BXjscls パッケージ
（BXJS 文書クラス集）
ユーザマニュアル

八登崇之 （Takayuki YATO; aka. “ZR”）

v1.0a　[2015/08/20]

注意

このマニュアルは**暫定版**であり、内容に不完全な点がある。

BXJS 文書クラスについては、“TEX Wiki”中の記事、[*1]およびそこからたどれるサイトにある情報も参照してほしい。

1　概要

本パッケージに含まれる文書クラス集（以下では **BXJS（文書）クラス**と呼ぶ）は、奥村晴彦氏製作の「pLATEX 2ε 新ドキュメントクラス」（以下では **JS（文書）クラス**と呼ぶ）の拡張版に相当する。JS クラスのレイアウトデザインと機能をほぼ踏襲しているが、以下の点で改良が加えられている。

- JS クラスは pLATEX と upLATEX のみをサポートするが、BXJS クラスはこれらに加えて pdfLATEX と XHLATEX と LuaLATEX をサポートしており主要エンジンの全てで使用可能である。
- (u)pLATEX 以外では各々のエンジンの日本語処理パッケージを利用するが、“標準設定”を用いることで、それらのパッケージの設定を書かずに済ませられるので、pLATEX 並に簡単に日本語の文書を書き始めることができる。
- JS クラスでは、フォントのオプティカルサイズを最適にするため、（基底フォントサイズが 10pt 以外の時に）TEX の版面拡大（mag）機能を利用しているが、これが他のパッケージと衝突して不具合を起こすことがある。BXJS クラスでは mag 機能を使う他に別の方式を選べるようにしている。
- 用紙サイズや基底フォントサイズについて、任意の値を指定することができる。

[*1] `http://oku.edu.mie-u.ac.jp/~okumura/texwiki/?BXjscls`

2　最も基本的な使い方

ここでは、BXJS クラスを“標準設定”（standard和文ドライバ）で用いる場合について解説する。この場合、`\documentclass` 命令を次のように書く。[*2]

```
\documentclass[⟨エンジン⟩,⟨ドライバ⟩,ja=standard,jafont=⟨フォント指定⟩,⟨他オプション⟩]
    {⟨クラス名⟩}
```

- ⟨エンジン⟩ の指定は必須で、実際に使っている「LaTeX のコマンド名」を書く。`latex`、`platex`、`uplatex`、`pdflatex`、`xelatex`、`lualatex` が指定できる。
- DVI 出力のエンジンを使う場合は、⟨ドライバ⟩ の指定が必須で、これは実際に使っている「DVI ウェアの名前」を書く。`dvips`、`dvipdfmx`、`dviout`、`xdvi` が指定できる。PDF 出力のエンジンの場合は ⟨ドライバ⟩ の指定は不要である。
- “標準設定”を適用するので `ja=standard` を指定する。（`ja` の代わりに `jadriver` と書いてもよい。）
- 既定以外のフォント設定を利用する場合は、⟨フォント指定⟩ にその名前を書く。既定の設定を用いる場合は `jafont=...` 自体を省略する。
- その他のクラスオプション（`a4paper` 等）については、多くの場合 JS クラスと同じものが使える。
- BXJS クラスについて、⟨クラス名⟩ は以下のものがある。
  - bxjsarticle： 章のないレポート（jsarticle に相当する）
  - bxjsreport： 章のあるレポート（jsbook＋`report` に相当する）
  - bxjsbook： 書籍（jsbook に相当する）
  - bxjsslide： スライド（jsarticle＋`slide` に相当する）

XeLaTeX で bxjsarticle クラスを用いた文書の例を示す。[*3]

```
\documentclass[a4paper,xelatex,ja=standard]{bxjsarticle}
\usepackage[unicode,colorlinks,
    pdftitle={いきなり日本語}]{hyperref}
\title{いきなり日本語}
\author{七篠 権兵衛}
\begin{document}
\maketitle

\section{日本語で{\LaTeX}する}
中身はまだない。

\end{document}
```

以下では各エンジンについて、挙動を少し詳しく説明する。

[*2] もちろんクラスオプションの順序は任意である。
[*3] 組版結果における日付の出力は JS クラスと同様の「2015 年 7 月 3 日」の形式になる。

2.1 pLaTeX の場合

例えば次の設定は：

```
\documentclass[a4paper,platex,dvipdfmx,ja=standard]{bxjsarticle}
```

対応する JS クラスを用いた次の設定と（ほぼ）等価になる[*4]：

```
\documentclass[a4paper,dvipdfmx]{jsarticle}
```

次のように jafont を指定した場合は：

```
\documentclass[a4paper,platex,dvipdfmx,ja=standard,jafont=ms]{bxjsarticle}
```

その値をプリセットオプションとして pxchfon が読み込まれる：

```
\documentclass[a4paper,dvipdfmx]{jsarticle}
\usepackage[ms]{pxchfon}
```

2.2 upLaTeX の場合

例えば次の設定は：

```
\documentclass[a5paper,uplatex,dvipdfmx,ja=standard]{bxjsarticle}
```

次の設定と（ほぼ）等価になる[*5]：

```
\documentclass[uplatex,a4paper,dvipdfmx]{jsarticle}
```

jafont オプションの扱いは pLaTeX の場合と同じである。

2.3 pdfLaTeX の場合

エンジン指定が pdflatex の場合、日本語処理パッケージとして bxcjkjatype（これ自体は内部で CJK パッケージを読み込む）を利用する。

例えば次の設定は：

```
\documentclass[a4paper,pdflatex,ja=standard]{bxjsarticle}
```

次の設定と大体同じであるが、ただし文書レイアウトは article でなく jsarticle とほぼ同じになっている：

```
\documentclass[a4paper]{article}
\usepackage[whole,autotilde]{bxcjkjatype}
```

jafont を指定した場合は：

[*4] すなわち、論理フォントは明朝が jis、ゴシックが jisg が使われる。なお、BXJS では mingoth 等の論理フォント変更のオプションはサポートされていない。

[*5] 論理フォントについては、従来のもの（明朝が upjisr-h、ゴシックが upjisg-h）に代わって、BMP 外の文字に対応したもの（明朝が upjpnrm-h、ゴシックが upjpngt-h）を採用した。組み方は従来のものと変わらない。

```
\documentclass[a4paper,pdflatex,ja=standard,jafont=ipaex]{bxjsarticle}
```

その値が bxcjkjatype のフォントプリセットになる。

```
\documentclass[a4paper]{article}
\usepackage[whole,autotilde,ipaex]{bxcjkjatype}
```

※補足：

- 自動的に文書本体が CJK* 環境[*6]で囲まれかつ \CJKtilde が有効な状態になっている。従っていきなり日本語を書き始めることができる。ただし和欧文間空白（四分空き）は手動で ~ を入れる必要がある。[*7]日本語出力の挙動の詳細については bxcjkjatype のマニュアルを参照してほしい。以下に完全な文書ソースの例を示す：

  ```
  \documentclass[a4paper,pdflatex,ja=standard]{bxjsarticle}
  \begin{document}
  日本語で~pdf{\LaTeX}~するテスト。
  \end{document}
  ```

- BXJS クラスでは（JS クラスと同様に）通常は和文に約 92.5% のスケールを施している。ところが、CJK パッケージは「和文のスケール」をサポートしていない。このため、エンジンが pdflatex の場合は他の場合と比べて和文が大きめに出力される。[*8]
- bxcjkjatype パッケージにおけるフォントの既定設定は「Type1 形式の IPAex フォント」（ipaex-type1 パッケージ）である。一方、ipaex プリセットを指定した場合は「TrueType形式の IPAex フォント」が使われるので、両者の出力は“PDF データとしては”異なる（見かけは同じのはずだが）。[*9]

2.4 XeLaTeX の場合

エンジン指定が xelatex の場合、日本語処理パッケージとして zxjatype （これ自体は内部で xeCJK パッケージを読み込む）を利用する。

例えば次の設定は：

```
\documentclass[a4paper,twocolumn,xelatex,ja=standard]{bxjsarticle}
```

次の設定と大体同じであるが、ただし文書レイアウトは jsarticle とほぼ同じになっている：

```
\documentclass[a4paper,twocolumn]{article}
\usepackage{zxjatype}
\setCJKmainfont[BoldFont=IPAexGothic]{IPAexMincho}% 明朝→ IPAex 明朝
\setCJKsansfont[BoldFont=IPAexGothic]{IPAexGothic}% ゴシック→ IPAex ゴシック
```

jafont を指定した場合は：

```
\documentclass[a4paper,xelatex,ja=standard,jafont=ms]{bxjsarticle}
```

[*6] CJKspace パッケージが読み込まれた下での CJK* 環境である。

[*7] CJK パッケージには自動で和欧文間空白を入れる機能はない。

[*8] 同じ理由で、pdflatex では scale オプションも無効になる。

[*9] ちなみに、bxcjkjatype には ipaex-type1 というオプションもあるが、この設定と既定設定（オプション無し）も動作は異なる。BXJS クラスが用いるのは既定設定の方である。

その値が zxjafont のプリセットになる。

```
\usepackage{zxjatype}
\usepackage[ms]{zxjafont}
```

2.5 LuaLaTeX の場合

エンジン指定が `lualatex` の場合、日本語処理パッケージとして LuaTeX-ja を利用する。
例えば次の設定は：

```
\documentclass[b5paper,9pt,lualatex,ja=standard]{bxjsarticle}
```

次の設定と（ほぼ）等価になる（ただし luatexja-preset は実際には読み込まれない）：

```
\documentclass[b5paper,9pt]{ltjsarticle}
\usepackage{luatexja-fontspec}
\usepackage[ipaex]{luatexja-preset}
```

`jafont` を指定した場合は：

```
\documentclass[b5paper,lualatex,ja=standard,jafont=ms]{bxjsarticle}
```

次の設定と（ほぼ）等価になる：

```
\documentclass[b5paper]{ltjsarticle}
\usepackage{luatexja-fontspec}
\usepackage[ms]{luatexja-preset}
```

※補足：

- luatexja-preset パッケージの読込が行われるのは `jafont` を指定した場合に限られる。

2.6 注意事項

主に JS クラスとの違いについての注意。

- ページレイアウトについて、JS クラスの設計思想を受け継いでいるが、全く同じになるわけではない。
- JS クラスの一部のオプションで、BXJS クラスでは使用不可能なものがある。（3.3 節参照。）
- BXJS クラスではページレイアウトを設定するために内部で geometry パッケージを読み込んでいる。そのため、後からユーザが geometry を読み込むことはできない。ページレイアウトを変更する場合は、BXJS クラスが用意している再設定用の命令（5.1 節参照）か、または geometry パッケージが提供する再設定用命令（`\geometry` 等）を利用する。
- hyperref パッケージにおける“PDF の文字コード”の設定はエンジンごとに適切な値が異なっていて複雑であり、これが不適切であるために PDF 文書情報（しおり等）が文字化けしてしまう事例が数知れない。そこで、文書クラス側でエンジン毎に適切な設定を予め行うようにした。[*10]（ただし文書クラ

[*10] 従って、(u)pLaTeX において、ほとんどの場合に pxjahyper パッケージを読み込む必要がない。ただし読み込んでも構わないし、必要な場合もある。

スが hyperref を読み込むわけではない。）

- jafont が無い場合の“既定のフォント設定”は多くのエンジンにおいて「IPAex フォント使用」であるが、(u)pLaTeX だけは異なっていて「何も指定しない状態」（JS クラスと同様）である。すなわち実際に使われる物理フォントの選択は DVI ウェアの設定に委ねられている。
- JS クラスは \ifdraft という公開名のスイッチ[*11]を用いているが、これは ifdraft パッケージと衝突する。そこで BXJS クラスでは代わりに \ifjsDraft の名前を用い、本文開始時に \ifdraft が未定義の場合に限り、\ifjsDraft を \ifdraft にコピーする処理にしている。

3 クラスオプション

3.1 BXJS クラスに特有のオプション

JS クラスには無く BXJS クラスで追加されたクラスオプション。

- エンジンオプション： 実際に使用するエンジン（LaTeX コマンド名）を指定する。有効な値は latex、platex、uplatex、pdflatex、xelatex、lualatex である。エンジンオプション（と次項の autodetect-engine の何れか）の指定は必須である。
- autodetect-engine： 使用しているエンジンを判定して、自動的に適切なエンジンオプションを設定する。[*12]
  ※ BXJS クラスの設計の思想としては、「LaTeX 文書がどのエンジンでコンパイルすべきものかはソース中に明示されるべき」と考えていて、従って、“人間が普通に”文書を作る際にはこのオプションの使用は推奨されない。このオプションは“LaTeX ソースの自動生成”が絡む処理を念頭において用意されている。
- ドライバオプション： DVI 出力のエンジンを用いる場合に、実際に使用する DVI ウェアの名前を指定する。有効な値は dvips、dvipdfmx、dviout、xdvi、そして特殊な値として dvipdfmx-if-dvi がある。これは「エンジンが DVI 出力の場合に限り dvipdfmx を指定する」ことを表すもので、autodetect-engine と一緒に使うことが想定されている。[*13]ドライバオプションの指定は必須である。
- nopapersize：“papersize special 出力”を抑止する。（JS クラスとは異なり、special 出力のオプション papersize は既定で有効である。）
  ※ papersize special を出力する他のパッケージとの干渉に対する対策。
- zw（既定）： \jsZw と等価な命令として \zw を定義する。
- nozw： zw の否定。
  ※命令名の衝突に対する対策。
- js（既定）： JS クラス（例えば bxjsreport の場合は jsbook）が読込済であるように振舞う。
  ※「JS クラスであるかによって挙動を変える」パッケージに対する対策。

[*11] スイッチなので LaTeX レベルの命令ではない。標準クラスではこれに相当するものは \if@draft という非公開の制御綴で定義されている。

[*12] 実はエンジンの判定は常に行っていて、エンジンオプションが指定された場合はそれが正しいかを検査して、誤りの場合はエラーを出すようにしている。

[*13] 「実際に dvipdfmx が指定された」場合は、dvipdfmx がグローバルオプションとしても働く。

- nojs： js の否定。
  ※つまり「JS クラスの一種である」と判定されると不都合な場合にこのオプションを指定する。
- ja=⟨名前⟩： 使用する和文ドライバの名前を指定する。（詳細は 4 節を参照。）標準で提供されている和文ドライバには minimal と standard がある。既定値は、エンジンが platex か uplatex の時は standard、それ以外は minimal。
- jadriver=⟨名前⟩： 「ja=⟨名前⟩」の（0.9 版以前で使われていた）別名。
- jafont=⟨名前⟩： “和文フォントプリセット指定”の名雨を設定する。
- japaram=⟨値⟩： “和文ドライバパラメタ”の値を設定する。
  ※ jafont と japaram の値がどのように解釈されるかは和文ドライバの仕様次第である。minimal 和文ドライバではこの 2 つの値は全く参照されない。2 節で解説した通り、standard和文ドライバでは jafont の値が利用される。現状では japaram は参照されない。
- base=⟨長さ⟩： 基底フォントサイズ（\normalsize のフォントのサイズ）を指定する。JS クラスの 10pt、11pt 等と同じ役割で、任意の値を指定できる。基底フォントサイズの既定値は 10 pt である。
  ※##pt の形のオプションには名前と実際に設定される値がずれているものが多く、例えば 11pt は 10.95 pt、14pt は 14.4 pt が実際の設定値である。[*14]これに対して base=14pt は文字通り 14 pt を設定する。
- jbase=⟨長さ⟩： 和文を基準にして基底フォントサイズを指定する。すなわち和文フォントの \normalsize のサイズを指定の長さとする。[*15]
- scale=⟨実数⟩： 和文スケール値を設定する。既定値は 0.924715（= 13 Q/10 pt）である。[*16]
  ※エンジンが (pdf)latex の時は和文スケールが無効になる。
- noscale： scale=1 と同値。
- paper={⟨横幅⟩}{⟨縦幅⟩}： 用紙サイズ設定。a4paper 等と同じ役割で、任意の値を指定できる。用紙サイズの既定値は A4 縦（210 mm × 294 mm）である。
- mag=⟨整数⟩： 版面拡大率（mag 値）の直接設定。既定は base から算出する。
  ※ mag 値が n の場合、版面が $n/1000$ 倍に拡大される。
- magstyle=⟨値⟩： “版面拡大”の実現方法を指定する。有効な値は mag、real、xreal の何れか。詳細は 3.5 節を参照。

3.2 JS クラスのオプションで使用可能なもの

これらについては名前だけ列挙するに留める。ただし、“JS クラス特有”（標準クラスに無い）オプションの一部については解説を加える。

■用紙サイズ指定 a3paper、a4paper、a5paper、a6paper、b4paper、b5paper、b6paper、a4j、a5j、b4j、b5j、a4var、b5var、letterpaper、legalpaper、executivepaper。
※ a4var は A4 変判（210 mm × 283 mm）、b5var は B5 変判（182 mm × 230 mm）。
※（BX）JS クラスでは a4j は a4paper と全く等価である。（他の b4j 等も同様。）

[*14] これは昔の LaTeX の“magstep”の習慣に由来する。
[*15] この場合に決定されるmag 値は和文スケール値にも依存することに注意。
[*16] これは JS クラスの設計に基づく値である。ただし実装の都合で、JS クラスの実際のスケール値はこれから僅かだけずれている。

■横置き　landscape。

■基底フォントサイズ　8pt、9pt、10pt、11pt、12pt、14pt、17pt、20pt、21pt、25pt、30pt、36pt、43pt、12Q、14Q、10ptj、10.5ptj、11ptj、12ptj。
※ 10pt、11pt、12pt、14pt、17pt、21pt、25pt、30pt、36pt、43pt はそれぞれmagstep の 0、0.5、1、2、3、4、5、6、7、8 である。8pt、9pt、20pt は文字通りの値。##Q ／##ptj は jbase=##Q ／ jbase=##pt を表す（つまり和文規準）。[*17]

■両面用レイアウト　oneside、twoside、vartwoside。
※ vartwoside は twoside と同様だが傍注が常に右側余白に出力される。

■段組み　onecolumn、twocolumn。

■表題ページ　titlepage、notitlepage。

■起こし　openright、openany。
※ jsbook のみ（BXJS では bxjsreport と bxjsbook）にのみ存在するオプション。

■数式配置　leqno、fleqn。

■オーバーフル警告　final、draft。

■papersize special 出力　papersize。
※ BXJS クラスでは papersize は既定で有効。

■英語化　english。

3.3　JS クラスのオプションで使用不可能なもの

- クラス変種指定： report、slide。
  ※ report 相当は bxjsreport、slide 相当は bxjsslide と別クラスになっている。
- トンボ出力： tombow、tombo、mentuke
  ※これは pLaTeX のカーネル命令を利用しているのでとりあえず除外。
- 和文フォントメトリック指定： jis、winjis、mingoth。
  ※異なるエンジンで汎用的に扱うのが難しい。
- 和文数式フォントの登録の制御： disablejfam。
  ※前項と同じ理由。[*18]

3.4　クラスオプション設定の既定値

- BXJS クラス共通： a4paper、onecolumn、final、ja=minimal、jafont は空、japaram は空、scale=0.924715、magstyle=mag

[*17] ちなみに JS クラスの（固定の）和文スケール値に従うと 10pt が jbase=13Q に相当するので 13Q というオプションは無い。
[*18] ただし、こちらは一部のエンジンだけでも対応したほうがよいかも知れない。

- bxjsarticle： 10pt、oneside、notitlepage
- bxjsreport： 10pt、oneside、titlepage、openany
- bxjsbook： 10pt、twoside、titlepage、openright
- bxjsslide： 36pt、oneside、notitlepage

3.5 magstyle オプション

JS クラスにおけるページレイアウト決定の過程では、基底フォントサイズが 10 pt 以外の場合に、“版面を拡大縮小する”という処理を採用している。[*19]これには、「フォントのオプティカルサイズの選択を最適にするため[*20]」という理由があり、またこれにより、多種の基底フォントサイズへの対応が容易になるという利点もある。[*21]ところがここで、JS クラスではこの“版面の拡大”を実現するために TEX エンジンが持つ版面拡大機能（仮に「mag 設定」と呼称する）を用いていて、これについて批判されることが多い。また、現実問題として、mag 設定が LATEX で用いられる機会は少ないため、実際に用いられた時にそれを想定していないパッケージが誤動作するという問題もある。

これらの問題を緩和するため、BXJS クラスでは“版面拡大”について他の実現方法を提供している。それを選択するのがクラスオプションの`magstyle` である。[*22]

- `magstyle=mag`（既定）： JS クラスと同様に、“版面拡大”のためにmag 設定を用いる。
- `magstyle=real`： mag 設定を一切用いず、代わりに、全てのページレイアウトのパラメタの値をスケールさせる。`\normalsize` や `\large` 等の高位フォントサイズ命令で指定されるフォントサイズもスケールさせるが、“オプティカルサイズの調整”は行わない。いわゆる「基本 35 書体」のようなオプティカルサイズでない[*23]フォントのみを用いるのであれば、この設定が最も適切である。
- `magstyle=xreal`： `real` と同様に、全てのページレイアウトのパラメタの値をスケールさせる。さらに、“オプティカルサイズの調整”を実現するために、NFSS の実装コードにパッチを当てる。[*24]この場合、mag 設定による不具合は起こらなくなるが、当然、NFSS のパッチのせいで別の不具合が起こる可能性はある。

4 和文ドライバ

BXJS クラスでは様々なエンジンについて、そのエンジンおよびそれに対応するパッケージが提供する日本語処理機能を活用することで、日本語用の文書クラスとしての機能を実現している。そこでの汎用性を確保す

[*19] 例えば、基底フォントサイズが 20 pt だとすると、まずは指定されたものの半分の縦横幅をもつ用紙に対して基底フォントサイズが 10 pt としてレイアウトを決定し、それを縦横 2 倍に拡大する、という過程をとっている。

[*20] LATEX の既定の欧文フォントである Computer Modern フォントがオプティカルサイズの性質をもつことは有名であるが、少々癖が強くて、本文を 10 pt（cmr10）で組んだ場合と 12 pt（cmr12）で組んだ場合でかなり異なった印象を受ける場合がある。JS クラスではそれを嫌って、本文（`\normalsize` のフォント）が必ず「cmr10 を拡大縮小したもの」で組まれることを企図しているのである。

[*21] BXJS で「任意の」基底フォントサイズが設定できるのもこの利点があるため。

[*22] ところで、このオプションキーの値（`mag`、`real`、`xreal`）はかなりイマイチなんだけど、もっと素敵な名前はないものだろうか……。

[*23] 或いは、オプティカルサイズに“変な癖”のない。

[*24] 要するに、`OT1/cmr/m/n/12` が要求された時に、`cmr12` でなくて `cmr10 at 12pt` が選ばれるようにする。

るため、“日本語処理機能と連携する部分”の実装をモジュールとして分離していて、これを**和文ドライバ**と呼ぶ。[*25]BXjscls のバンドルでは次の 2 つの和文ドライバを提供している。

- standard和文ドライバ： 各エンジンについて、最も一般的に用いられる特定の“日本語処理機能”（例えば `lualatex` なら LuaTEX-ja）を連携対象とした和文ドライバ。(u)pLATEX 上の JS クラスと同じくらい容易に日本語が書き始められることを目指している。
- minimal 和文ドライバ： “何も実装されていない”和文ドライバ。上級ユーザがプレアンブルや自作パッケージ等にアドホックな連携コードを書いて、好きな“日本語処理機能”との連携を実現するために用いることを想定している。

和文ドライバは自分で作製することも可能である。[*26]`bxjsja-XXX.def`（`XXX` は任意の文字列[*27]）の名前のファイルに実装コードを書いてそのファイルを配置すると、`ja=XXX` のオプション指定でその和文ドライバを利用できる。

なお、和文ドライバ指定オプション（`ja`）の既定値は `minimal` である。現実には、ほとんど全ての場合に `standard` が用いられると思われるが、種々の理由があって、これを既定値にはしていない。
※ただし、(u)pLATEX については、日本語処理機能がエンジン自体に備わっていて不可分なため少し異なる扱いになっている。[*28]`minimal` を用いる意義がほとんどないため、`standard` が既定値になっている。

5 ユーザ用命令

原則として、BXJS クラスで追加されたものだけを説明する。

5.1 レイアウト設定関連

BXJS クラスではページレイアウトの設定に geometry パッケージを用いて次の手順で行っている。

1°（基底フォントサイズにより決定されたmag 値を実際に設定する。）
2° geometry で次のパラメタを設定する。
（a）クラスオプションで指定された用紙サイズ、および `truedimen`。
（ii）bxjsarticle ／ bxjsreport の場合は次のパラメタ値。

```
headheight=10pt, footskip=0.03367\paperheight,
headsep=\footskip-\topskip, includeheadfoot,
hscale=0.76, hmarginratio=1:1, vscale=0.83, vmarginratio=1:1
```

（iii）bxjsbook の場合は次のパラメタ値。

```
headheight=10pt, headsep=6mm, nofoot, includeheadfoot,
hmargin=36mm, hmarginratio=1:1, vscale=0.83, vmarginratio=1:1
```

（iv）bxjsslide の場合は次のパラメタ値。

```
noheadfoot, hscale=0.9, hmarginratio=1:1,
vscale=0.944, vmarginratio=1:1
```

[*25] graphicx パッケージ等の「ドライバ」と類似した概念のためこの名称を用いた。
[*26] 和文ドライバの実装に必要な連携仕様の情報については、ソースコード説明書（bxjscls.pdf）の付録 A を参照してほしい。
[*27] カテゴリコード 11 または 12 の文字からなる必要がある。
[*28] JS クラスの実装から分離した「日本語処理関連」のコードを minimal に配置している。

3° 後処理を行う。以下の処理が含まれる。

- `textwidth` を全角幅の整数倍に、`textheight` を整数行分の自然長になるように丸める。
- `marginpar` 関連の設定を行う。

ページレイアウトの再設定のために次の命令が用意されている。

- `\setpagelayout{`⟨設定⟩`}`： 現在のページレイアウトの設定の一部を修正する。⟨設定⟩ は geometry のパラメタの記述であり、現在の設定に追記して geometry が再設定を行った後、再び 3° の後処理が行われる。
- `\setpagelayout*{`⟨設定⟩`}`： 用紙以外の設定をリセットして改めてページレイアウトの設定を行う。具体的には、2° の(i)と ⟨設定⟩ の内容を用いて geometry が再設定を行った後、再び 3° の後処理が行われる。

なお、`\geometry` 命令を直接呼び出すことも可能である。当然この場合は 3° の後処理は行われない。

5.2 和文用設定関連

- `\jsZw`： 和文の全角幅を表す。
- `\zw`： `\jsZw` の別名。[*29]ただし `nozw` 指定時は定義されない。

5.2.1 standard和文ドライバの場合

`standard` 和文ドライバ使用時を指定した時）は和文に関連する文書ソース記述をエンジンに依らずに共通になることを目指している。従って、和文関連の組版パラメタの設定*1 についても「共通の命令」が提供される。

- 和文ファミリ変更命令： pLaTeX と同様に、`\mcfamily` で「明朝」、`\gtfamily` で「ゴシック」に変更される。`\textmc`、`\textgt` も使える。
- 欧文ファミリ変更命令での和文の連動: JS クラスと同様[*30]に、`\rmfamily` で和文が「明朝」、`\sffamily` および `\ttfamily` で和文が「ゴシック」に変更される。
- `\zw`： 和文の全角幅を表す。例えば `2\zw` が pLaTeX の `2zw` に相当する。
- `\jQ`、`\jH`、`\trueQ`、`\trueH`： それぞれ pLaTeX の単位 Q、H、trueQ、trueH に相当する長さ。
- `\ascQ`： `\trueQ` を和文スケール値で割った長さ。[*31]例えば、`\fontsize{10\ascQ}{16\trueH}` で和文のサイズが 10 Q になる。
- `\setxkanjiskip{`⟨長さ⟩`}`: 和欧文間空白の量を指定する。pLaTeX での `\setlength{\xkanjiskip}{`⟨長さ⟩`}` に相当する。
- `\getxkanjiskip`： 現在の和欧文間空白の量を表す文字列に展開される。pLaTeX での `\xkanjiskip`

[*29] LuaTeX-ja では「実際の全角幅」を表す命令 `\zw`（pLaTeX の `zw` と本当に等価）が規定されている。`lualatex` エンジン指定かつ和文ドライバが `standard` の場合はこの `\zw` の定義がそのまま使われる。（従って `nozw` は実質的に無効である。）なお、`\jsZw` は「規約上の全角幅」であり、「実際の全角幅」と本来は一致するはずだが、実際には計算誤差のせいで僅かに値が異なる。

[*30] ちなみに、(u)pLaTeX の既定ではこの連動は起こらない。

[*31] 命令名は“anti-scaled Q”の略。

の読出[*32]に相当する。

- `\autoxspacing`／`\noautoxspacing`： 和欧文間空白の挿入を有効／無効にする。pLaTeX の同名の命令と同等。
- `\setkanjiskip{`⟨長さ⟩`}`： 和文間空白の量を指定する。pLaTeX での `\setlength{\kanjiskip}{`⟨長さ⟩`}` に相当する。
- `\getkanjiskip`： 現在の和文間空白の量を表す文字列に展開される。pLaTeX での `\kanjiskip` の読出[*33]に相当する。
- `\autospacing`／`\noautospacing`： 和文間空白の挿入を有効／無効にする。pLaTeX の同名の命令と同等。

例えば、pLaTeX において、次のように「和文間空白」を利用して均等割りを行うという技法が知られている。

```
%% \kintouwari{<整数 n>}{<テキスト>}
% n 全角の幅にテキストを均等割りで出力する.
\newcommand\kintouwari[2]{{%
  \setlength{\kanjiskip}{\fill}%
  \makebox[#1zw][s]{#2}}}
```

これと同等のものを、次のようにエンジン非依存な形で書くことができる。

```
\newcommand\kintouwari[2]{{%
  \setkanjiskip{\fill}%
  \makebox[#1\zw][s]{#2}}}
```

[*32] TeX 言語でいうと `\the\xkanjiskip`。
[*33] TeX 言語でいうと `\the\kanjiskip`。