	−1/2 後								1/4 掛				
	行頭禁則和字（cl-v）											1/4		
	前置省略記号（cl-vi）													×
	和字間隔（cl-vii）	−1/2 後								1/2 掛				
	仮名・漢字等（cl-viii）											1/4		
	親文字群中の文字（ルビ付き）（cl-ix）	1/2 掛			1/4 掛			1/2 掛				1/4		
	欧文間隔（cl-x）													
	欧文用文字（cl-xi）					1/4			1/4	1/4			1/4	
	縦中横中の文字（xii）											1/4		
	行　頭	−1/2 先	×	×	×	ベタ組又は禁止								
	段落 1 行目の行頭	−1/2 先又は−1 先	無	無		無								

表２　文字間での分割の可否

		後ろに配置											
		始め (cl-i)	終わり (cl-ii)	禁則約 (cl-iii)	中点類 (cl-iv)	禁則和 (cl-v)	前置省 (cl-vi)	和間隔 (cl-vii)	仮名等 (cl-viii)	ルビ付 (cl-ix)	欧文間 (cl-x)	欧文用 (cl-xi)	縦中横 (xii)
前に配置	始め括弧類（cl-i）	否	否	否	否	否	否	否	否	否	否	否	否
	終わり括弧類等（cl-ii）		否	否	否	可 or 否							
	行頭禁則約物類（cl-iii）		否	否	否	可 or 否							
	中点類（cl-iv）		否	否	否	可 or 否							
	行頭禁則和字（cl-v）		否	否	否	可 or 否							
	前置省略記号（cl-vi）	否	否	否	否	否	否	否	否	否	否	否	否
	和字間隔（cl-vii）		否	否	否	可 or 否							
	仮名・漢字等（cl-viii）		否	否	否	可 or 否							
	親文字群中の文字（ルビ付き）（cl-ix）		否	否	否	可 or 否							
	欧文間隔（cl-x）		否	否	否	可 or 否							
	欧文用文字（cl-xi）		否	否	否	可 or 否							
	縦中横中の文字（xii）		否	否	否	可 or 否							

表３　行の調整処理で空ける処理が可能な箇所

		後ろに配置												
		始め (cl-i)	終わり (cl-ii)	禁則約 (cl-iii)	中点類 (cl-iv)	禁則和 (cl-v)	前置省 (cl-vi)	和間隔 (cl-vii)	仮名等 (cl-viii)	ルビ付 (cl-ix)	欧文間 (cl-x)	欧文用 (cl-xi)	縦中横 (xii)	行末
前に配置	始め括弧類（cl-i）													×
	終わり括弧類等（cl-ii）													1/2 まで
	行頭禁則約物類（cl-iii）													
	中点類（cl-iv）													
	行頭禁則和字（cl-v）					可			可	可			可	
	前置省略記号（cl-vi）													×
	和字間隔（cl-vii）													
	仮名・漢字等（cl-viii）					可			可	可			可	
	親文字群中の文字（ルビ付き）（cl-ix）					可			可	可			可	
	欧文間隔（cl-x）													
	欧文用文字（cl-xi）													
	縦中横中の文字（xii）					可			可	可			可	

読み効率を高める日本語電子リーダー設計の試み*

小林潤平**

1 はじめに

日本語文章をスムーズに目を動かして読んだときの速度を計算すると，1分あたり1,200文字程度となる．しかし，日本語文章の平均的な読み速度は，1分あたり500〜600文字と言われている[1,2]．計算上の速さと，実際の平均的な速さには，大きな開きがある．

多くの情報が文字を介して伝達される現在，もし電子リーダーにおける表示の工夫で「読み」を効率化できれば，その効果の大きさは計り知れない．小説や新聞，実用書や仕事の書類などを，普段の2〜3倍の速さで読みたいとのニーズも報告されている[3]．

そこで筆者らは，電子リーダーにおける表示の工夫によって，文章を読み進める目の動きを効率化すべく，非効率な目の動きを改善する様々な仕組みを検討し，検証を重ねてきた[4–9]．本稿では，目の動きをスムーズにする電子リーダーの設計指針と，現在までに見出された仕組みについて述べる．

2 読みの眼球運動と電子リーダーの課題

読みを効率化する電子リーダーの設計にあたっては，人間の視覚特性に起因する，読書中の眼球運動を考慮することが重要である．

人間の視野は，中心部分でもっとも解像力に優れ，周辺部では低下するという特徴をもつ[10–12]．そして，解像力の高い視野中心部はとても狭く，視野中心から約2.5度離れると，解像力は半分に低下する．細かな文字を識別できるのは，視野中心部に限られる[13]．日本語の場合，その広さは約10〜12文字程度と報告されている[14,15]．したがって，視野中心部に収まらない長さの文字列を読むためには，次々と視点を移動していく必要がある．

視野中心部で文字を認識している注視状態は「停留」と呼ばれ，次の停留点への移動運動は「サッカード」と呼ばれる．図1に示すように，読書中は停留とサッカードが繰り返される．サッカードのうち，一度通り過ぎた場所に戻るサッカードは逆行運動，行末から次行頭へのサッカードは改行***運動と呼ばれる．読書中の眼球運動は，停留，サッカード，逆行運動，改行運動の4要素から構成されている[16–19]．

＊本稿は以下に加筆したものである．
小林潤平：読み効率を高める日本語電子リーダー設計の試み，情報の科学と技術，Vol. 66, No. 10, pp. 525–530 (2016)

＊＊大日本印刷株式会社

1）斎田真也：速読と眼球運動，基礎心理学研究，Vol. 23, No. 1, pp. 64–69 (2004)
2）小林潤平，川嶋稔夫：日本語文章の読み速度の個人差をもたらす眼球運動，映像情報メディア学会誌，Vol. 72, No. 10, pp. J154–J159 (2018)
3）森田愛子：速読は有益か—速読に対するニーズの調査—，読書科学，Vol. 56, No. 3, pp. 113–123 (2015)
4）小林潤平，関口隆，新堀英二，川嶋稔夫：文節間改行レイアウトを有する日本語リーダーの読み効率評価，人工知能学会論文誌，Vol. 30, No. 2, pp. 479–484 (2015)
5）小林潤平，関口隆，新堀英二，川嶋稔夫：文節単位の階段状ベースラインを有する日本語リーダーの可読性，電子情報通信学会論文誌．D, Vol. J99-D, No. 1, pp. 13–22 (2016)
6）小林潤平，関口隆，新堀英二，川嶋稔夫：日本語リーダーにおける読み速度と眼球運動の行長依存性に基づく最適行長の検討，電子情報通信学会論文誌．D, Vol. J99-D, No. 1, pp. 23–34 (2016)
7）小林潤平，関口隆，新堀英二，川嶋稔夫：文節単位を考慮した文字配置の工夫がもたらす日本語電子リーダーの可読性向上，人工知能学会論文誌，Vol. 32, No. 2, pp. A-AI30_1–24 (2017)
8）小林潤平，新堀英二，川嶋稔夫：文字配置と背景着色の工夫がもたらす日本語電子リーダーの読み速度向上，映像情報メディア学会誌，Vol. 71, No. 10, pp. J241–J246 (2017)
9) Kobayashi, J., Shinbori, E., and Kawashima, T.: Stepped and tiered line text layout for improving reading rate in Japanese electronic text readers, in *SID Symposium Digest of Technical Papers,* SID 2019, pp. 410–413 (2019)
10) Wertheim, T.: Über die indirekte Sehschärfe, *Zeitschrift für Psychologie und Physiologie der Sinnesorgane,* Vol. 7, pp. 172–187 (1894)
11) Jones, L. A. and Higgins, G. C.: Photographic granularity and graininess. III. Some characteristics of the visual system of importance in the evaluation of graininess and granularity, *Journal of the Optical Society of America,* Vol. 37, No. 4, pp. 217–263 (1947)
12）苧阪直行：周辺視機能の精神物理学的研究，風間書房 (1983)

＊＊＊本稿では「改行」という用語を「行末での折り返し」の意味に限定して用いる．「段落を新しく始めるために行を改める」という意味ではないことに注意されたい．

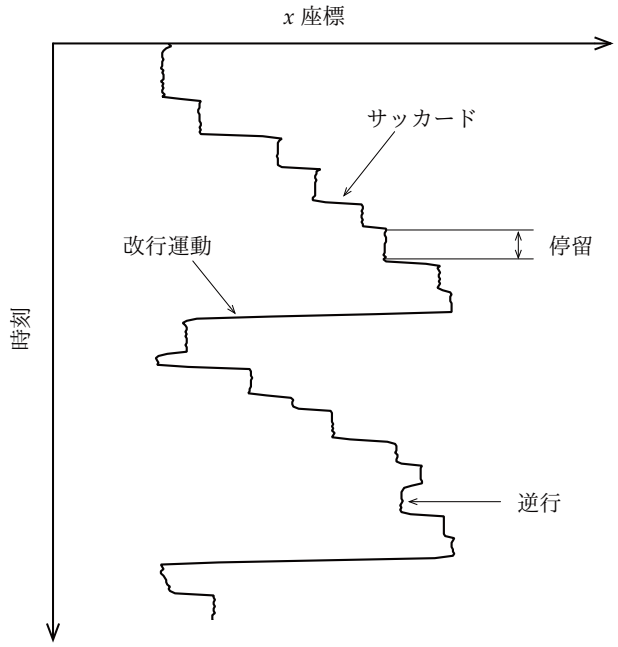


図 1：眼球運動の例　横軸は水平方向の視線位置，縦軸は時間で上から下に経過．停留，サッカード，改行運動，逆行運動が確認できる[6]．

停留中には，視野中心部で文字を認識すると同時に，視野周辺部で次の停留先の選定を行う[20,21]．もっとも短時間で単語を認知できる停留場所は「最適停留位置」と呼ばれる[22–26]．英語やフランス語などのスペースを有する言語の最適停留位置は各単語の中心付近であり，最適停留位置から外れた場所に停留すると，同一単語内で再停留が発生しやすくなる[27–29]．したがって，読み効率の向上においては，最適な停留位置への的確な視点移動が重要となる．

電子リーダーにおける読みの効率向上の実現を目指すにあたって，画面上に表示した文章を柔軟に変更できる電子リーダーの場合は，視点を固定したまま文字側を書き替えて読み進めるなど，読者の目の動きを電子リーダーで代替するような表示設計も可能となる．しかし，読み心地を維持するとなると，なかなか難しいとの結果が報告されてきた．

例えば，視点移動を電子リーダーによる文字表示の切替で代替する仕組みのひとつに，Rapid Serial Visual Presentation（RSVP）と呼ばれる表示方式がある[30]．RSVP は同じ場所にひとつの短文や単語を次々と切り替えながら表示していく方式であり，視点を大きく動かすことなく読み進められるその特徴から，読みの苦手な読者，視野欠損をもつロービジョンの読者，または画面の小さなデバイスに対して有益な表示方式とされる[31–35]．しかし，RSVP は読者が表示切替タイミングを制御することが難しく，自動的に次々と切り替わる短文や単語を読み続けるための，極めて高い集中力が求められ，瞬目ですら読みの妨げとなる．そのために，速く読めたとしても，心地よく感じた読者はほとんどいなかったことが報告されている[36]．

また，視点移動の代替のかわりに，視点移動を誘導する電子リーダーの仕組みも検討されている．例えば，日本語文章の先頭から末尾まで 7 文字ずつ区切ってチャンク化し，先頭のチャンクから順番に 1 回

13）福田忠彦：図形知覚における中心視と周辺視の機能差，テレビジョン学会誌，Vol. 32, No. 6, pp. 492–498 (1978)
14）Ikeda, M. and Saida, S.: Span of Recognition in Reading, *Vision Research,* Vol. 18, No. 1, pp. 83–88 (1978)
15）神部尚武：読みの眼球運動における一つの停留中の情報の受容範囲，国立国語研究所報告，Vol. 96, pp. 59–80 (1989)
16）Rayner, K. and Pollatsek, A.: *The psychology of reading,* Lawrence Erlbaum Associates (1989)
17）苧阪良二，中溝幸夫，古賀一男：眼球運動の実験心理学，名古屋大学出版会 (1993)
18）斎田真也：読みと眼球運動，眼球運動の実験心理学，pp. 167–198，名古屋大学出版会（1993）
19）神部尚武：日本語の読みと眼球運動，読み—脳と心の情報処理，pp. 1–16，朝倉書店（1998）
20）Poulton, E. C.: Peripheral vision, refractoriness and eye movements in fast oral reading, *British Journal of Psychology,* Vol. 53, No. 4, pp. 409–419 (1962)
21）Hochberg, J.: Components of literacy: Speculations and exploratory research, in Levin, H. and Williams, J. P. eds., *Basic Studies on Reading,* chapter 10, pp. 125–138, Basic Books (1970)
22）O'Regan, J. K., Lévy-Schoen, A., Pynte, J., and Brugaillère, B.: Convenient fixation location within isolated words of different length and structure, *Journal of Experimental Psychology: Human Perception and Performance,* Vol. 10, No. 2, pp. 250–257 (1984)
23）McConkie, G. W., Kerr, P. W., Reddix, M. D., Zola, D., and Jacobs, A. M.: Eye movement control during reading: II. Frequency of refixating a word, *Perception and Psychophysics,* Vol. 46, No. 3, pp. 245–253 (1989)
24）Vitu, F., O'Regan, J. K., and Mittau, M.: Optimal landing position in reading isolated words and continuous text, *Perception and Psychophysics,* Vol. 47, No. 6, pp. 583–600 (1990)
25）Nazir, T. A., O'Regan, J. K., and Jacobs, A. M.: On words and their letters, *Bulletin of the Psychonomic Society,* Vol. 29, No. 2, pp. 171–174 (1991)
26）Kajii, N. and Osaka, N.: Optimal viewing position in vertically and horizontally presented Japanese words, *Perception and Psychophysics,* Vol. 62, No. 8, pp. 1634–1644 (2000)
27）McConkie, G. W., Kerr, P. W., Reddix, M. D., Zola, D., and Jacobs, A. M.: Eye movement control during reading: II. Frequency of refixating a word, *Perception and Psychophysics,* Vol. 46, No. 3, pp. 245–253 (1989)
28）O'Regan, J. K.: Optimal viewing position in words and the strategy-tactics theory of eye movements in reading, in Rayner, K. ed., *Eye Movements and Visual Cognition,* pp. 333–354, Springer New York (1992)

ずつ順次弾ませていくことで，弾むチャンクに視線を誘引しながら読ませる表示方式が提案されている[37]．この表示方式では，視線を誘引するマーカーが文章中の文字そのものであり，誘引に従って視点を移していくことで，一定の理解度を維持したまま，読み速度を通常表記の 3.5〜12 倍に向上できた結果が報告されている[37]．しかし，次々と弾んでいくチャンクを目で追い続けなければならず，読者がその時々に読みたい箇所を読むための制御が困難という課題があった．

以上のように，読みを効率化する電子リーダーの仕組みとして，眼球運動の代替や強制的な視線誘導といった仕組みは発展途上にあり，現在の技術ではさらなる改善が難しい状況にあった．

ここで，先に述べたように，停留中には視野中心部で文字を認識すると同時に，解像力の低い視野周辺部で次の停留先を選定する．このとき，視野周辺部ではどのような情報を獲得しているのであろうか．

英語のような単語をスペースで区切りながら表記する言語では，スペースによる単語間の視覚的な境界情報が，視点移動に対して重要な役割を担っているとされる[38–41]．そして，単語間のスペースを除いた場合には，読み速度は 30〜50％低下することが報告されている[42, 43]．

一方，漢字と仮名が混合する日本語文章は，スペースを用いずに表記される．日本語文章の視点移動単位は文節とされるが[44–46]，文節の間にスペースを挿入しても，読みは速くならないことが報告されている[47–50]．漢字と仮名が混合した日本語文章において停留先を決める手掛かりは，解像力の低い視野周辺部において視覚的に目立つ漢字であるために[51–53]，文節間のスペース情報は冗長であると結論付けられた[53]．わざとひらがなを使う，わざと漢字を使うなど，いままでも多くの方々が工夫してきた書記法も，濃い漢字に視線が誘引される性質をふまえたものであったと捉えることができる．

しかし，既存の日本語組版が完璧なわけではない．日本語文章における平均停留時間および平均サッカード距離は，それぞれ約 0.25 秒および約 5 文字である[54–56]．もしスムーズに停留とサッカードを繰り返すことができれば，計算上は 1 分あたり 1,200 文字の速さで読むことができる．しかし，日本語の平均的な読み速度は 1 分あたり 500〜600 文字であり[57, 58]，計算上の速さと実際の平均的な速さには，大きな差がある．

この差が生じる要因は，最適な停留場所に的確にサッカードできないことに起因する，非効率な視点移動にあるとされる[59–61]．すなわち，日本語文章においては，視覚的に目立つ漢字への視線誘引を前提としながらも，各文節への的確な目の動きを促すための新たな表示方式が必要とされていた．

3 日本語電子リーダーの設計と実験

そこで筆者らは，日本語文章において，文節の境界情報として機能するような新たな仕組みを導入し，読み心地や理解度を維持したまま，文節単位のスムーズな視点移動を促す電子リーダーの実現を目指した．定量指標と改善指針は，表 1 の通りである．

29) Vitu, F., O'Regan, J. K., Inhoff, A. W., and Topolski, R.: Mindless reading: Eye-movement characteristics are similar in scanning letter strings and reading texts, *Perception and Psychophysics,* Vol. 57, No. 3, pp. 352–364 (1995)

30) Forster, K. I.: Visual perception of rapidly presented word sequences of varying complexity, *Perception and Psychophysics,* Vol. 8, No. 4, pp. 215–221 (1970)

31) Chen, H.-C.: Effects of reading span and textual coherence on rapid-sequential reading, *Memory and Cognition,* Vol. 14, No. 3, pp. 202–208 (1986)

32) Williamson, N. L., Muter, P., and Kruk, R. S.: Computerized Presentation of Text for The Visually Handicapped, *Advances in Psychology,* Vol. 34, pp. 115–125 (1986)

33) Rubin, G. S. and Turano, K.: Low vision reading with sequential word presentation, *Vision Research,* Vol. 34, No. 13, pp. 1723–1733 (1994)

34) Castelhano, M. S. and Muter, P.: Optimizing the reading of electronic text using rapid serial visual presentation, *Behaviour and Information Technology,* Vol. 20, No. 4, pp. 237–247 (2001)

35) Legge, G. E.: *Psychophysics of reading in normal and low vision,* Lawrence Erlbaum Associates Publishers (2007)

36) Rubin, G. S. and Turano, K.: Reading without saccadic eye movements, *Vision Research,* Vol. 32, No. 5, pp. 895–902 (1992)

37) Kawashima, T., Terashima, T., Nagasaki, T., and Toda, M.: Enhancing visual perception using dynamic updating of display, in Grieser, G. and Tanaka, Y. eds., *Intuitive Human Interfaces for Organizing and Accessing Intel lectual Assets,* Vol. 3359 of *Lecture Notes in Computer Science,* pp. 127–141, Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

38) McConkie, G. W. and Rayner, K.: The span of the effective stimulus during a fixation in reading, *Perception and Psychophysics,* Vol. 17, No. 6, pp. 578–586 (1975)

39) Pollatsek, A. and Rayner, K.: Eye movement control in reading: The role of word boundaries, *Journal of Experimental Psychology: Human Perception and Performance,* Vol. 8, No. 6, pp. 817–833 (1982)

40) Morris, R. K., Rayner, K., and Pollatsek, A.: Eye movement guidance in reading: the role of parafoveal letter and space information., *Journal of experimental psychology. Human perception and performance,* Vol. 16, No. 2, pp. 268–281 (1990)

41) Rayner, K., Fischer, M. H., and Pollatsek, A.: Unspaced text interferes with both word identification and eye movement control, *Vision Research,* Vol. 38, No. 8, pp. 1129–1144 (1998)

42) Rayner, K. and Pollatsek, A.: Reading unspaced text is not easy: Comments on the implications of Epelboim et al.'s (1994) study for models of eye movement control in reading, *Vision Research,* Vol. 36, No. 3, pp. 461–465 (1996)

表 1：日本語電子リーダーの設計指針

指 標		指 針
読み速度		増加
眼球運動	– 停留	削減
	– 停留時間	短縮
	– 順行サッカード長	伸長
	– 逆行による過剰停留	削減
	– 改行運動中の過剰停留	削減
	– 1 文節あたりの停留数	1 回以下
文章内容の理解度		維持
読み心地		維持

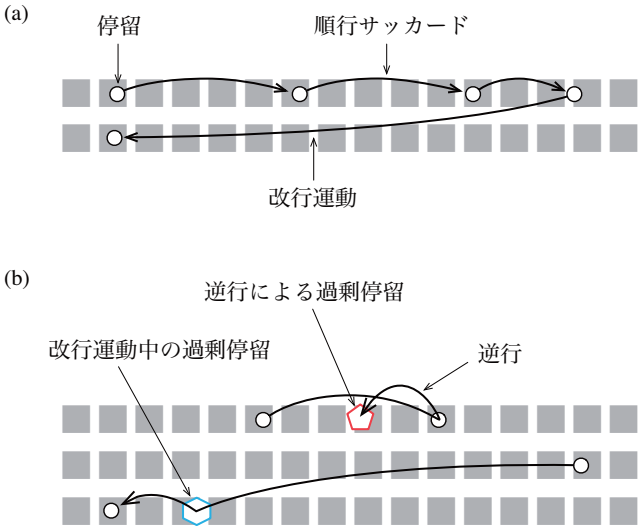


図 2：文章を読む眼球運動の模式図　(a) 停留と順行サッカードを繰り返し，改行運動によって行末から次行頭へ移る．(b) 過剰停留の定義．逆行によって発生した停留を「逆行による過剰停留」，改行時に行頭へ 1 回のサッカードで到達できず追加で発生した停留を「改行運動中の過剰停留」と定義．順行サッカード長を伸ばすこと，2 種類の過剰停留を減らすことが，電子リーダー設計の基本方針である．

表 1 の設計指針に基づき，図 2 に示す眼球運動を詳細に分析しながら，順行サッカード長を伸ばすように，2 種類の過剰停留を減らすようにと，文字の並べ方や動き，文字色や背景色，文字サイズやフォントなど，様々なパラメータをひとつずつ変更できる実験装置を組み，評価検証を重ねた．

文章を読む実験協力者の目の動きは，nac 社製の視線検出装置 EMR-9 を用いて 1/60 秒間隔で計測し，停留の場所やサッカード長，過剰停留の変化を分析した．

評価用の文章は，1 話の文字数が 2,000 字程度の星新一氏のショートショート作品を用いた．実験協力者あたり 1 話 1 回のみの閲覧に制限するとともに，読む文章や読む順番を含む実験条件の組み合わせが実験協力者間で重複しないように，あらかじめ調整した．

理解度が低下していないことを確認するために，文章を読み終えた直後に，内容に関して質問した．質問の難易度は，熟読したり暗記しなくとも，飛ばし読みをせずに読めば答えられるように設計した．読後の質問に答えられなかった場合の計測データは棄却し，文章を変更して再計測した．

43) Rayner, K., Fischer, M. H., and Pollatsek, A.: Unspaced text interferes with both word identification and eye movement control, *Vision Research,* Vol. 38, No. 8, pp. 1129–1144 (1998)

44) 中條和光：「読み」の認知モデル：日本語文章の読みに関する実験的研究，協同出版 (1999)

45) 神部尚武：読みの眼球運動と読みの過程，国立国語研究所報告，Vol. 85, pp. 29–66 (1986)

46) 神部尚武：日本語の読みと眼球運動，読み—脳と心の情報処理，pp. 1–16，朝倉書店 (1998)

47) 御領謙：付録，読むということ，pp. 155–174，東京大学出版会 (1987)

48) 松田真幸：日本語文の読みに及ぼす文節間空白の影響，基礎心理学研究，Vol. 19, No. 2, pp. 83–92 (2001)

49) 藤木大介：日本語文の読みにおける分節単位の検討，広島大学心理学研究，Vol. 2, pp. 21–27 (2002)

50) Sainio, M., Hyönä, J., Bingushi, K., and Bertram, R.: The role of interword spacing in reading Japanese: an eye movement study, *Vision Research,* Vol. 47, No. 20, pp. 2575–84 (2007)

51) Osaka, N.: Effect of peripheral visual field size upon eye movements dllring Japanese text processing, in O'Regan, J. and Lévy-Schoen, A. eds., *Eye Movements from Physiology to Cognition,* pp. 421–429, Elsevier (1985)

52) Osaka, N.: Eye fixation and saccade during kana and kanji text reading: comparison of English and Japanese text processing, *Bulletin of the Psychonomic Society,* Vol. 27, No. 6, pp. 548–550 (1989)

53) Sainio, M., Hyönä, J., Bingushi, K., and Bertram, R.: The role of interword spacing in reading Japanese: an eye movement study, *Vision Research,* Vol. 47, No. 20, pp. 2575–84 (2007)

54) 神部尚武：読みの眼球運動と読みの過程，国立国語研究所報告，Vol. 85, pp. 29–66 (1986)

55) 斎田真也：読みと眼球運動，眼球運動の実験心理学，pp. 167–198，名古屋大学出版会 (1993)

56) 神部尚武：日本語の読みと眼球運動，読み—脳と心の情報処理，pp. 1–16，朝倉書店 (1998)

57) 斎田真也：速読と眼球運動，基礎心理学研究，Vol. 23, No. 1, pp. 64–69 (2004)

58) 小林潤平，川嶋稔夫：日本語文章の読み速度の個人差をもたらす眼球運動，映像情報メディア学会誌，Vol. 72, No. 10, pp. J154–J159 (2018)

59) 山本三吾：讀書と眼球運動に就ての一實驗，心理学研究，Vol. 10, No. 5–6, pp. 773–787 (1935)

60) 斎田真也：速読と眼球運動，基礎心理学研究，Vol. 23, No. 1, pp. 64–69 (2004)

4　見出された方式と知見

研究を始めた頃は，文字の並びを大胆に変更する方式や，表示を高速で切り替える方式など，現在の一般的な日本語書記法から大きく離れた表示方式を検討し，検証を重ねた．しかし，残念なことに，そのような表示方式は，読みの改善につなげることができなかった．読み能力は幼少期の訓練の賜物であり[62]，培った能力が容易に流用できる表示方式でないと，即時の改善にはつながらないことが予想された．その結果，検討する仕組みは，現在の書記法に近付くとともに，どんどん細かなものとなっていった．

見出された特徴的な表示方式のひとつは，「改行位置を文節間に設定し，さらに文節ごとに文字ベースラインを階段状に下げていく表示方式（階段状ベースライン方式）」である．図 3–(a) に示すように，隣り合う文節を上下にずらすことで視覚的な境界情報を付与し，文節単位の視認性を高める効果を狙ったものである．現在の日本語レイアウトでは一直線となるように設定する文字ベースラインを，本レイアウトでは文節単位で段階状にずらしていく．一段の下げ幅は文字高の 7.5%分と微小である．

もうひとつの特徴的な表示方式は，「改行位置を文節間に設定し，さらに文節ごとに異なる位相で文字自体を微振動させる表示方式（微振動テキスト方式）」である．図 3–(b) に示すように，文節単位の微振動による疎密によって，文節間の境界情報を動的に形成することで，文節単位の視点移動を促す効果を狙ったものである．文章の先頭から順に，文節を 4 グループに分けて，0.25 秒ずつタイミングをずらして振動させる．微振動の振幅は文字幅の 3%分と，こちらも微小である．

これら方式の定量的な効果を示したのが，図 4 である．停留数の減

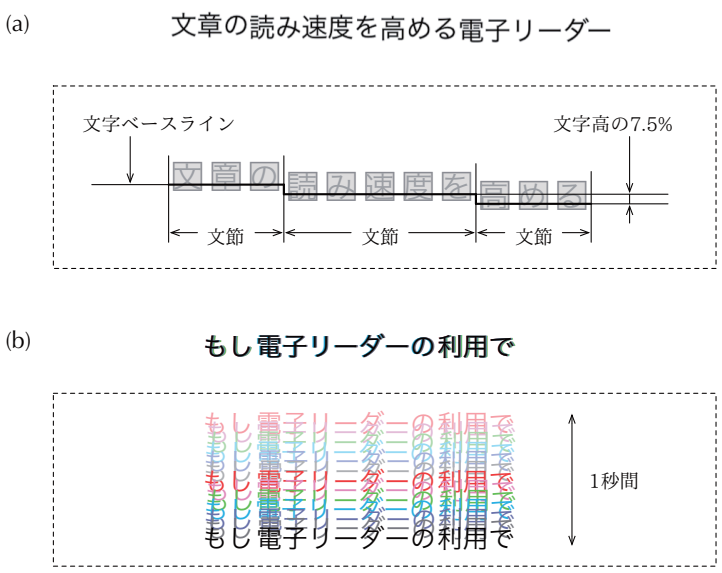


図 3：本研究で見出された 2 つの表示方式　(a) 改行位置を文節間に設定し，さらに文字ベースラインを文節ごとに階段状に下げていく表示方式[63]．(b) 改行位置を文節間に設定し，さらに文節ごとに異なる位相で文字そのものを微振動させる表示方式．図は 1 秒間の文字移動軌跡と振動パターンを示している[64]．

61) Kawashima, T., Terashima, T., Nagasaki, T., and Toda, M.: Enhancing visual perception using dynamic updating of display, in Grieser, G. and Tanaka, Y. eds., *Intuitive Human Interfaces for Organizing and Accessing Intel lectual Assets,* Vol. 3359 of *Lecture Notes in Computer Science,* pp. 127–141, Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

62) Wolf, M.：プルーストとイカ：読書は脳をどのように変えるのか？，インターシフト (2008)，小松淳子（翻訳）

63) 小林潤平，関口隆，新堀英二，川嶋稔夫：文節単位の階段状ベースラインを有する日本語リーダーの可読性，電子情報通信学会論文誌 . D, Vol. J99-D, No. 1, pp. 13–22 (2016)

64) 小林潤平，関口隆，新堀英二，川嶋稔夫：文節単位を考慮した文字配置の工夫がもたらす日本語電子リーダーの可読性向上，人工知能学会論文誌，Vol. 32, No. 2, pp. A-AI30_1–24 (2017)

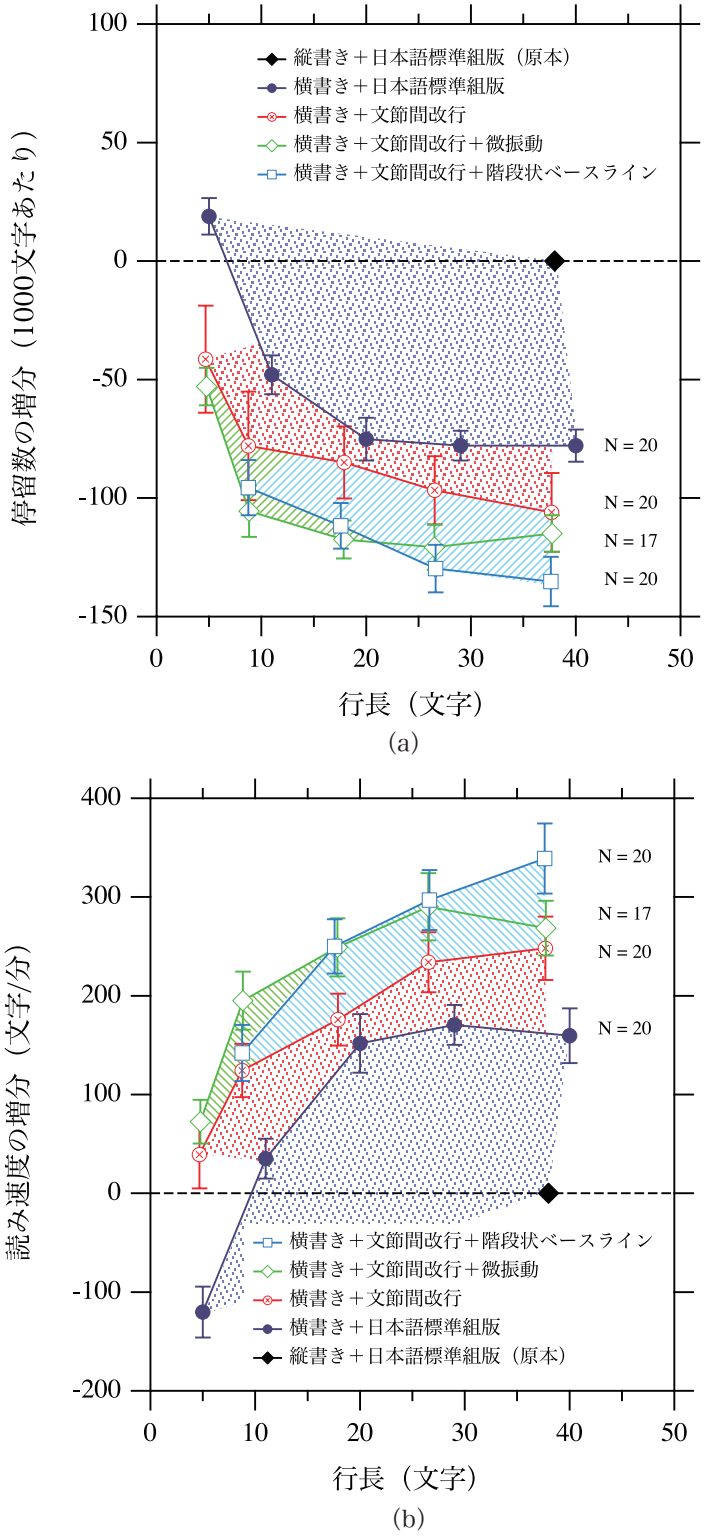


図 4：各方式における停留数と読み速度の変化　(a) 停留数の減分，(b) 読み速度の増分，誤差範囲は標準誤差．

分が図 4–(a)，読み速度の増分が図 4–(b)，誤差範囲は標準誤差である．図 4 を参照しながら，現在までにわかってきたことを述べる．

読み速度の向上は，停留数の削減で実現

読み速度は，停留数と平均停留時間によって決定される．すなわち，読み速度を向上させるためには，停留数を減らすか，停留時間を

減らすか，または両方を減らすかの三択となる．

現在までに見出した読み速度の向上を促す表示方式は，停留時間の短縮ではなく，停留数の削減によって実現していることがわかった．平均停留時間については，現在までに検証した方式において，有意な差が認められないか，おおむね同一水準にあった．停留時間を削減する表示方式については，今後の検討課題のひとつである．

短行ほど，読み速度は低下する

一行の長さが短くなると，停留数は増加し，読み速度は低下する結果となった．

行長に関しては，その最適な長さについて古くから議論されてきた[65–73]．様々な研究の結果，行長については，長過ぎると「行末から行頭に至る正確な視線移動が困難になる」一方で，短過ぎると「一目で受容可能な情報量に満たず充分な読み能力を発揮できない」というトレードオフの関係が存在し，それらの妥協点が最適とされてきた[74, 75]．

筆者らの実験結果では，行長は長いほど，速く読める結果となった[76]．ただし，一行 20 文字から 40 文字の範囲では，概ね同等の読み速度を示すことがわかった．また，読者がもっとも読みやすいと感じた行長は，一行 25 文字前後であった．これらの結果から，読みに最適な行長は一行 20〜30 文字程度，と結論付けた[76]．

横書きの方が，速く読める

縦書きと横書きの読みについては，慣れの影響が大きいことが指摘されている[77, 78]．古くより，縦書きと横書きどちらにも慣れている場合には，読み速度や理解度は同等との結果が報告されてきた．半世紀前には，縦書きの文章が多かった．1960 年に報告された永野らの実験では，縦書きの方が横書きよりも速く読まれており，その結果について「現状（1960 年当時）における読書環境では，縦書きのものが横書きよりもはるかに多いことから，慣れの度合いが反映されたもの」と推察されている[78]．現代でも，横書きよりも縦書きの読みに慣れた人は，縦書きの方が読みやすいと感じることになる．

一方，現代は横書きの文章も多い．大学生 12 名の小規模な予備実験でも，縦書きの方が速い学生はゼロであり，横書きの方が縦書きよりも平均 10%程度速く読まれていた．図 4–(b) においても，縦書き文章の読み速度を基準にとると，一行 20〜40 文字の範囲において，横書き文章の方が 1 分あたり 150 文字以上も速いことがわかる*．

人体構造の点から検討すると，横組みの方が読みやすい可能性が高いと思われる．人間の場合，頭を動かさずに目だけ動かして見える範囲は，上下よりも左右に広い[79]．すなわち，眼球運動的に無理なく読める範囲は，横組みの方が広いことを意味する．

文節間で改行**すると，速く読める

改行位置を文節間に設定すると，速く読めることがわかった[80]，眼球運動分析より，文節を分断しないように行の折り返し位置を調整すると，視点移動の効率化につながることがわかった[81]．これによって

65）Huey, E. B.: *The psychology and pedagogy of reading; with a review of the history of reading and writing and of methods, texts, and hygiene in reading,* Macmillan (1908)

66）Tinker, M. A. and Paterson, D. G.: Studies of typographical factors influencing speed of reading: III. Length of line, *Journal of Applied Psychology,* Vol. 13, No. 3, pp. 205–219 (1929)

67）田中廣吉：言語及讀方の基本的研究，目黒書店（1916）

68）Duchnicky, R. L. and Kolers, P. A.: Readability of text scrolled on visual display terminals as a function of window size, *Human Factors: The Journal of the Human Factors and Ergonomics Society,* Vol. 25, No. 6, pp. 683–692 (1983)

69）草島時介，村石昭三：文字配列の合理化に関する実験的研究，Technical report，国立国語研究所（1954）

70）Dyson, M. C. and Kipping, G. J.: The effects of line length and method of movement on patterns of reading from screen, *Visible Language,* Vol. 32, No. 2, pp. 150–181 (1998)

71）Dyson, M. C. and Haselgrove, M.: The influence of reading speed and line length on the effectiveness of reading from screen, *International Journal of Human-Computer Studies,* Vol. 54, No. 4, pp. 585–612 (2001)

72）Beymer, D., Russell, D. M., and Orton, P. Z.: Wide vs. narrow paragraphs: An eye tracking analysis, in Costabile, M. F. and Patern`o, F. eds., *Human-Computer Interaction - INTERACT 2005,* Vol. 3585 of *Lecture Notes in Computer Science,* pp. 741–752, Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

73）石井亮登，森田ひろみ：縦スクロール表示された文章の快適な読み速度と眼球運動，情報処理学会論文誌，Vol. 54, No. 6, pp. 1784–1793 (2013)

74）Rayner, K. and Pollatsek, A.: *The psychology of reading,* Lawrence Erlbaum Associates (1989)

75）Rayner, K.: The work of the eyes, in Rayner, K., Pollatsek, A., Ashby, J., and Clifton, Jr., C. eds., *Psychology of Reading,* pp. 91–134, Psychology Press (2012)

76）小林潤平，関口隆，新堀英二，川嶋稔夫：日本語リーダーにおける読み速度と眼球運動の行長依存性に基づく最適行長の検討，電子情報通信学会論文誌 . D, Vol. J99-D, No. 1, pp. 23–34 (2016)

＊正確には，図 4 の実験結果は，縦書き文章は「ページ型」で表示し，横書き文章は「縦スクロール型」で表示したときの読み速度であることに注意されたい．ただし，一行 20〜40 文字で表示されている場合は，スクロール型であってもスクロール操作時間は相対的に短く，読み動作は概ね同等と言える．

＊＊先述の通り，ここでの改行は「行末での折り返し」を意味しており，「段落を新しく始めるために行を改める」という意味ではないことに注意されたい．

全体の停留数が減り，読み速度の向上をもたらしたことがわかった．

長行に効く「文節単位の階段状ベースライン」

文節間での改行と，文節単位の階段状ベースラインを組み合わせた表示方式では，縦書きの原本よりも約 1.5 倍の速さで読むことができた．このとき，100％の実験協力者が理解度を維持し，91％の実験協力者が読み心地の低下を感じることなく読めていた．

この方式では，順行サッカードの伸長と逆行の減少が確認され[82]，一文節あたりの停留数も 1 回に近付くことがわかった[83]．隣り合う文節が上下にずれて配置されるレイアウト上の視覚的特徴が，次の停留先を選定する視覚処理系に正の影響を与えている可能性が推察された．一方で，短い行長では充分な効果が得られないこともわかった．

短行に効く「文節単位の微振動テキスト」

先に述べたように，日本語の読み速度は行長が短くなるほど低下する．この読み速度の低下は，行長が短くなるほど，順行サッカード長 (図 2) が短くなることに起因する．この順行サッカードの短縮化は，階段状ベースライン方式でも改善できなかったため，さらに強く文節単位を視知覚できる仕組みを検討した．

一行の文字数が 10 文字程度の短い行長では，文節間での改行と，文字の微振動を組み合わせた方式で表示すると，最も速く読めることがわかった．微振動テキスト方式は，実験協力者全員がはじめて体験した表示方式であったが，100％の実験協力者が問題なく読み進め，100％の実験協力者が理解度を維持し，76％の実験協力者が読み心地の低下を感じることなく読めていた．

文節単位の微振動テキスト方式は，一行 10 文字程度の短い行長において，文節単位の視点移動を促す効果をもつことがわかった．微振動する文節単位のまとまりや，動的な疎密変化による境界情報が，視点移動の新たな手がかりとして機能した可能性が推察された[84]．

5 おわりに

文章を読み進める目の動きを効率化すべく，様々な仕組みを検討し，検証を重ねた．その結果，文節にもとづく改行位置の調整や，文節単位で文字ベースラインを階段状にずらす表示方式，文節単位で左右に微振動させる表示方式が，文節単位でのスムーズな視点移動を促すことがわかった．そして，それらの仕組みを電子リーダーに組み込むことで，読み速度を向上できることがわかった．

日本語の組版は，まだまだ多くの可能性を残している．もっとスムーズに読めて，そしてもっとよく理解できるような表示方式を見出すべく，今後も研究していきたい．

公立はこだて未来大学 松原 仁 教授に機材の便宜をお図りいただくとともに，公立はこだて未来大学学生の方々に多大なご協力をいただいた．ここに感謝の意を表する．

77）田中廣吉：言語及讀方の基本的研究，目黒書店（1916）

78）永野賢，林四郎，渡辺友左：新聞の文章のわかりやすさに関する調査研究，国立国語研究所年報，Vol. 11, pp. 76–130 (1960)

79）渡部叡 , 坂田晴夫 , 長谷川敬 , 吉田辰夫 , 畑田豊彦：視覚の科学，写真工業（1975）

80）小林潤平，関口隆，新堀英二，川嶋稔夫：文節間改行レイアウトを有する日本語リーダーの読み効率評価，人工知能学 会論文誌，Vol. 30, No. 2, pp. 479–484 (2015)

81）小林潤平，関口隆，新堀英二，川嶋稔夫：文節単位を考慮した文字配置の工夫がもたらす日本語電子リーダーの可読性向上，人工知能学会論文誌，Vol. 32, No. 2, pp. A-AI30_1–24 (2017)

82）小林潤平，関口隆，新堀英二，川嶋稔夫：日本語リーダーにおける読み速度と眼球運動の行長依存性に基づく最適行長の検討，電子情報通信学会論文誌 . D, Vol. J99-D, No. 1, pp. 23–34 (2016)

83）Kobayashi, J., Sekiguchi, T., Shinbori, E., and Kawashima, T.: Stepped-line text layout with phrased segmentation for readability improvement of Japanese electronic text, in *IEEE International Symposium on Multimedia (ISM 2015),* pp. 553–558 (2015)

84）小林潤平，関口隆，新堀英二，川嶋稔夫：文節単位を考慮した文字配置の工夫がもたらす日本語電子リーダーの可読性向上，人工知能学会論文誌，Vol. 32, No. 2, pp. A-AI30_1–24 (2017)

組版についてのアクセシビリティ要件
Accessibility Requirements on Japanese Typography

村田　真

この文書は，組版についての要件のうちアクセシビリティに関係するものを整理するためのものである．DAISY 教科書を実践している現場からの声を重視して書かれている．

This document summarizes those requirements on Japanese typography which are relevant to accessibility. Most of them come from practitioners of DAISY textbooks in Japan.

We provide English translation so as to solicit feedbacks from Mainland China and Taiwan and to encourage further works in W3.

1　はじめに（Introduction）

どんな日本語組版が健常者にとって読みやすいか論じられることはあるが，どんな日本語組版がアクセシブルか議論されることはまずない．文字を拡大できる電子書籍のほうが中高年には読みやすいと言われる程度である．

一方，アクセシビリティ関係者は，普通の紙の本を読むことができない人たちと向き合ってきた．紙の本が読めない・読み難い状況にはいろんなものがあり，その原因も多岐にわたる．紙では不可能なさまざまの対策が電子書籍では可能になる（紙の本が読めない・読み難い状況とその原因を参照）．

紙の本が読めない・読み難い状況とその原因
https://github.com/Advanced Publishing-Laboratory/A11Y/wiki/紙の本が読めない-読み難い状況とその原因

対策の多くは，W3C の Web Content Accessibility Guidelines (WCAG) 2.0 と User Agent Accessibility Guidelines (UAAG) 2.0 に示されている．それらのうち，文字組版に関するものを抜粋する．

- 文字の大きさを変更する
- フォントを変える
- 行間，字間，単語間を変更する
- 行端揃え，余白のサイズ，ボーダーを設定する
- 背景色と文字色を変更する

これらは日本語に特化したものではなく，ほとんどの言語にあてはまる．逆にいえば，Web Content Accessibility Guidelines (WCAG) 2.0 と User Agent Accessibility Guidelines (UAAG) 2.0 は日本語固有の対策には踏み込んでいない．

いっぽう，日本におけるデイジー教科書は日本語組版に固有なアクセシビリティ対策にも踏み込んできた．この文書は，それらをまとめたものである．

2 定義 (Definitions)

ルビ ruby

親文字 base character

総ルビ general-ruby

パラルビ para-ruby

文節 bunsetsu　文を句切りながら発音して，実際の言語としてはそれ以上に句切ることはない個々の部分（橋本進吉）

Read aloud a Japanese sentence. Pause as much as possible without being unnatural. Units delimited by such pause are bunsetus.

備考：文節はひとつ以上の語からなり．文はひとつ以上の文節からなる．「私は」という文節は，「私」という語と「は」という助詞からなる．「私は学生です」という文は，「私は」という文節を含む．

Note: A bunsetsu consists of one or more words, while a sentence consists of one or more bunsetsus. A bunsetsu 私は is a word 私 (a subject pronoun) followed by は (a particle), while a sentence 私は学生です comprises this bunsetsu.

分かち書き wakachi-gaki　文節ごとに間隔を空ける表記

rendering of Japanese where bunsetsus are separated by space

備考：分かち書きにはいろんな流儀がある．光村図書の流儀では，学年ごとに変えている．点字では音節数を考慮して分かち書きをする．国立国語研究所の「文節の仕様について」もある．

ルビ
https://www.w3.org/TR/jlreq/ja/#term.ruby
ruby
https://www.w3.org/TR/jlreq/#term.ruby
親文字
https://www.w3.org/TR/jlreq/ja/#term.base-characters
base character
https://www.w3.org/TR/jlreq/#term.base-characters
総ルビ
https://www.w3.org/TR/jlreq/ja/#term.general-ruby
general-ruby
https://www.w3.org/TR/jlreq/#term.general-ruby
パラルビ
https://www.w3.org/TR/jlreq/ja/#term.para-ruby
para-ruby
https://www.w3.org/TR/jlreq/#term.para-ruby
文節 bunsetsu
https://ja.wikipedia.org/wiki/文節
光村図書の流儀
http://www.mitsumura-tosho.co.jp/webmaga/kotoba/detail01.html
文節の仕様について
http://pj.ninjal.ac.jp/corpus_center/csj/manu-f/bunsetsu.pdf

3 ルビの表示に関する要件 (Requirements on ruby rendering)

1）総ルビが必要（general ruby is needed）

理由（Reasons）

- 漢字がほとんど読めない子がいる

（Some students cannot read CJK ideographic characters well.）

数値的な根拠（もしあれば）（Evidence if any）

- 61%の児童が総ルビを要求する（2017 年度の DAISY 教科書一

般申請利用者へのアンケート結果）
（61% students of DAISY text users require general ruby）

2）原本通りのパラルビが必要
（para-ruby as in the paper version is needed）

理由（Reasons）
- だいたい読めるが該当学年の漢字は読めない
（Some students cannot read CJK ideographic characters for their grade.but can read easy ones.）
- 総ルビで何回か読んだあとは難しい漢字だけのルビでいい
（Once you read general-ruby textbooks a few times, para-ruby is good enough.）
- 紙の教科書はパラルビなので，それを忠実に再現する
（Faithfully mimick printed textbooks, which has para-ruby only.）

数値的な根拠（もしあれば）（Evidence if any）
- 32%の児童がこれで十分としている（2017 年度の DAISY 教科書一般申請利用者へのアンケート結果）
（61% students of DAISY text users feel para-ruby is good enough）

3）ルビ無し表示が必要（rendering without ruby is needed）

理由（Reasons）
- ルビがあると親文字が読みにくくなる
（ruby makes base characters difficult to read）
- ルビがなくても十分読める（ようになった）生徒
（some students can read text without ruby）

数値的な根拠（もしあれば）（Evidence if any）
- 総ルビでも原本通りのパラルビでもないものを要求する児童が7%いるが詳細は不明（2017 年度の DAISY 教科書一般申請利用者へのアンケート結果）
（7% of the students requiring DAISY textbooks require ruby different from para-ruby or general ruby.）

4）同一コンテンツを総ルビ／パラルビ／ルビ無しと表示し分けられることが必要
（the same content should be renderable as general ruby, para-ruby, or no ruby）

理由（Reasons）
- 製作コスト（authoring cost）
- 配布コスト（distribution cost）
- 利用者の管理コスト（management cost by students）

数値的な根拠（Evidence if any）
- 現在は，総ルビとパラルビの二種類を作っているため，小学校 4 教科 6 学年分で 16 GB から 20 GB 必要になっている．これが半

分になる．

- 現在の学校の通信環境だと，総ルビとパラルビの二種類を 4 教科分，ダウンロードするのに，4 教科で 1 時間 47 分から 2 時間 13 分必要とする．これが半分になる．

5）指定した学年以降で習う漢字だけルビを表示する

（ruby should be visible only when it is beyond the specified grade）

この方法を採用している実装（ChattyBooks）がある．

（An implementation, ChattyBooks, adopts this approach.）

ChattyBooks
http://www.sciaccess.net/jp/ChattyBooks/

理由（Reasons）

- だいたい読めるが該当学年の漢字は読めない
  （Some students can read low-grade kanjis but cannot read kanjis beyond their grades.）
- 総ルビで何回か読んだあとは難しい漢字だけのルビでいい

6）ルビと親文字がはっきり区別できるような表示が必要

（the base character and ruby should be easily distinguishable）

理由

- ディスレクシアのうちの一部の人には，ルビが草かんむりのお化けに見える．ルビの色を変える等の表示の工夫をすることで，ルビと親文字を別の文字と認識できる場合がある
  （To some dyslexia people, ruby looks like a strange variation of the “grass” radical. Use of a different colour for ruby helps such people to separate ruby and base characters.）

7）両側ルビはそれほど必要ではない

（no strong requirements on double-sided ruby）

紙の教科書には存在するが，現在の DAISY 教科書には存在していない．括弧書きなどで代用している．

（double-sided ruby exists in printed textbooks, but does not exist in current DAISY extbooks. Parenthesized expressions are used instead.）

Media overlay 再生のとき，どのように読み上げとハイライト表示を行うか？

4 分かち書きに関する要件 (Requirements on wakachi-gaki)

1）分かち書きが必要

（wakachi-gaki rendering is needed）

理由（Reasons）

- 分かち書きがないと語の区切りが分からない子がいる
  （Some students cannot tell bunesetsu boundaries without

wakachi-gaki rendering)

- 印刷された（分かち書きではない）文章にスラッシュを書き込んでいる人は多い.
  (People add the slash character as bunsetsu boundaries on printed (non-wakachi-gaki) text.)

備考：DAISY 教科書では，分かち書きはなされていないものが多いが，分かち書きされているものが一部ある.
(Most DAISY textbooks do not provide wakachi-gaki rendering, but some do.)

2) 分かち書きのない表示が必要
(non-wakachi-gaki rendering is needed)

理由（Reasons）

- 分かち書きがないほうが教科書の表示に近い
  (Non-wakachi-gaki rendering is close to printed textbooks and printed matters in general.)
- 分かち書きがなくても十分読める（ようになった）児童
  (Some students once needed wakachi-gaki rendering will eventurally find it unecessary.)

3) 同一コンテンツを分かち書き／分かち書き無しと表示し分けられることが必要
(The same content should be renderable with or without wakachi-gaki)

理由（Reasons）

- 製作コスト（Authoring cost）
- 配布コスト（Distribution cost）
- 利用者の管理コスト（Management burden on students）

数値的な根拠

- 総ルビとパラルビの二種類を作る場合と同様

4) 人手による正確な分かち書き
(Precise wakachi-gaki specified by humans)

理由（reason）

- 義務教育段階では正確な分かち書きが求められる
  (Textbooks for compulsory education are required to correctly separate bensetsus for wakachi-gaki.)

備考：形態素解析による分かち書きだと完全な正確性はない.
(Wakachi-gaki by morphological analysis is not perfectly reliable.)
低学年になるほど，漢字が少ないので，プログラムによる分かち書きは難しくなる.
(Lower-grade textbook are hard for morphological analysis and thus automated wakachi-gaki.)

5 行の分割に関する要件 (Requirements on line-breaking)

1）文節単位での行の分割が必要

(Line breaking between bunsetsus only)

理由（reason）

- 語を認識することに困難を抱える児童にとって，文節途中での行の分割は理解の妨げとなる．教科書では文節単位で行を分割するようにおおむねなっている．

  (Those students who cannot recognize bunsetsus easily are confused by line breaks within bunsetsus. Some textbooks allow line breaking between bunsetus only.)

2）文節単位でない行の分割

(Line breaking almost anywhere)

一般の印刷物では，文節単位ではない行の分割を行っている．

6 その他

いくつかの対策が提案されているが，広く認められているわけではない．

- 文節ごとに文字のベースラインを変える
- 縦書きのものを横書きで表示する，およびその逆

ルビの簡便な配置ルール（案）

小林　敏

0 はじめに

ルビ処理のむつかしさ

ルビの組版処理は，次のような事項を考慮して，その配置位置を決める必要がある．

(1) 親文字とルビの対応をどう処理すればよいのか
(2) 親文字の文字列の全長に比べ，ルビ文字の文字列の全長が短い場合，どう処理すればよいのか
(3) 親文字の文字列の全長に比べ，ルビ文字の文字列の全長が長い場合，どう処理すればよいのか，親文字の文字列からルビ文字の文字列がはみ出してよいのか
(4) 親文字の文字列からルビ文字の文字列がはみ出した場合，前後に配置する文字や約物に掛かってよいのか，また，このことは，親文字とルビとの配置位置に影響するのか
(5) 親文字の文字列からルビ文字の文字列がはみ出した場合において，行の先頭又は末尾に配置するときは，親文字の文字列の先頭又は末尾を行の先頭又は末尾にそろえるのか，それともルビ文字の文字列の先頭又は末尾を行の先頭又は末尾にそろえるのか
(6) 親文字が複数の場合，2行にわたる分割をしてよいのか

活字組版では，こうした場合，原則的な考え方に従って処理し，問題があれば校正段階で修正の赤字が入り，それに従って直していた．いってみれば，個別箇所で臨機応変の工夫をして配置位置を決めていた．コンピュータ組版では，ある程度は，配置方針を決め，一定のルールに従って配置処理をしていたが，個別箇所で，親文字とルビ文字の対応を変える，あるいは配置処理方針を変えるなどの工夫も必要とした．

Web での配置処理

Web での配置処理を考えた場合，活字組版のように個別箇所で配置位置を工夫する処理は避けるのが望ましい．となると，上記の問題をすべて解決できる配置処理方針を決め，処理系に実装していく必要がある．活字組版で理想とされていた処理方法をカバーするとなると，かなり複雑な処理方針を考える必要がある．

しかし，ある程度理想的な配置処理を考えたとしても，ルビ処理では，どうしても例外事項が発生し，問題がでる可能性がある．

親文字からのルビのはみ出し　活字組版では，一般に親文字からはみ出したルビは，漢字には掛けないが，仮名には掛けてよいとする場合が多かった．しかし，ルビが片仮名の場合は，親文字の前後の仮名にも掛けないという考え方をする出版社もあった．また，親文字の前後の一方が漢字で，他方が仮名の場合，漢字には，はみ出したルビを掛けないで字間を空けるが，仮名の方もバランスをとって，はみ出したルビを掛けないで字間を空ける処理をしている例もあった．

2行にわたる分割　コンピュータ組版では，複数の親文字に対し，ルビを平均に配置するグループルビの場合，分割禁止とする．しかし，活字組版では，必ずしも分割禁止としないで，複合語などでは，分割していた．分割しない処理は，行の調整処理で極端な調整を必要とする場合もあったからである．

そこで，理想ではないが，誤読はされない，といった範囲で，例外のあまりでない，また，機械的に処理できる簡便な配置処理方針を考える必要があるように思われる．

以下は，こうした簡便な配置処理方法の一案である．なお，用語は JLReq（日本語組版処理の要件，Requirements for Japanese Text Layout）による．

1　簡便な配置ルールで考慮した事項

配置ルールで考慮した事項

簡便な配置ルールを考えるにあたっては，次のような事項を考慮し，また前提とした．

(1) ルビは，親文字の読み方又は意味を示すものである．そこで，誤読されないことを第一とした．

(2) できるだけ例外のでない配置処理を考えた．したがって，配置処理でも複雑な処理は要求していない．

(3) 縦組と横組とで配置処理法を変えることなく，共通の処理ができる方法とした．

(4) 親文字とルビの配置位置は，行中にある場合でも，行頭又は行末にある場合でも同じ配置位置とした．また，親文字の前後に配置する文字クラスにより親文字とルビの配置位置を変えることはしない方法とした．つまり，ルビの配置位置は，ルビの種類と親文字とルビの字数などにより配置位置を決めれば，その配置位置は，その他の条件で変化させない方法とした．

(5) 処理法は，原則として JIS X 4051（日本語文書の組版方法）で規定している方法によった．ただし，処理系定義として採用できる処理方法（オプション）を採用した場合がある．

(6) ルビの文字サイズは，親文字の文字サイズの 1/2 を初期値（デフォルト値）として採用した．ただし，ルビの文字サイズが親文字の文字サイズの 1/2 以外であっても採用できる配置方法とした．

(7) ルビを親文字の両側に配置する例もあるが，ここでは，片側に配置する場合に限定した．両側に配置する処理は，今後の課題とした．

注釈としてのルビ　行間にルビと同じ配置位置に注釈を配置する方法もある（行間注）．この配置処理は，ここでは適用範囲としない．

サイズの基準　JIS X 4051 では，ルビの文字サイズを親文字の 1/2 としているので，サイズの基準をルビ文字のサイズとしている例が多い．しかし，ここでは，ルビ文字のサイズは 1/2 と限定しないので，親文字のサイズを基準として記述する．

ルビの文字サイズ　活字組版では，3.5 ポイントのルビを準備していない場合もあった．そこで，7 ポイントの親文字にルビを付ける場合，4 ポイントのルビ文字を使用していた例もある．

また，見出しなど，親文字が大きな文字の場合は，1/2 以下にしていた例もある．

ルビの種類

ルビの種類は，親文字とルビとの対応関係から，次の 3 つとする（JLReq，“3.3.1　ルビの使用”参照）．

(1) モノルビ

(2) 熟語ルビ

(3) グループルビ

どのルビとするかは，親文字とルビとの対応関係による．モノルビは 1 文字の親文字とルビ文字が対応し，熟語ルビは，複数の親文字のそれぞれの 1 文字とルビ文字が対応し，かつ複数の親文字とルビとを一体として扱い，グループルビでは，複数の親文字の全体に対しルビが対応する（図 1 参照）．いずれのルビかは指定による．

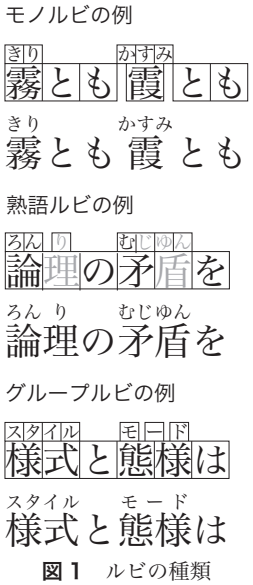


図 1　ルビの種類

2 ルビの簡便な配置ルール

ルビの文字サイズと配置位置

ルビの文字サイズと親文字に対する行送り方向の配置位置は，次による．

(1) ルビの文字サイズは，親文字の文字サイズの 1/2 を初期値（デフォルト値）とする．

(2) 縦組のルビは，親文字の右側とし，親文字の文字の外枠とルビ文字の外枠を接して配置する（図 2 参照）．

(3) 横組のルビは，親文字の上側とし，親文字の文字の外枠とルビ文字の外枠を接して配置する（図 3 参照）．

以下では，モノルビ，熟語ルビ，グループルビの配置処理を解説するが，熟語ルビは，処理がやや複雑なので，モノルビ，グループルビ，熟語ルビの順序で解説する．

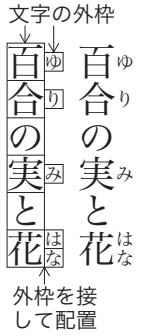


図 2　縦組のルビの例

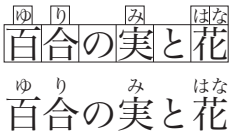


図 3　横組のルビの例

モノルビの配置処理

モノルビの配置処理は，次による．

(1) ルビの字数が 2 字以上の場合は，ルビ文字の文字列の字間はベタ組とする．なお，ルビ文字が連数字中の文字（cl-24），単位記号中の文字（cl-25），欧文用間隔（cl-26），欧文用文字（cl-27）のように固有の字幅を持つ文字の場合には，それぞれの文字の固有の字幅に応じて配置する（図 4 参照）．

(2) ルビ文字の文字列と親文字の字詰め方向の中心をそろえて配置する（図 5 参照）．

(3) モノルビの場合，親文字とそれに付くルビ文字の文字列は，一体として扱い，2 行に分割してはならない．

(4) 親文字よりルビ文字の文字列の全長が長い場合，親文字からはみ出したルビ文字を親文字の前又は後ろに配置する漢字等（cl-19），平仮名（cl-15），片仮名（cl-16）などに掛けてはならない（図 5 参照）．ただし，次の場合に限り，ルビを掛ける（図 6 参照）．

—親文字の前に配置する終わり括弧類（cl-02），句点類（cl-06），読点類（cl-07），和字間隔（cl-14）又は中点類（cl-05）の後ろのアキ（このアキは，中点類（cl-05）以外，通常は親文字の二分（中点類（cl-05）は四分），ただし，行の調整処理で二分アキや四分アキが詰められている場合は，調整で詰められた空き量までとする，例えば，四分アキとなっていれば，四分まで）

—親文字の後ろに配置する始め括弧類（cl-01），和字間隔（cl-14）又は中点類（cl-05）の前のアキ（始め括弧類（cl-01）は通常は親文字の二分，中点類（cl-05）は四分），ただし，行の調整処理で二分アキや四分アキが詰められている場合は，調整で詰められた空き量までとする，例えば，四分アキとなっていれば，四分まで）

(5) 親文字よりルビ文字の文字列の全長が長い場合，行頭ではルビ

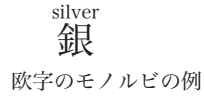


図 4　欧字のモノルビの例

はみ出したルビと前後の文字の関係　JIS X 4051 では，親文字群（親文字及びそれに付随するルビ）と，その前後に配置する文字との関係について，ルビ文字を最大でルビ文字の文字サイズまで，親文字群の前後に配置する仮名にかけてよいという規定とともに，“処理系定義として，ルビ文字を前又は後ろの文字にかけずに配置してもよい”との規定も書かれている．

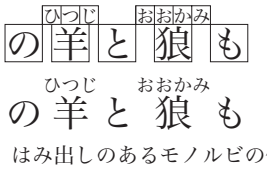


図 5　はみ出しのあるモノルビの例 1

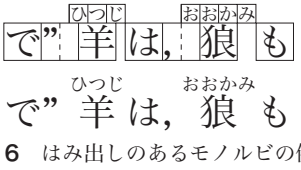


図 6　はみ出しのあるモノルビの例 2

文字の文字列の先頭を行頭にそろえ（図 7 参照），行末ではルビ文字の文字列の末尾を行末にそろえる（図 8 参照）．

グループルビの配置処理

グループルビの配置処理は，次による．

(1) ルビ文字及び親文字が連数字中の文字（cl-24），単位記号中の文字（cl-25），欧文用間隔（cl-26），欧文用文字（cl-27）のように固有の字幅を持った文字以外の平仮名（cl-15），片仮名（cl-16），漢字等（cl-19）などの場合は，それぞれをベタ組にした場合のルビ文字の文字列の全長と，親文字の文字列の全長を比較し，次のように配置する．

—ルビ文字の文字列の全長と親文字の文字列の全長が同じ場合は，それぞれをベタ組とし，ルビ文字の文字列及び親文字の文字列の字詰め方向の中心をそろえて配置する（図 9 参照）．

—ルビ文字の文字列の全長が親文字の文字列の全長より短い場合は，ルビ文字の文字列の字間及びその前後を空け，それぞれの文字列の全長を同じにし，各文字列の字詰め方向の中心をそろえて配置する．空ける量は，ルビ文字の文字列の字間の空き量の大きさ 2 に対し，親文字の文字列の先頭からルビ文字の文字列の先頭までの空き量及び親文字の文字列の末尾からルビ文字の文字列の末尾までの空き量を 1 の比率で空ける（図 10 参照）．ただし，ルビ文字の文字列の先頭及びルビ文字の文字列の末尾の最大の空き量は，親文字の文字サイズの 1/2 とし，ルビ文字の文字列の字間の空き量を均等に増やす（図 11 参照）．

—ルビ文字の文字列の全長が親文字の文字列の全長より長い場合は，親文字の文字列の字間及びその前後を空け，それぞれの文字列の全長を同じにし，各文字列の字詰め方向の中心をそろえて配置する．空ける量は，親文字の文字列の字間の空き量の大きさ 2 に対し，ルビ文字の文字列の先頭から親文字の文字列の先頭までの空き量及びルビ文字の文字列の末尾から親文字の文字列の末尾までの空き量を 1 の比率で空ける（図 12 参照）．

(2) 親文字が連数字中の文字（cl-24），単位記号中の文字（cl-25），欧文用間隔（cl-26），欧文用文字（cl-27）のように固有の字幅を持った文字で，かつ，ルビ文字が平仮名（cl-15），片仮名（cl-16），漢字等（cl-19）などの場合は，次による（図 13 参照）．

—ルビ文字の文字列の全長と親文字の文字列の全長が同じ場合は，それぞれをベタ組とし，ルビ文字の文字列及び親文字の文字列の字詰め方向の中心をそろえて配置する．

—ルビ文字の文字列の全長が親文字の文字列の全長より短い場合は，ルビ文字の文字列の字間及びその前後を空け，それぞれの文字列の全長を同じにし，各文字列の字詰め方向の中心をそろえて配置する．空ける量は，ルビ文字の文字列の字間の空き量の大きさ 2 に対し，親文字の文字列の先頭からルビ文字の文字列の先頭までの空き量及び親文字の文字列の末尾からルビ文字の文字列の末尾までの空き量を 1 の比率で空ける．

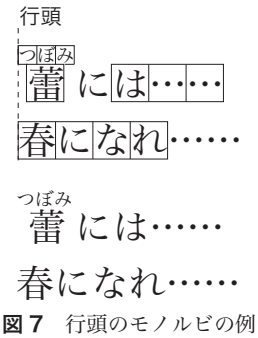


図 7　行頭のモノルビの例

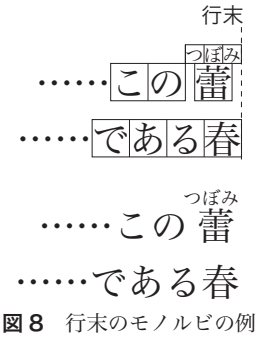


図 8　行末のモノルビの例

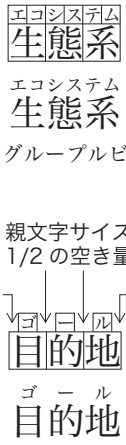


図 9　グループルビの例 1

図 10　グループルビの例 2

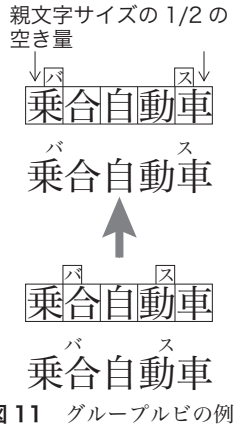


図 11　グループルビの例 3

—ルビ文字の文字列の全長が親文字の文字列の全長より長い場合は，それぞれをベタ組とし，ルビ文字の文字列及び親文字の文字列の字詰め方向の中心をそろえて配置する．この場合，ルビ文字は親文字からはみ出すことになる．

(3) ルビ文字が連数字中の文字（cl-24），単位記号中の文字（cl-25），欧文用間隔（cl-26），欧文用文字（cl-27）のように固有の字幅を持った文字で，かつ，親文字が平仮名（cl-15），片仮名（cl-16），漢字等（cl-19）などの場合は，次による（図 14 参照）．

—ルビ文字の文字列の全長と親文字の文字列の全長が同じ場合は，それぞれをベタ組とし，ルビ文字の文字列及び親文字の文字列の字詰め方向の中心をそろえて配置する．

—ルビ文字の文字列の全長が親文字の文字列の全長より短い場合は，それぞれをベタ組とし，ルビ文字の文字列及び親文字の文字列の字詰め方向の中心をそろえて配置する．

—ルビ文字の文字列の全長が親文字の文字列の全長より長い場合は，親文字の文字列の字間及びその前後を空け，それぞれの文字列の全長を同じにし，各文字列の字詰め方向の中心をそろえて配置する．空ける量は，親文字の文字列の字間の空き量の大きさ 2 に対し，ルビ文字の文字列の先頭から親文字の文字列の先頭までの空き量及びルビ文字の文字列の末尾から親文字の文字列の末尾までの空き量を 1 の比率で空ける．

(4) 親文字の文字列よりルビ文字の文字列の全長が長い場合，親文字からはみ出したルビ文字を前又は後ろに配置する文字に掛けてよいかどうかはモノルビの処理で説明した方法による（図 14 参照）．また，親文字の文字列よりルビ文字の文字列の全長が長い場合の行頭又は行末での配置処理も，モノルビの処理で説明した方法による．

(5) グループルビの場合，親文字の文字列とそれに付くルビ文字の文字列は，一体として扱い，2 行に分割してはならない．

熟語ルビの配置処理

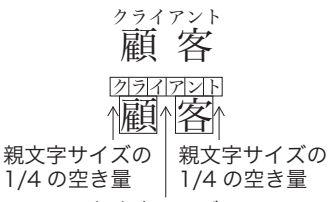
熟語ルビの配置処理は，次による．

(1) 熟語ルビでは，各親文字とルビ文字とが対応している．この各親文字に対応したそれぞれのルビ文字の文字列をベタ組にした場合において，各親文字に対応したそれぞれのルビ文字の文字列のすべてにおいて，その全長が親文字の文字サイズ以下のときは，次による．

—各親文字に対応したルビ文字が 1 字の場合は，親文字とルビ文字の字詰め方向の中心をそろえて配置する（図 16 参照）．

—各親文字に対応したルビ文字が 2 字以上の場合は，ルビ文字の文字列の字間をベタ組とし，親文字とルビ文字の文字列の字詰め方向の中心をそろえて配置する（図 16 参照）．

(2) 各親文字に対応したそれぞれのルビ文字の文字列をベタ組にした場合において，各親文字に対応したそれぞれのルビ文字の文字



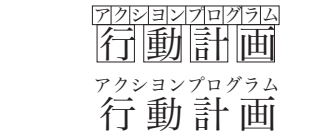


図 12　グループルビの例 4

図 13　欧字を含むルビの例

図 14　はみ出しのあるグループルビの例

グループルビの 2 行にわたる分割　グループルビは，一体として扱うことから分割禁止としている．しかし，前述したように 2 行に分割する例もあり，このことを考慮すると，熟語ルビと同様に，グループルビでも，親文字とルビの組合せを考慮した分割ができることが望ましいといえよう．

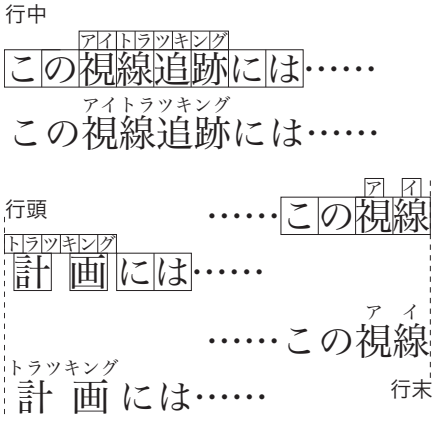


図 15　グループルビの分割例

図 16　熟語ルビの例 1

列の全長が 1 つでも親文字の文字サイズを越えるときは，グループルビと同じ配置処理とする（図 17，図 18 参照）．

(3) 熟語ルビは，各親文字間で，各親文字とルビ文字の組合せを維持したうえで，2 行にわたる文字間での分割ができるものとする．この場合，行末又は行頭で親文字が 1 字なったときは，モノルビと同じ配置処理とし，親文字が 2 字以上となったときは，ここで説明した熟語ルビの配置処理とする（図 19 参照）．

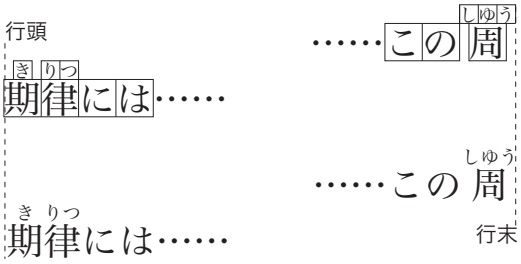
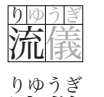


図 19　熟語ルビの分割例

(4) ルビ文字の文字列の全長が親文字の文字列の全長より長い場合，親文字からはみ出したルビ文字を親文字の前又は後ろに配置する文字に掛けてよいかどうかは，モノルビの配置処理と同じとする．また，親文字の文字列よりルビ文字の文字列の全長が長い場合の行頭又は行末での配置処理も，モノルビの処理で説明した方法による．



りゅうぎ
流儀

しゅうきりつ
周期律

図 17　熟語ルビの例 2

もんしょう
紋章

ちゅうしゃく
注釈

図 18　熟語ルビの例 3

3 ルビとアクセシビリティ

ルビによるアクセシビリティの向上

視覚障害者などのアクセシビリティを向上させるためにルビは一定の役割を果たしている．そこで，アクセシビリティに関係するルビの問題について補足しておく．

読書が困難となる原因はさまざまであり，これに伴いアクセシビリティを向上させるために要求される要件も異なってくる．例えば，以下のような要求がある．

—漢字がほとんど読めない子がいるので，すべての漢字にルビを付ける総ルビが必要となる．

—学習が進むと読める漢字も増える．また，総ルビで何回か読んだあとでは，難しい漢字だけのルビでよいので，一部の漢字だけにルビを付けるパラルビが必要である..

—ルビと親文字との区別がつかず，別の文字と認識するケースがあるので，ルビと親文字がはっきり区別できるような表示方法が必要である．あるいは，漢字が読める人にとっては，かえってルビが余分なものとなる可能性もあるので，ルビを付けない方法も必要となる．

—括弧書きなどで代用できることから，両側のルビはそれほど必要としない．

総ルビとパラルビ　総ルビとパラルビについては，JLReq の“3.3.2　ルビの付け方”で解説されている

ルビの必要性　2017 年度の DAISY 教科書一般申請利用者へのアンケート結果によれば，61%の児童が総ルビを要求している．

また，同調査では，32%の児童がパラルビで十分としている．

総ルビで何回か読んだあとは，難しい漢字だけのルビでいいということである．また，紙の教科書はパラルビなので，それを忠実に再現する必要性もある．

アクセシビリティにとってのルビの表示要件

上記の要求例から，アクセシビリティにとってのルビの表示要件としては，次のようなことが求められる．

(1) 総ルビが必要である．

(2) パラルビが必要である．なお，パラルビでは，学習の進行に伴い学ぶ漢字が増えていくので，コンテンツの内容や読者層に応じて，指定した学年以降で習う漢字についてルビを表示する必要がある．

(3) ルビなしが必要である．

(4) 製作コスト，配布コスト，利用者の管理コストを考慮すると，同一コンテンツで，総ルビ，パラルビ及びルビなしと表示し分けられる必要がある．

(5) ルビと親文字がはっきり区別できるようにルビの色を変える等の表示方法の工夫が必要である．

小学校・中学校で学習する漢字　小学校で学習する漢字は，“小学校学習指導要領”で定められている．この漢字は，一般には“教育漢字”と呼ばれている．2017（平成29年）に告示された“小学校学習指導要領”において，“別表　学年別漢字配当表”として学年別に1026字の漢字が示されている．

中学校では，学年別に学ぶ漢字は示されていないが，教育漢字以外の常用漢字の1110字を学び，小学校・中学校を通じて常用漢字の2136字すべてを学ぶことになっている．

W3C Editor's Draft

Rules for Simple Placement of Ruby

W3C Editor's Draft 17 February 2019

This version:
https://w3c.github.io/jlreq/docs/simple-ruby/

Latest published version:
https://www.w3.org/TR/simple-ruby/

Latest editor's draft:
https://w3c.github.io/jlreq/docs/simple-ruby/

Previous editor's draft:
https://florian.rivoal.net/ruby/

Editor:
Florian Rivoal (Invited Expert)

Author:
Toshi Kobayashi

Participate:
GitHub w3c/jlreq
File a bug
Commit history
Pull requests

This document is also available in this non-normative format: Original Japanese (PDF)

Abstract

A simple set of rules for placement of Ruby text in Japanese typography.

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 was initially written in Japanese and translated to English by the Japanese Writing Technology Working Group of the Advanced Publishing Laboratory of Keio University.

It represents the subjective view of its authors and contributors as to one possible approach to address the problem, and does not claim to be the only possible solution. It is submitted to present a non-Japanese speaking audience with this particular approach, and to encourage discussion of this topic.

The original Japanese version is available in PDF format.

This document was published by the Internationalization Working Group as an Editor's Draft.

GitHub Issues are preferred for discussion of this specification.

Publication as an Editor's Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 1 March 2019 W3C Process Document.

Table of Contents

§ 1. Introduction

This section is non-normative.

§ 1.1 The Difficulties of Ruby Processing

When performing ruby layout, the following factors need to be considered in order to decide on the position:

1. How to handle the correspondence between the base characters and the ruby
2. What to do when the string of base characters is longer than the ruby string
3. What to do when the string of base characters is shorter than the ruby string
4. When the ruby string protrudes from the base character string, whether it can be allowed to be laid over the characters preceding or following, and whether this affects the position of the base characters

> NOTE: Protrusion of ruby from base characters
> In movable type typesetting, ruby characters protruding from their base characters may not hang over neighboring kanji, but often were allowed to hang over neighboring kana. However, when the ruby is katakana, some publishers would set it so that it would not hang over the kana neighboring the base character. Also, when the characters around the base characters were kanji on one side and kana on the other side, for the sake of balance the ruby would sometimes be set so as to hang neither over the kanji nor over the kana (which would therefore both be spaced away from the base character).

5. When the ruby string protrudes from the base character string, and the base character string is at the start or the end of the line, whether the base character string or the ruby string should be aligned with the line edge
6. When there are multiple base characters, whether there can be line wrap opportunity between them

> NOTE: Wrap opportunities
> In computer-based typesetting, rather than always suppressing line wrap opportunities, they would be allowed in cases like compound words. This is because it may otherwise triggers very large spacing adjustments during justification.

In movable type typography, such matters were resolved based generic principles, and could always be corrected during the proofreading phase. Essentially, each case was adjusted individually in a flexible manner.

In computer-based typesetting, the layout needs to be more or less determined based on predetermined rules, but it remained necessary to adjust the results in certain cases, for example by changing the association between base characters and the ruby string, or by switching to a different placement policy.

§ 1.2 Web Ruby placement

When thinking about computing placement for web content, it is not practical to decide on the positioning case by case as was done in movable type typography. It is therefore necessary to decide upon comprehensive rules that provide solutions to all the problems listed above, so that placement may be determined fully automatically. Considering all the possibilities that existed in movable type typesetting, the system to be designed needs to be very complex.

However, when considering the ideal positioning of ruby, it seems inevitable that exceptions will occur, causing issues.

In such cases, rather than ideal positioning, we must at least make sure that the positioning causes no misunderstanding; there are also practical limits to how complex the system can be in order to be practically implementable.

The following is a proposal for a simple processing system. The target audience is implementers and specification writers. Note that the terminology is based on that defined in JLReq [JLREQ].

§ 2. Matters considered by the simple placement rules

This section is non-normative.

§ 2.1 Matters considered by the placement rules

Here are the fundamental assumptions underlying the simple placement rules.

1. Ruby is used to display the reading or the meaning of the base characters. I therefore consider avoiding misreadings as the number one priority.

> NOTE: Ruby as notes
> Notes are sometimes placed between lines similarly to how ruby is laid out (inter-linear notes). The processing of this arrangement is not covered in this document.

2. I have devised a method that attempts to reduce exceptions as much as possible. Therefore, there is no requirement for complex processing.
3. The method is agnostic to horizontal vs vertical writing, and will use the same logic in either case.
4. The method places the ruby string relative to the base character string the same way when they occur in the middle, start, or end of the line. Moreover, this method does not change the relative position of the ruby string to the base character string depending on preceding or subsequent characters. In other words, this method calculates a position for the ruby relative to the base string that does not change depending on context.
5. Generally speaking, the processing method is based on JIS X 4051 [JIS4051] (Formatting rules for Japanese documents). However, in some cases, optional steps are used.
6. The ruby font size is set to half of the base character's size as a default. However, the method supports using different sizes than 1/2.

> NOTE: Reference size
> Because the size of ruby characters used in JIS X 4051 [JIS4051] set to 1/2 there are many examples that use the size of the ruby character as the reference. However, since ruby is not restricted to 1/2, this document uses the size the base character as the reference.

> NOTE: Size of ruby characters
> In movable type typesetting, when ruby characters of size 3.5 points were not available, based characters of 7 points where sometimes paired with ruby characters of 4 points. Also, ruby associated with large base characters, such as those in titles, are sometimes smaller than 1/2.

7. While there are cases of ruby on both sides of the base string exist, the method defined here only handles ruby on one side. Handling both sides is left as a future exercise.

§ 2.2 Types of ruby

Ruby may be divided into the following 3 different types, based on the relationship between the ruby and the base characters (see JLReq "3.3.1 Usage of Ruby" [JLREQ]).

1. Mono-ruby
2. Jukugo-ruby
3. Group-ruby

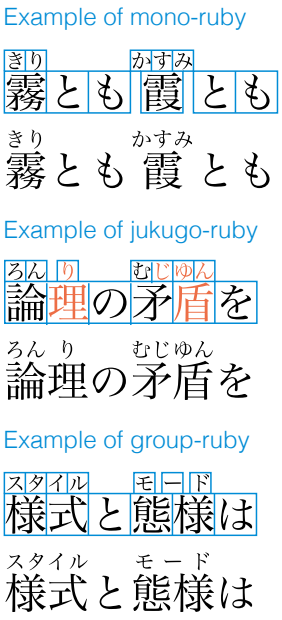


Figure 1 Types of ruby

Which one to use depends on the relationship between the ruby and the base characters. Mono-ruby is used to connect ruby to a single base character, Jokugo-ruby is used when multiple base characters each have a corresponding ruby and at the same time the whole group needs to be processed together, and group-ruby is used when ruby is attached to a group of base characters together (see Figure 1). Each is used when specified.

§ 3. Rules for Simple Placement of Ruby

§ 3.1 Ruby character size and character placement

The size of the ruby characters and their placement in the inline direction relative to the base characters is as follows:

1. The size of the ruby is by default set to half of the size of the base characters.
2. In vertical text, ruby is placed to the right of the base characters, and the character frame of the ruby is placed flush against the character frame of the base characters.

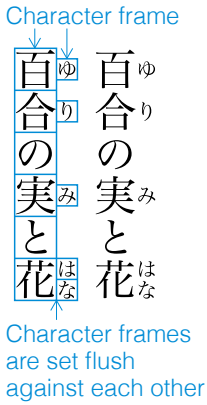


Figure 2 Example of vertical ruby

3. In horizontal text, ruby is placed to the top of the base characters, and the character frame of the ruby is placed flush against the character frame of the base characters.

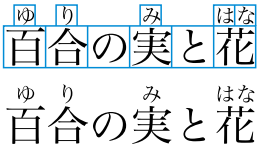


Figure 3 Example of horizontal ruby

The following sections describe in detail the placement of mono-ruby, jukugo-ruby, and group-ruby. However, since jukugo-ruby is more complex, it is explained last.

§ 3.2 Placement of mono-ruby

Mono-ruby is placed as follows:

1. When the ruby is made of two or more characters, each character in the ruby string is placed immediately next to its neighboring character, without any inter-letter spacing. Furthermore, when the ruby is composed of characters such as Grouped numerals (cl-24), Unit symbols (cl-25), Western word space (cl-26), or Western characters (cl-27) [JLREQ] which have their own individual width, they are placed based on each character's metrics.

Figure 4 Example mono-ruby with western characters

2. The center of the ruby string and of the base character string are aligned in the inline direction. (see Figure 5).

3. Since the base character and its associated ruby form a single unit there is no line wrapping opportunity inside a mono-ruby.
4. When the ruby string is longer than the base character string, the part of the ruby string that extends beyond the base characters must not hang over the characters preceding or following, if they are ideographic characters (cl-19), Hiragana (cl-15), Katakana (cl-16), etc [JLREQ]. Space is introduced accordingly between these preceding or following characters and the base characters.

> NOTE: Protruding over surrounding characters
> The main placement method defined in JIS X 4051 [JIS4051] allows some amount of overhang over the preceding and following base characters, but recognizes the method defined here as an allowed variant.

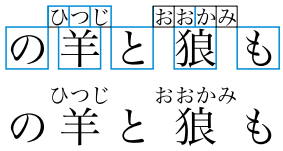


Figure 5 Example 1 of mono-ruby protruding

However, in the following cases, the ruby characters do hang over the preceding or following characters (see Figure 6).

- If the character preceding the base character is one of: Closing brackets (cl-02), Full stops (cl-06), Commas (cl-07), Full-width ideographic space (cl-14), or Middle dots (cl-05) [JLREQ], then the ruby must hang over the blank portion at the end the character. (This blank portion is usually half the character's width, except in the case of Middle dots (cl-05) [JLREq] where it is a fourth of the character width). However, if this blank part has been compressed due to justification or similar processing of the line, then the ruby may only hang over the resulting compressed blank space (e.g. if it was reduced from half to a quarter em, hang at most a quarter em).
- If the character following the base character is one of: Opening brackets (cl-01) or Full-width ideographic space (cl-14), Middle dots (cl-05) [JLREQ], then the ruby must hang over the blank portion at the start the character. (This blank portion is usually half the character's width for Opening brackets (cl-01), or a quarter of the character's width for Middle dots (cl-05) [JLREQ]) However, if this blank part has been compressed due to justification or similar processing of the line, then the ruby may only hang over the resulting compressed blank space (e.g. if it was reduced from half to a quarter em, hang at most a quarter em).

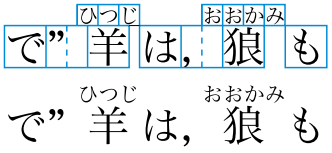


Figure 6 Example 2 of mono-ruby protruding

5. When the ruby string is longer than the base character string, and the ruby falls at the start of the line, then the start of the ruby string is aligned with the line's start edge (see Figure 7), while if the ruby falls at the end of the line, then the end of the ruby string is aligned with the line's end edge (see Figure 8).

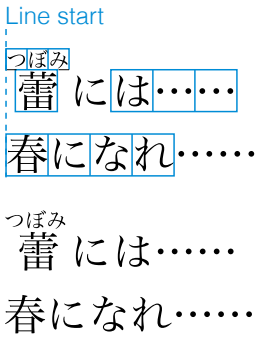


Figure 7 Example of mono-ruby at the line start

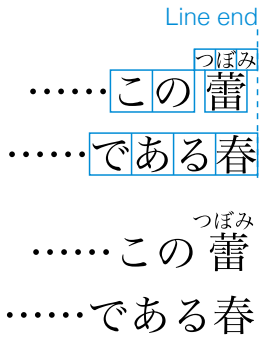


Figure 8 Example of mono-ruby at the line end

§ 3.3 Placement of group-ruby

Group-ruby is placed as follows:

1. When the ruby string and the base character string are composed of characters such as Hiragana (cl-15), Katakana (cl-16), Ideographic characters (cl-19), and so on, excluding characters like Grouped numerals (cl-24), Unit symbols (cl-25), Western word space (cl-26), or Western characters (cl-27) [JLREQ] which have their own individual width, the way they are positioned depends on how their respective lengths would compare if they were each laid out without any inter-letter spacing:

- When their respective lengths would be the same, both are laid out without inter-letter spacing and placed such that their respective centers in the inline direction are aligned (see Figure 9).

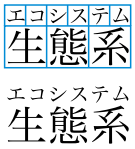


Figure 9 Example 1 of group-ruby

- When the ruby string is shorter than the base character string, space is inserted between every character in the ruby string as well as at the start and the end of the ruby string so that it becomes the same length as the base character string, then their centers in the inline direction are aligned. The size of the space inserted between each of the ruby characters is twice the size of the space inserted at the end and at the start (see Figure 10).

Figure 10 Example 2 of group-ruby

However, the size space inserted at the start and end must be capped at no more than half the size of one base character, and the space inserted between each ruby character is enlarged to compensate (see Figure 11).

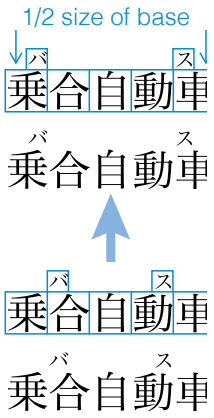


Figure 11 Example 3 of group-ruby

- When the ruby string is longer than the base character string, space is inserted between every character in the base character string as well as at the start and the end of the base character string so that it becomes the same length as the ruby string, then their centers in the inline direction are aligned. The size of the space inserted between each of the base characters is twice the size of the space inserted at the end and at the start (see Figure 12).

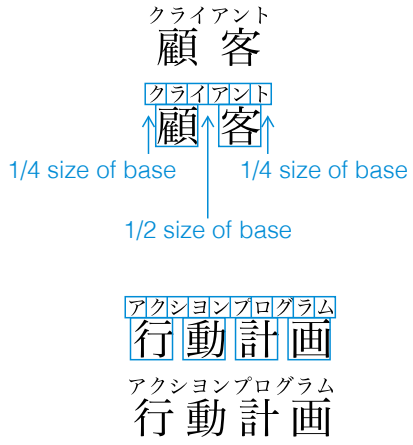


Figure 12 Example 4 of group-ruby

2. When the base character string is composed of characters like Grouped numerals (cl-24), Unit symbols (cl-25), Western word space (cl-26), or Western characters (cl-27) which have their own individual width, and the ruby string is composed of characters such as Hiragana (cl-15), Katakana (cl-16), Ideographic characters (cl-19) [JLREQ], and so on, the placement depends on the following (see Figure 13):

   - When their respective lengths would be the same, both are laid out without inter-letter spacing and placed such that their respective centers in the inline direction are aligned.
   - When the ruby string is shorter than the base character string, space is inserted between every character in the ruby string as well as at the start and the end of the ruby string so that it becomes the same length as the base character string, then their centers in the

inline direction are aligned. The size of the space inserted between each of the ruby characters is twice the size of the space inserted at the end and at the start.

- When the ruby string is longer than the base character string, both are laid out without inter-letter spacing and placed such that their respective centers in the inline direction are aligned. In this case, the ruby string protrudes from the base character string.

Figure 13 Example of ruby with western characters

3. When the ruby string is composed of characters like Grouped numerals (cl-24), Unit symbols (cl-25), Western word space (cl-26), or Western characters (cl-27) which have their own individual width, and the base character string is composed of characters such as Hiragana (cl-15), Katakana (cl-16), Ideographic characters (cl-19) [JLREQ], and so on, the placement depends on the following (see Figure 13):
   - When their respective lengths would be the same, both are laid out without inter-letter spacing and placed such that their respective centers in the inline direction are aligned.
   - When the ruby string is shorter than the base character string, both are laid out without inter-letter spacing and placed such that their respective centers in the inline direction are aligned.
   - When the ruby string is longer than the base character string, space is inserted between every character in the base character string as well as at the start and the end of the base character string so that it becomes the same length as the ruby string, then their centers in the inline direction are aligned. The size of the space inserted between each of the base characters is twice the size of the space inserted at the end and at the start.

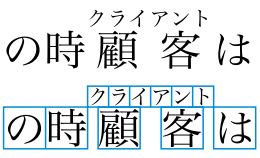


Figure 14 Example of protruding group-ruby

4. When the ruby string is longer than the base character string and protrudes, whether and how it hangs over characters preceding or following the base character string is handled in the

same way as with mono-ruby (see Figure 14). Also, when the ruby string is longer than the base character string, protrudes, and is located at the start or end of the line, the processing is also identical to that of mono-ruby.

5. In the case of group ruby, the base character string and its associated ruby string are treated as a unit, so there is no line wrapping opportunity inside either string.

> NOTE: Wrap opportunities in group-ruby
> As group-ruby is treated as a unit, there is no wrap opportunity. However, in some exceptional cases where it is wrapped, it is then processed similarly to jukugo-ruby.

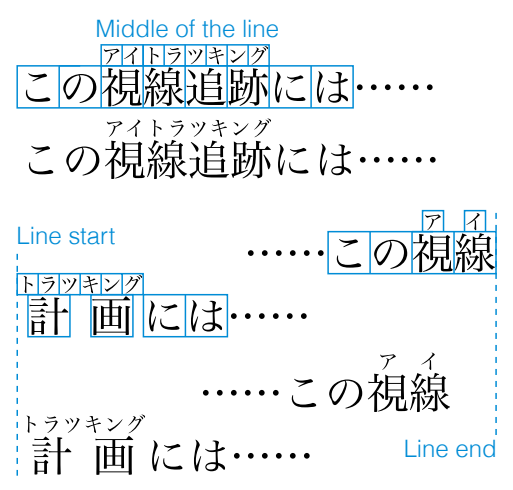


Figure 15 Wrapping group-ruby

§ 3.4 Placement of Jukugo-ruby

Jukugo-ruby is placed as follows:

1. With jukugo-ruby, each base character is associated with its own ruby string. When the length of each of these ruby string laid out without inter-letter spacing is shorter than the length of all their corresponding base characters, placement is determined as follows:

    - When the ruby string associated with an individual base character is 1 character long, the ruby character and the base character are placed such that their respective centers in the inline direction are aligned (see Figure 16).

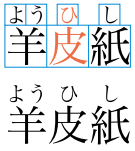


Figure 16 Example 1 of jukugo-ruby

- When the ruby string associated with an individual base character is 2 characters long or more, the ruby string is laid out without inter-letter spacing, and placed such that its center and the center of its base character are aligned in the inline direction (see Figure 16).

2. If even a single ruby string is longer than its corresponding base character when laid out without inter-letter spacing, the processing is identical to group-ruby (see Figure 17 and Figure 18).

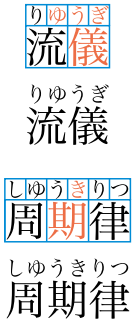


Figure 17 Example 2 of jukugo-ruby

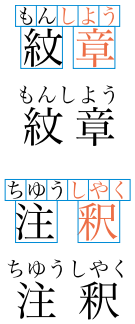


Figure 18 Example 3 of jukugo-ruby

3. With jukugo-ruby, individual base characters and their associated ruby string are treated as a unit, and line wrap opportunities are allowed between two base characters. When such a line wrap occurs, if a single base character that is part of the jukugo is placed alone at the end or at the start of a line, it is processed identically to mono-ruby; conversely when several base characters that are part of the jukugo are placed together at the end or start of a line, they are processed together as has been described in this section about jukugo-ruby (see Figure 19).

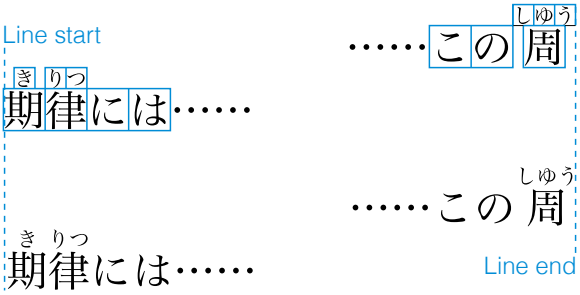


Figure 19 Example of wrapping jukugo-ruby

4. When the ruby string is longer than the base character string and protrudes, whether and how it hangs over characters preceding or following the base character string is handled in the same way as with mono-ruby. Also, when the ruby string is longer than the base character string, protrudes, and is located at the start or end of the line, the processing is also identical to that of mono-ruby.

§ 4. Ruby and Accessibility

This section is non-normative.

§ 4.1 Accessibility Improvements Using Ruby

Ruby plays a role in improving accessibility for people with visual impairments, and other sources of reading difficulties. Therefore, this section examines the relationship between ruby and accessibility.

Reading difficulties can be caused by a variety of factors, and therefore, requirements to improve accessibility also vary. For example, here are some common requirements:

- To accommodate young children who cannot read any kanji, general-ruby must be added to all kanji.

  > NOTE: General-Ruby and Para-Ruby
  > See JLReq section “3.2.2 Choice of Base Characters to be Annotated by Ruby” for an explanation of “general-ruby” and “para-ruby”. [JLREQ]

- As studies progress, a greater number of kanji is known. After having read general-ruby many times, ruby on difficult kanji only becomes sufficient. Therefore para-ruby on only some of the kanji is required.

> NOTE: The Need for Ruby
> According to the results of the 2017 DAISY survey towards general users of textbooks, 61% of children require general-ruby. Based on the same survey, para-ruby is found to be sufficient for 32% of children. This means that after having read general-ruby multiple times, having ruby on difficult kanji only is found sufficient. Moreover, printed textbooks use para-ruby, and faithful digital reproduction is needed.

> NOTE: Kanji Studied by Elementary and Middle School Students
> Kanji to be learned during primary school are defined in the “Primary School Learning Guidelines”. Those kanji are often called “educational kanji”. In the list published in 2017, 1026 kanji are listed and spread across the various school years according to the “Kanji Allotment Table by Grade”. For middle school, the split between each grade is undefined, but students are required to study the 1110 “characters in common use” no included in educational kanji so as to have studied all 2136 characters in common use by the end of middle school.

- Some people have difficulties in visually distinguishing between ruby characters and the base characters to which they are attached, and misread the combination as a different character altogether. There must be a display method that enables clearly distinguishing between the two. Also, for those who already know how to read the kanji, there must be a way to hide the ruby.
- As inline parenthesised annotations can be used instead, there is no strong need for double-sided ruby.

§ 4.2 Ruby Display Requirements for Accessibility

Based on the above, we can gather the following ruby display requirements for accessibility:

1. Support for general-ruby is required.
2. Support for para-ruby is required. Moreover, as the number of kanji known increases with the level of studies, based on the content and on the level of the reader, it must be possible to only display ruby for kanji assigned to a particular school year (or later).
3. Support for hiding ruby is required.
4. Considering the cost of production, distribution, and of user management, it is necessary to support ruby-less display, general-ruby display, and para-ruby display with the same content.
5. A method to clearly visually distinguish the ruby characters and their based characters, such as displaying them in different colors, is required.

§ A. References

§ A.1 Informative references

[JIS4051]

Formatting rules for Japanese documents (『日本語文書の組版方法』). Japanese Standards Association. 2004.

[JLREQ]

Requirements for Japanese Text Layout. Yasuhiro Anan; Hiroyuki Chiba; Junzaburo Edamoto; Richard Ishida; Tatsuo KOBAYASHI; Toshi Kobayashi; Kenzou Onozawa; Felix Sasaki; Seiichi Kato; Hajime Shiozawa et al. W3C. 3 April 2012. W3C Note. URL: https://www.w3.org/TR/jlreq/

CSS space expansion

A Collection of Interesting Ideas, 14 October 2018

This version:

https://specs.rivoal.net/css-space-expansion/

Issue Tracking:

GitLab

Inline In Spec

Editor:

Florian Rivoal

Abstract

This proposal explores a way to turn Zero Width Spaces into visible spaces. The driving use case is support for optional spacing in Japanese (know as “分かち書き”) for the benefit of language learners and people with dyslexia. If adopted, this proposal is expected to be incorporated into [CSS-TEXT-3] or [CSS-TEXT-4].

Table of Contents

§ 1. Introduction

This section is non-normative.

In a number of languages and writing system, such as Japanese or Thai, words are not deliminated by spaces (or any other character) as is the case in English (See Approaches to line breaking for a discussion the approach various languages take to word separation and line breaking).

However, even if text without spaces is the dominant style in such languages, there are cases where making word boundaries (or phrase boundaries) visible through the use of spaces is desired. This is a purely stylistic effect, with no implication on the semantics of the text.

In Japan for instance, this is commonly done in books for people learning the language—young children or foreign students. People with dyslexia also tend to find this style easier to read.

The mechanism proposed in this specification builds upon the existing use of U+200B ZERO WIDTH SPACE in the document markup as a word (or phrase) delimiter. While this practice is not that common, it is a semantically valid use of that unicode character, and the ability to trigger stylistic effects based on it can only encourage its use.

§ 2. Expanding Zero Width Spaces: the 'zero-width-space-expansion' property

Name:	***'zero-width-space-expansion'***
Value:	none | space | ideographic-space
Initial:	none
Applies to:	inline boxes
Inherited:	yes

Percentages:	N/A
Media:	visual
Computed value:	as specified
Canonical order:	per grammar
Animation type:	discrete

ISSUE 1 This name is too verbose. To be bikeshedded.

ISSUE 2 Should we allow more freeform values, like <string>, possibly limited to 1 character?

This property enables all instances of U+200B ZERO WIDTH SPACE to be replaced by the specified character. Instances `<wbr>` are considered equivalent to U+200B, and are also replaced. This substitution happens before layout, so all layout operations that depend on the characters in the content (such as CSS Text Module Level 3 §white-space-rules, line breaking, or intrinsic sizing) must use that character instead of the original U+200B.

'none'
: This property has no effect.

'space'
: All instances of U+200B ZERO WIDTH SPACE are replaced by U+0020 SPACE.

'ideographic-space'
: All instances of U+200B ZERO WIDTH SPACE are replaced by U+3000 IDEOGRAPHIC SPACE.

Like 'text-transform', this property transforms text for styling purposes. It has no effect on the underlying content, and must not affect the content of a plain text copy & paste operation.

ISSUE 3 This almost looks like instead of a new property, we could just add two new values of the 'text-transform' property. However:

- this property seems saner to handle before CSS Text Module Level 3 §white-space-rules, but 'text-transform' happens after that. Or maybe which stage 'text-transform' applies at depends on which value it uses?
- these two properties seems better to cascade separately

EXAMPLE 1

Unlike books for adults, Japanese books for young children often feature spaces between sentence segments, to facilitate reading.

Absent any particular styling, the following sentence would be rendered as depicted below.

```
<p>むかしむかし、<wbr>あるところに、<wbr>おじいさんと<wbr>おばあさんが<wbr>すん
```

むかしむかし、あるところに、おじいさんとおばあさんがすんでいました。

✧ ✧ ✧

Phrase-based spacing can be achieved with the following css:

```
p {
  zero-width-space-expansion: ideographic-space;
}
```

むかしむかし、　あるところに、　おじいさんと　おばあさんが　すんでいました。

✧ ✧ ✧

Another common variant additionally restricts the allowable line breaks to these phrase boundaries. Using the same markup, this is easily achieved with the following css:

```
p {
  word-break: keep-all;
  zero-width-space-expansion: ideographic-space;
}
```

むかしむかし、　あるところに、
おじいさんと　おばあさんが
すんでいました。

EXAMPLE 2

In addition to making the source code more readable, using `<wbr>` rather than U+200B in the markup also allow authors to classify the delimiters into different groups.

In the following example, `<wbr>` elements are either unmarked when they delimit a word, or marked with class `p` when they also delimit a phrase.

```
<p>らいしゅう<wbr>の<wbr>じゅぎょう<wbr>に<wbr class=p
>たいこ<wbr>と<wbr>ばち<wbr>を<wbr class=p
>もって<wbr>きて<wbr>ください。
```

Using this, it is possible not only to enable the rather common phrase based spacing, but also word by word spacing that is likely to be preferred by people with dyslexia to reduce ambiguities, or other variants such as a combination of phrase-based spacing and of word-based wrapping.

Figure 1 *Usual rendering*

らいしゅうのじゅぎょうにたいことばち
をもってきてください。

✧ ✧ ✧

Figure 2 *Phrase spacing*

```
p wbr.p {
  zero-width-space-expansion: ideographic-space;
}
```

らいしゅうのじゅぎょうに　たいことば
ちを　もってきてください。

✧ ✧ ✧

Figure 3 *Word spacing*

```
p wbr {
  zero-width-space-expansion: ideographic-space;
}
```

らいしゅう　の　じゅぎょう　に　たい
こ　と　ばち　を　もって　きて　くだ
さい。

✧　　✧　　✧

Figure 4 *Phrase spacing, word wrapping*

```
p {
  word-break: keep-all;
}
p wbr.p {
  zero-width-space-expansion: ideographic-space;
}
```

らいしゅうのじゅぎょうに　たいこと
ばちを　もってきてください。

✧　　✧　　✧

Figure 5 *Word spacing and wrapping*

```
p {
  word-break: keep-all;
}
p wbr {
  zero-width-space-expansion: ideographic-space;
}
```

らいしゅう　の　じゅぎょう　に
たいこ　と　ばち　を　もって　きて
ください。

§ Apendix A. Security and Privacy Considerations

This appendix is non-normative.

There are no known security or privacy impacts of this feature.

The W3C TAG is developing a Self-Review Questionnaire: Security and Privacy for editors of specifications to informatively answer. As far as currently known, here are the answers to the Questions to Consider:

Does this specification deal with personally-identifiable information?
No

Does this specification deal with high-value data?
No.

Does this specification introduce new state for an origin that persists across browsing sessions?
No.

Does this specification expose any other data to an origin that it doesn't currently have access to?
No.

Does this specification enable new script execution/loading mechanisms?
No.

Does this specification allow an origin access to a user's location?
No.

Does this specification allow an origin access to sensors on a user's device?

No.

Does this specification allow an origin access to aspects of a user's local computing environment?

No.

Does this specification allow an origin access to other devices?

No.

Does this specification allow an origin some measure of control over a user agent's native UI?

No.

Does this specification expose temporary identifiers to the web?

No.

Does this specification distinguish between behavior in first-party and third-party contexts?

No.

How should this specification work in the context of a user agent's "incognito" mode?

No difference in behavior is expected or needed.

Does this specification persist data to a user's local device?

No.

Does this specification have a "Security Considerations" and "Privacy Considerations" section?

Yes, this is the role of this Appendix.

Does this specification allow downgrading default security characteristics?

No.

§ Conformance

Conformance requirements are expressed with a combination of descriptive assertions and RFC 2119 terminology. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in the normative parts of this document are to be interpreted as described in RFC 2119. However, for readability, these words do not appear in all uppercase letters in this specification.

All of the text of this specification is normative except sections explicitly marked as non-normative, examples, and notes. [RFC2119]

Examples in this specification are introduced with the words "for example" or are set apart from the normative text with `class="example"`, like this:

> EXAMPLE 3
> This is an example of an informative example.

Informative notes begin with the word "Note" and are set apart from the normative text with

`class="note"`, like this:

> Note, this is an informative note.

§ Index

§ Terms defined by this specification

ideographic-space, in §2

none, in §2

space, in §2

zero-width-space-expansion, in §2

§ Terms defined by reference

[css-display-3] defines the following terms:

- inline box

[css-sizing-3] defines the following terms:

- intrinsic sizing

[CSS-TEXT-3] defines the following terms:

- line breaking
- text-transform

[css-values-3] defines the following terms:

- <string>

[css-values-4] defines the following terms:

- !

[HTML] defines the following terms:

- wbr

§ References

§ Normative References

[CSS-DISPLAY-3]
Tab Atkins Jr.; Elika Etemad. CSS Display Module Level 3. URL: https://drafts.csswg.org/css-display/

[CSS-SIZING-3]
Tab Atkins Jr.; Elika Etemad. CSS Intrinsic & Extrinsic Sizing Module Level 3. URL: https://drafts.csswg.org/css-sizing-3/

[CSS-TEXT-3]
Elika Etemad; Koji Ishii. CSS Text Module Level 3. URL: https://drafts.csswg.org/css-text-3/

[CSS-VALUES-3]
Tab Atkins Jr.; Elika Etemad. CSS Values and Units Module Level 3. URL: https://drafts.csswg.org/css-values-3/

[CSS-VALUES-4]

Tab Atkins Jr.; Elika Etemad. CSS Values and Units Module Level 4. URL: https://drafts.csswg.org/css-values-4/

[HTML]

Anne van Kesteren; et al. HTML Standard. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[RFC2119]

S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

§ Informative References

[CSS-TEXT-4]

Elika Etemad; Koji Ishii; Alan Stearns. CSS Text Module Level 4. URL: https://drafts.csswg.org/css-text-4/

§ Property Index

Name	Value	Initial	Applies to	Inh.	%ages	Media	Anim-ation type	Canonical order	Com-puted value
'zero-width-space-expansion'	none | space | ideographic-space	none	inline boxes	yes	N/A	visual	discrete	per grammar	as specified

§ Issues Index

ISSUE 1 This name is too verbose. To be bikeshedded. ↵

ISSUE 2 Should we allow more freeform values, like <string>, possibly limited to 1 character? ↵

ISSUE 3 This almost looks like instead of a new property, we could just add two new values of the ‘text-transform’ property. However:

- this property seems saner to handle before CSS Text Module Level 3 §white-space-rules, but ‘text-transform’ happens after that. Or maybe which stage ‘text-transform’ applies at depends on which value it uses?
- these two properties seems better to cascade separately

↵

CSS で実装すべき機能

Florian Rivoal

出版業界にとって必要な仕様書の実現

組版に関する要求事項で，日本の出版業界にとってまだ満たされていないものがある．

推進する方向・方針に同意がされていなものの他に，ブラウザー会社にとって優先順位が低く，そのことだけにより止まってしまっているプロジェクトがいくつかある．仕様書が未完成のままだったり，仕様書がほぼ完成の状態であるにもかかわらず，実装およびバグの修正がされていなかったりしている例がある．これら多くの場合，アプローチ自体には問題がなく，ブラウザー会社も納得している．したがって，その機能が重要だと考えている団体からの出資があれば推進できる．

仕様書が未完成の場合と実装サポートのためのテスト作成の場合は，当 WG のメンバーと関係者で対応ができる．また，実装とバグ修正が必要な場合も，一部の件は同じ者ができる．なお，大きなプロジェクトならば，オープンソースコンサルティング会社（例：Igalia）に頼むことも考えられる．

こうした組版の要求事項について，短期間・中期間で支援すれば効率的に推進できると想定しているテーマは下記のとおりである．

テーマ１：自動分割段組

概　要

作者・出版社が指定したサイズ（例えば一画面分）よりコンテンツが多く，そのサイズよりはみ出す場合，自動的に数列の段組に分ける機能である．この機能は，スクロール式の端末でも段組を利用でき，役立つもので，ページめくりに近い操作がブラウザーで可能になる．

なお，自動分割の入っていない現状の段組の仕様書は，完成に近いが，微調整が必要なところもある．

現　状

段組そのものの組版：仕様書はほぼ完成，実装でバグが多少ある．

段組の自動分割：CSS WG で納得できるコンセプトがまとまっていなく，仕様書そのものも作成されていない．

参　照

・本報告書の“CSS 段組の新しい処理方法の提案”

テーマ２：ル　ビ

概　要

ルビは通常の日本語組版だけでなく，アクセシビリティでも重要である．ブラウザーの Firefox のルビの実装は悪くないが，その他のブラウザーでは，その機能は十分でない．例えば，熟語ルビの表示，あいは総ルビからパラルビへの切替えは，まだ対応できない状態である．ルビの配置位置の詳細についても，調整が必要である．

さらに，ルビマークアップ（HTML）の仕様書のドラフトそのものが簡単なものなので，不足している事項がある．十分な機能をもった仕様書の案もできており，ブラウザー会社も賛成であると表明している．しかし，HTML を担当しているワーキンググループには，まだ提出されていない状態である．

小書きの仮名を直音の仮名に直す　少し細かいことであるが，ルビ内の小書きの仮名（ぁぃぅ等）を直音を示す普通の仮名（あいう等）に変換の問題がある．この機能は，アクセシビリティのうえでも重要であり，編集者からも望まれている事項である．最近この機能が仕様書に追加された．ただし，仕様書と Firefox での実装は完成しているが，他のブラウザーの実装はまだである．

現　状

中間仕様書，Firefox で中間実装はすすんでいるが，他のブラウザーで未実装である．

参　照

・本報告書の“ルビの簡便な配置ルール”
・本報告書の“Rules for Simple Placement of Ruby”
・https://drafts.csswg.org/css-ruby-1/
・http://darobin.github.io/html-ruby/
・https://drafts.csswg.org/css-text-3/#valdef-text-transform-full-size-kana

Rules for Simple Placement of Ruby　このドキュメントは，以下にも掲載されている．

Rules for Simple Placement of Ruby.pdf (https://w3c.github.io/jlreq/docs/simple-ruby/

テーマ３：分かち書き

概　要

絵本，小学生用教科書，ディスレクシアの対応などで使用される分かち書きを可能にする機能である．まだ CSS WG に提案していないが，コンセプトを fantasai 氏と開発しており，CSS WG で了解を得られことを期待している．仕様書の作成も，実装も比較的に簡単だと予想している．

現　状

未提案のドラフトがほぼ完成している．

参　照

本報告書の“CSS space expansion”

CSS space expansion　このドキュメントは，以下にも掲載されている．

CSS space expansion.pdf (https://specs.rivoal.net/css-space-expansion/)

テーマ４：行送りの安定化

概　要

CSS の従来の計算方法では，行幅・行間・配置の計算が不均一になり，綺麗に揃わない場合が多い．要素を行幅＋行間の倍数に指定することが望ましく，新しい計算方法の指定により修正できる．

現　状

初期ドラフトの段階である．

参　照

・https://drafts.csswg.org/css-inline/#line-sizing-property
・https://drafts.csswg.org/css-inline/#leading-trim-property
・https://drafts.csswg.org/css-inline/#line-fill
・https://drafts.csswg.org/css-rhythm/#block-height
・https://drafts.csswg.org/css-box/#margin-trim

テーマ５：字間の処理

概　要

本報告書の“簡便な行組版ルール”で述べている約物の字幅とアキ処理の問題である．また，和文と欧文との間のアキを自動調整する必要があるが，現状のブラウザーでは出来ていない．

CSS WG の css-text-4 ドラフトにある案を“簡便な行組版ルール”に基づいてレビューし，調整した上で実装を推進すべきだと考えている．

現　状

初期ドラフトの段階である．

参　照

・本報告書の“簡便な行組版ルール”
・https://drafts.csswg.org/css-text-4/#text-spacing-property

テーマ６：縦 書 き

概　要

多少のバグが実装に残っているが，縦書きの対応は，ほぼ完成している．正式版の完成とはいえ，安心して使用できる最終バグの修正と実装の検査が必要である．

現　状

仕様書も実装もほぼ完成している．

参　照

・https://www.w3.org/TR/css-writing-modes-3/

CSS 段組の新しい処理方法の提案

Florian Rivoal

1　従来の CSS 仕様による段組処理の問題点

現在の CSS 仕様書に従い段組の指定はできる．しかし，ウェブコンテンツには，この CSS 仕様書に従った段組は，ほとんど使われていない．CSS 仕様書に従った段組を紙への印刷等のページのあるメディアに出力し，使用する場合は問題はないが，パソコン，携帯，スマホなどのスクロール式端末で，CSS 仕様書に従った段組を表示する場合，次のような理由により，非常に使いにくいからである（具体的な例は後述する）．

- コンテンツの分量が多く，一画面分に入りきらなく，コンテンツが画面からはみ出した場合，各段ごとにスクロールを繰り返す必要があり，とても読みにくい．
- CSS の仕様に従った段組の使用が少ないので，ブラウザーの開発会社としては段組に関するバグ修正の優先度が低く，細かい問題が多く残ってる．

2　問題を解決する新処理方法の提案

スクロール式端末で段組を表示する場合，コンテンツが入りきらないときは，画面サイズ（またはページの作成者が決めたサイズ）に従い，自動的に段組の行送り方向のサイズ（横組では上下のサイズ，縦組では左右のサイズ）に制限をつけ，分割する必要がある．これにより各段ごとでのスクロールを回避でき，読みやすくなる．

そこで，著者・編集者・開発者などの指定により，行送り方向の段組の幅に制限をつける新しい処理方法を検討・作成し，CSS WG へ提案することを考えている．こうした処理方法については，ファンタサイ，Google のタブ・アットキンス，段組仕様書のエディターのレイチェル・アンドリューも賛成している．

なお，この新しい処理方法を CSS-Scroll-Snap と組み合わせて使えば，段組の一列単位でのスクロールができ，ページ組版に似た感覚での読書が可能になる（縦組の具体例は後述する）．

この処理方法により，スクロール式メディアでの段組組版が使いやすくなるとともに，さらに次のような２つのメリットがある．

- ウェブでの段組の使用が増え，ブラウザーの開発会社にとっての

印刷された書籍の段組の例　以下の図 1 は横組３段組の見開きページの例である．段の中の丸付き数字は段の配置順序を示す（以下同じ）．

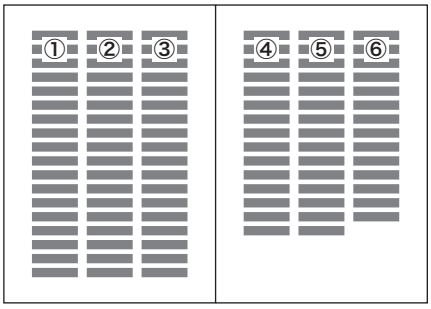


図 1　書籍の見開きページの例

段組の優先度が上がり，段組組版に関するバグの多くがなおる．現状の EPUB リーダソフトはブラウザーエンジンを利用しているものが多いため，その対応もよくなると想定できる．

- EDRLab が開発している Readium のような EPUB リーダでは，ページに分ける組版ができないブラウザーエンジンを利用しているので，段の大きい段組組版を使って，ページ組版の真似をしようとしている．そのため従来の処理方法の制限により，ページの並ぶ方向のコントロールが出来なく，縦組の適切な処理が出来ない．また，Scroll-Snap の使用も出来なく，いろいろ制限がある．提案している新処理方法では，こうした問題が解決され，画面上（EPUB リーダを含めて）で，ページらしい組版が簡単に出来る．

3 表示例による補足

従来の CSS 仕様書に従った表示例

従来の CSS 仕様書に従った横組 3 段組の表示例を図 2 と図 3 に図解して示す（参考として紙に印刷する書籍の段組例を前ページの図 1 に図解して示す）．

図 2 は，行送り方向のサイズ（横組では上下のサイズ，縦組では左右のサイズ）を制限しない場合である．この例では，図 2 のようにページ単位での分割はされないので，読むときは段ごとにスクロールを繰り返す必要がある．こうした配置は，とても読みにくい．

図 3 は，行送り方向のサイズを制限した場合である．制限された行送り方向のサイズで分割されるが，横組においては，はみ出した④以下の段は右方向に配置され（④以下の段は①②③段の下側にならない），スクロール方向が通常の場合と異なる．こうした配置も読みやすいとはいえない．

ただし，図 2 の場合において，ページというサイズを持った紙へのプリントまたは PDF ファイルとして，ページ単位で出力すると図 4 のようになる．

新処理方法の段組の表示例

新しい処理方法では，段組の行送り方向のサイズを画面のサイズで制限できる．コンテンツが多くなっても画面のサイズに応じ，スクロール方向に合わせ，文書全体を段組に整形し，段組に応じた数列ごとの分割が可能となる．

これにより一軸方向のスクロールで，一画面分ずつ読むことができる．整形された段ごとに自動的にスクロールが止まる仕組みもできる．

図 5 に表示例を図解して示す．

新処理方法による画面での縦組のページ表示

新処理方法では，ページの配列順として初期設定の下向きではなく，左向きが指定できる．

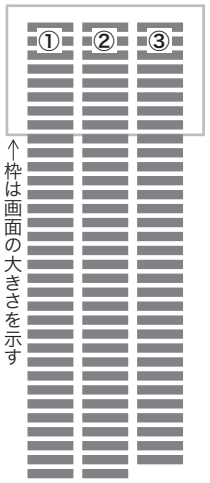


図 2　従来の CSS による画面表示例 1

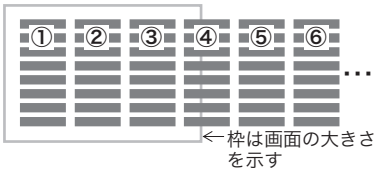


図 3　従来の CSS による画面表示例 2

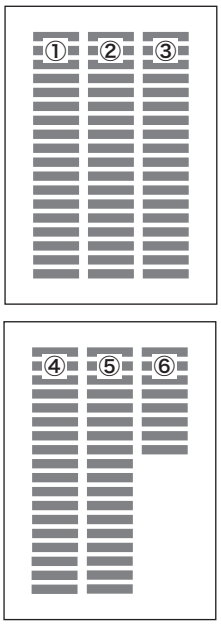


図 4　印刷の場合の従来の CSS による出力

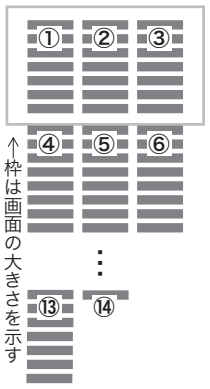


図 5　新処理方法の表示例

縦組（1 段組）において，版面の高さと幅を画面のサイズに指定することによって，スクロール式の画面で，縦組のページをめくる感覚に従った表示ができる（縦組でのページの進行は右から左となる）．自動的にページごとにスクロールが止まる仕組みもできる．

図 6 に表示例を図解してを示す．

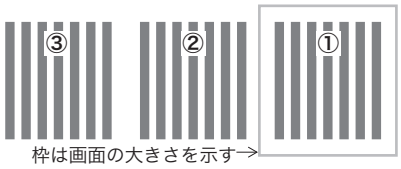


図 6　新処理方法の縦組（1 段組）の表示例