# Creating Flow Frames for Posters, Brochures or Magazines using flowfram.sty version 2.0

Nicola L.C. Talbot

Dickimaw Books
dickimaw-books.com

2025-11-24

This document is also available as HTML (`ffuserguide.html`).

Version 2.0 has been mostly rewritten using LaTeX3 commands and updated to take into account changes to the output routine in the LaTeX kernel. Some defaults have changed. If there are any compatibility issues you can rollback to v1.18 or v1.17. For example:

```
\usepackage{flowfram}[=1.18]
```

Note that v1.18 was never publicly released but is just a minor change to v1.17 to allow for rollback with corrections.

# Contents

# List of Examples

If an example shows the icon 📎🗎 then the source code is embedded in the PDF as an attachment. If your PDF viewer supports attachments, you can extract the self-contained example file to try it out for yourself. Alternatively, you can click on the download icon 📥🗎 which will try downloading the example source code from your closest CTAN mirror, but make sure that this user manual matches the version on CTAN first. You can also try using:

```
texdoc -l ffuserguide-example⟨nnn⟩
```

where ⟨*nnn*⟩ is the example number zero-padded to three digits to find out if the example files are installed on your device.

# 1. Introduction

*This chapter provides a brief overview of the package, the package options and the various frame types.*

This document is the user manual for the flowfram package. Advanced users wanting further details of the package should read the documented code `flowfram-code.pdf`. Sample files are provided with the documentation.

The flowfram package is a LaTeX package designed to enable you to create text frames in a document such that the contents of the document environment flow from one frame to the next in the order that they were defined. This is useful for creating posters or magazines or any other form of document that does not conform to the standard one or two column layout. There's an optional helper application called FlowframTk (distributed separately) if you prefer to use a graphical user interface to set up the document layout (see §7.3).

Each frame has an identification number (IDN) that's unique to the particular type of frame that may be used to identify it. You may also assign an identification label (IDL) for reference purposes to make it easier to remember. In general, a command or environment with both a starred and unstarred version that requires an ID will need the IDN for the unstarred version and the IDL for the starred version. Some commands may allow keywords (`all`, `odd` or `even`) instead of a numeric value for the unstarred versions.

i

The flowfram package tries to make TeX do something it wasn't originally designed to do. It modifies the output routine and may not always perform as desired. Extra care must be taken if a paragraph spans frames of unequal width due to the asynchronous nature of TeX's output routine. (See §9.2.) The flowfram package has only been tested on a limited number of packages, and may well conflict with packages that also modify the output routine or rely on it being unmodified. You can switch back to the normal output routine with `\FlowFramRestoreOR` and then restore flowfram's modified output routine with `\FlowFramUnrestoreOR` if necessary (see §1.5).

If the geometry package is required, it's best to load it first and setup the page dimensions before flowfram is loaded. You can load geometry afterwards and the typeblock dimensions will automatically be updated after it has been loaded, but they won't be correct if you later change the settings with `\geometry`. The lengths governing the typeblock are:

```
\typeblockwidth
```

The width of the typeblock. This is initialised to \textwidth.

```
\typeblockheight
```

The height of the typeblock. This is initialised to \textheight.

```
\typeblockoffsety
```

The vertical offset of the typeblock. This is initialised to the sum of \topmargin, \headheight, \headsep and \voffset.

These lengths identify the dimensions and vertical offset of the typeblock to ensure that all defined frames are in the correct relative positions. If these dimensions change then the available area used by commands like \onecolumn, \twocolumn and \Ncolumn might be incorrect.

The geometry package's showframes option will be confused by flowfram's frames. You can instead use flowfram's draft option to show the layout.

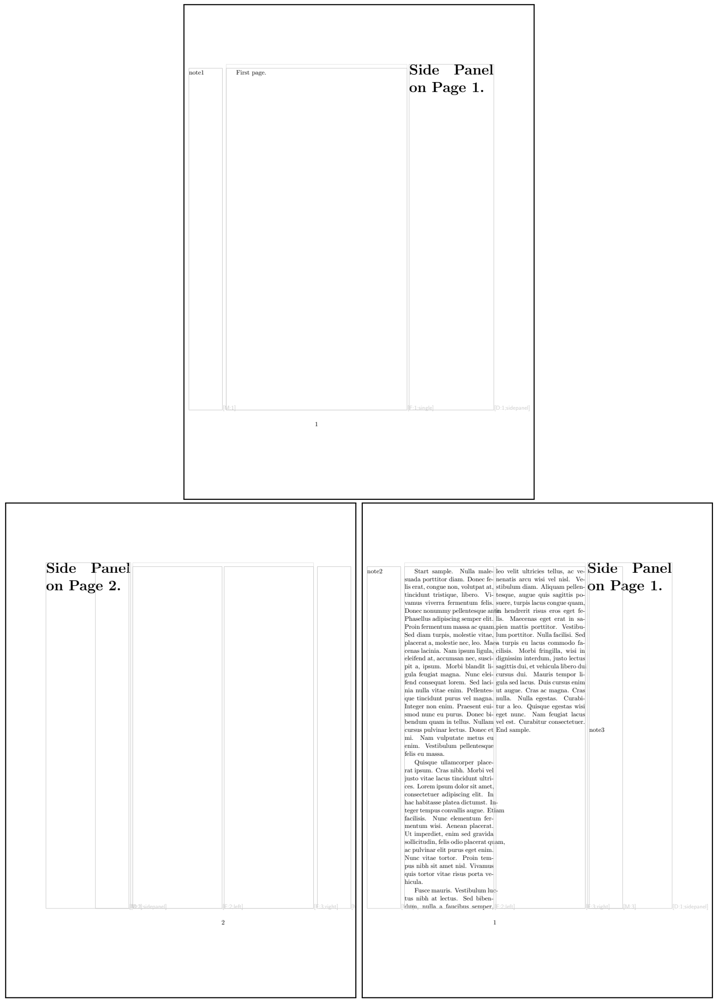
If you need to switch frames on and off, it's best to use the pages=absolute package option if the page counter is reset in the document.

Example 1 creates three flow frames. The first is labelled "single" (only shown on absolute page 1), and the other two are labelled "left" and "right" (shown on all pages except the first). The example also creates a dynamic frame labelled "sidepanel" that's shown on all pages (the default page list setting). The draft package option is used to show the outlines of each frame and the typeblock. The flow frames created with \onecolumn 1 and \twocolumn are slightly shorter than the typeblock due to the default adjustment setting (see §5.1).

```
\documentclass[11pt]{book}
\usepackage[a4paper,outer=3in,marginparwidth=2cm,
 columnsep=4pt]{geometry}
\usepackage[pages=absolute,draft]{flowfram}
\usepackage{lipsum}% provide dummy text
\pagestyle{plain}
\onecolumn[1][single]% only use on page 1
\setflowframe*{single}{margin=inner}
\twocolumn[>1][left,right]% all pages except p.1
\setflowframe*{left}{margin=left}
\setflowframe*{right}{margin=right}
\newdynamicframe{2in}{\typeblockheight}
% width & height
 {\typeblockwidth+\columnsep}{0pt}% position
 [sidepanel]% label
\newcommand{\Hugebf}{\Huge\bfseries}
\setdynamicframe*{sidepanel}
 {evenx={-2in-\columnsep}, style={Hugebf}}
\setdynamiccontents*{sidepanel}{Side Panel on
Page \thepage.}
\begin{document}
First page.\marginpar{note1}
\mainmatter
Start sample.\marginpar{note2} \lipsum[3-5]
End sample.\marginpar{note3}
\end{document}
```

Note that this example document has narrow columns with a small gap between them. This can cause overfull lines and a cramped layout, but it's for illustrative purposes only.

Example 1: Frame Page Lists



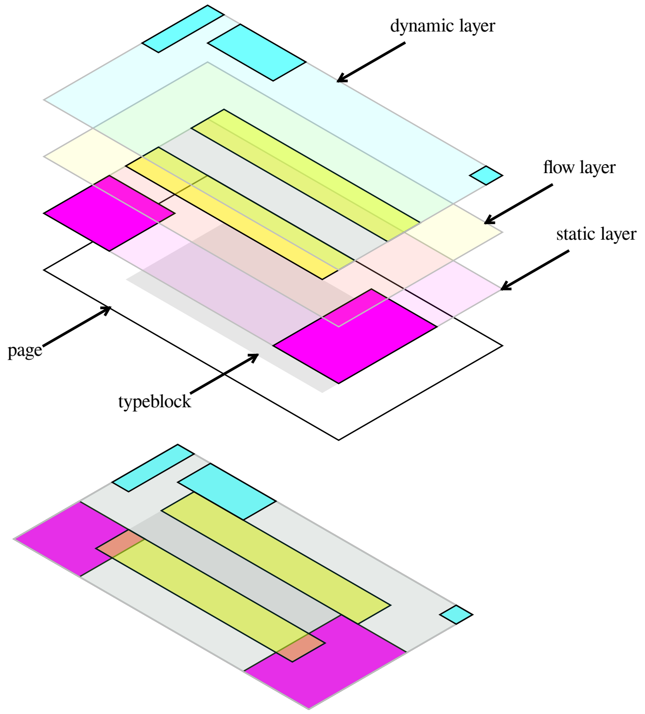


Figure 1.1.: Frame Layers

In Example 1 the page counter is reset by `\mainmatter` (which also does `\cleardoublepage`). With `pages=relative`, the page list "1" refers to any page where the page counter is 1 so the "single" flow frame would be used on both the first and third page and the "left" and "right" flow frames (which have page list >`1`) wouldn't be selected on the third page but would still be selected on the second page. However, with `pages=absolute` the page lists now refer to the absolute page.

The flowfram package provides three types of frame: flow frames, static frames and dynamic frames, each with a custom width, height and position. (These may be changed later, see §2.4.4.) The main contents of the document environment flow from one flow frame to the next in the order of definition (for the flow frames that are valid on the current page), whereas the contents of the static and dynamic frames are set explicitly using commands such as `\setstaticcontents` (see §2.2) and `\setdynamiccontents` (see §2.4).

Unless otherwise stated, all co-ordinates are relative to the bottom left hand corner of the typeblock. If you have a two-sided document, the absolute position of the typeblock may vary depending on the values of `\oddsidemargin` and `\evensidemargin`, and all the frames will shift accordingly unless otherwise indicated. Alternative positions for even pages can be set with the evenx and eveny attributes but those co-ordinates are still relative to the position of the even page typeblock.

The page list (pages), exclusion list (excludepages), and the hide and hidethis settings determine whether or not a frame is valid for a particular page.

## 1.1. Frame Stacking Order

The material on each page is placed in the following order:

1. Each static frame that's valid for the current page in ascending order of IDN (shown in magenta in figure 1.1).

2. Each flow frame that's valid for the current page in ascending order of IDN (shown in yellow in figure 1.1).
3. Each dynamic frame that's valid for the current page in ascending order of IDN (shown in cyan in figure 1.1).
4. Bounding boxes if the `draft` package option has been used.

This ordering can be used to determine if you want something to overlay or underlay everything else on the page. Note that the frames do not interact with each other. If you have two or more overlapping frames, the text in each frame will not attempt to wrap around the other frames, but will simply overwrite them.

For non-rectangular frames, see §3.2. (That is, normal static or dynamic frames that have their content shaped using `\parshape` or `\shapepar`/`\Shapepar`.)

2 Example 2 demonstrates the layers with a strange overlapping layout. Note that this example is intentionally weird and is not a recommendation!

First, some flow frames are defined. I've given them labels to make them easier to reference when changing their attributes:

```
\newflowframe
 {0.4\typeblockwidth}% width
 {0.6\typeblockwidth}% height
 {0pt}% x
 {0.6\typeblockwidth}% y
 [upper]% label
\setflowframe*{upper}{border=plain}

\newflowframe
```

```
 {0.4\typeblockwidth}% width
 {0.6\typeblockwidth}% height
 {0.6\typeblockwidth}% x
 {0pt}% y
 [lower]% label
\setflowframe*{lower}{border=shadowbox}

\newflowframe
 {0.4\typeblockwidth}% width
 {0.6\typeblockwidth}% height
 {0.3\typeblockwidth}% x
 {0.3\typeblockwidth}% y
 [middle]% label
\setflowframe*{middle}{backcolor=yellow}
```

This will need the fancybox package for the `\shadowbox` command. (More complex borders can be created with tikz or using FlowframTk.)

Order of definition matters. The above are defined in the order: `upper` (IDN 1), `lower` (IDN 2), and `middle` (IDN 3). This means that the document text will start in `upper`, flow into `lower`, and then into `middle`.

A custom counter helps to demonstrate the order in which contents are processed:

```
\newcounter{sample}
```

Next a dynamic frame is defined:

```
\newdynamicframe
 {0.5\typeblockwidth}% width
 {0.5\typeblockwidth}% height
 {0.75\typeblockwidth}% x
 {0.25\typeblockwidth}% y
 [sample]% label
```

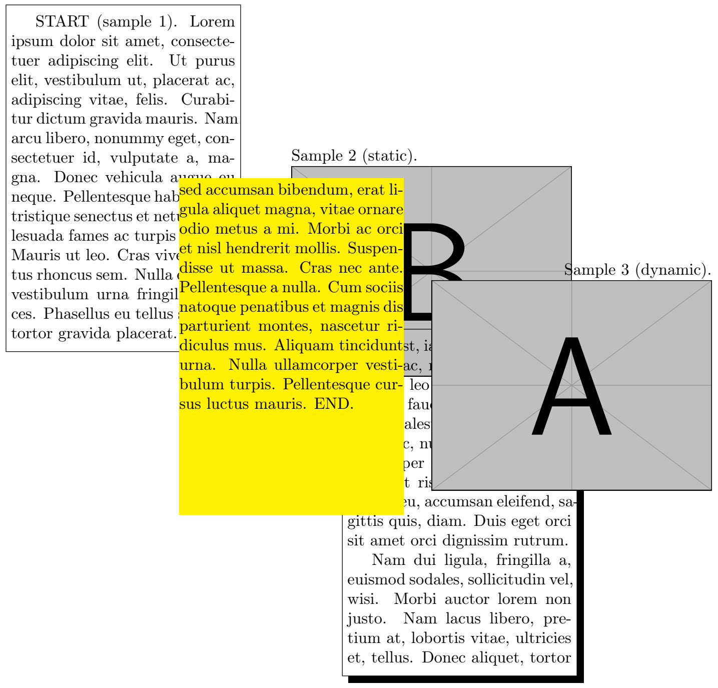
The contents are then set so that the sample counter is incremented and shown, and an image (provided by mwe) is displayed (this will need the graphicx package):

```
\setdynamiccontents*{sample}{%
 \hfill\stepcounter{sample}
  Sample \thesample\␣(dynamic).\par
 \includegraphics[width=\linewidth]
   {example-image-a}
}
```

Labels are unique to the particular frame's *type*. A static frame is defined in a similar way and, in this case, it happens to have the same label (`sample`) as the dynamic frame. This isn't a problem as they are different types of frame. However, it can be a bit confusing.

```
\newstaticframe
 {0.5\typeblockwidth}% width
 {0.5\typeblockwidth}% height
 {0.5\typeblockwidth}% x
 {0.4\typeblockwidth}% y
 [sample]% label
```

Example 2: Frame Stacking Order



START (sample 1). Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque hab
tristique senectus et net
lesuada fames ac turpis
Mauris ut leo. Cras viv
tus rhoncus sem. Nulla
vestibulum urna fringil
ces. Phasellus eu tellus
tortor gravida placerat.

sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. END.

eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor

The document uses the lipsum package to provide sample text:

```
START.
(sample \stepcounter{sample}\thesample).
\lipsum[1-2]
END.
```

Frames must be defined in the preamble, but the contents of static or dynamic frames may be set in the document body. This example sets the `sample` static frame content after the sample text (but it's still on page 1):

```
\setstaticcontents*{sample}{%
 \stepcounter{sample}
  Sample \thesample\␣(static).\par
 \includegraphics[width=\linewidth]
   {example-image-b}
}
```

Some things to note from Example 2:

- The static frame has its contents set in a box register, which means that the sample counter is incremented as soon as the contents are set, but `\setstaticcontents` was placed at the end of the one-paged document. The first increment of the sample counter occurs at the start of the first paragraph of the document text, so the counter has the value 1 when `\setstaticcontents` is processed. Therefore the contents of the static frame shows "Sample 2".

  Static frames are always positioned on the page first, so the sample static frame is drawn first and is partially obscured by the other frames with overlapping positions. Although the flow frames were filled first (because the document text came before `\setstaticcontents`), they are still drawn on top of the static frame.

- The document text automatically flows into the flow frames (using a similar mechanism to the way the standard two-column mode works). However the column order is determined by the order of definition of the flow frames. In this example, the order leads to the document text starting at the top left, continuing in the bottom right, and then moving into the middle of the page. Flow frames are always positioned after static frames so the middle and lower frames obscure the static frame that has an overlapping position.

- The dynamic frame has its contents stored in a token list variable which isn't expanded until the output routine positions it on the page so, although the frame contents were set in the preamble, the sample counter increment in the dynamic frame contents doesn't take effect until the end of the current page. Dynamic frames are always positioned after the other types of frame, so the sample dynamic frame obscures the others with overlapping positions.

- If more content is added to the document text before `\setstaticcontents` then the first page may be completed before the contents of the static frame are set. In which case, the "B" image won't show until the next page and the sample counter increment won't occur until after it has been incremented by the dynamic frame when the first page is shipped.

So it's important to consider the order when defining frames but it's also important to consider the frame type and when and how the different types of frame are filled and processed.

## 1.2. Package Options

```
\usepackage[⟨options⟩]{flowfram}
```

Some options may only be set when the package is loaded, some may be set in the preamble (before any command affected by the setting is used) and a few may be used anywhere. For those options that may be changed after the package has loaded, you can set them with:

```
\flowframsetup{⟨key=value list⟩}
```

Available options are listed in the sections below or see §9.3 for an alphabetical list.

### 1.2.1. Thumbtabs, Contents and Mini-Tocs

Thumbtabs are dynamic frames with a special label that can be created with `\makethumbtabs` (see §6.1). If enabled, they correspond to a particular sectioning unit (such as chapter), with the frame's content set to the number or the title or both. These dynamic frames are aligned against the edge of the page and are designed to be shown only in the applicable sectioning unit.

Each thumbtab frame has a corresponding index frame which may be shown in the table of contents. This requires adjusting the `\tableofcontents` command to show the index frames. The relevant sub-block in the table of contents may be adjusted so that it fills the height of the corresponding thumbtab index frame

No thumbtabs information will be written to the `ttb` file until they are first enabled with `\enablethumbtabs`. This can later be suspended with `\disablethumbtabs`. While the thumbtabs are disabled, options such as `unstarred-thumbtabs` and `matter-thumbtabs` will have no effect.

Mini-tocs are a sub-set of the table of contents that may be displayed at the start of the applicable sectioning unit. This may or may not be the same unit as the thumbtabs (if also present). The default is to match them if both thumbtabs and mini-tocs are required.

The mini-toc information is obtained when flowfram's modified `\tableofcontents` parses the `toc` file.

This means that if the `adjust-toc=off` option is used to restore the `\tableofcontents` definition that's in effect when flowfram is first loaded, then neither mini-tocs nor thumbtab indexes can be supported.

**`adjust-toc`**=⟨*setting*⟩

Specifies whether `\tableofcontents` should be adjusted to allow for thumbtab indexes and mini-tocs. This may take one of the following values:

**`adjust-toc=header`**

Adjust the table of contents, including the header. This may change the way the page header is displayed, if a page style with headers is used.

**`adjust-toc=notheader`**

Adjust the table of contents, but not the header.

**`adjust-toc=off`**

Don't adjust the table of contents. There's no support for thumbtab indexes or mini-tocs with this setting.

**`backmatter-sections`**=⟨*setting*⟩ *initial:* **`no-number`**

Classes that support `\backmatter` tend to suppress the chapter number but not section numbers. As this can cause interference, flowfram by default adds code to `\backmatter` that sets secnumdepth to `-1`. This option may be used to prevent this.

**backmatter-sections=no-change**

If `\backmatter` is defined, flowfram's additional code won't alter secnumdepth.

**backmatter-sections=no-number**

Make `\backmatter` (if defined) set secnumdepth to `-1`.

**toc-thumbtabs=**⟨*setting*⟩ *default:* **true**; *initial:* **false**

Indicates whether or not thumbtab indexes should be shown in `\tableofcontents` and whether or not to align them. Note that the alignment setting should only be used if the table of contents is on a single page. It's not appropriate for longer content. This setting has no effect with `adjust-toc=off`.

**toc-thumbtabs=aligned**

Divide the table of contents into blocks that are the same height as the corresponding thumbtab. This assumes that the start of the table of contents is aligned with the vertical offset identified when the thumbtabs were created. If the thumbtabs are shifted too far up or down the ⟨*y-offset*⟩ will need to be adjusted in the first optional argument of `\makethumbtabs`. This setting is only applicable for singled-paged table of contents.

**toc-thumbtabs=unaligned**

Don't align the sub-block in the table of contents with the corresponding thumbtab and only show the thumbtab indexes on the first page of the table of contents.

**toc-thumbtabs=true**

Show the thumbtab indexes on every page of the table of contents.

**toc-thumbtabs=false**

Don't show the thumbtab indexes in the table of contents.

**unstarred-thumbtabs=**⟨*boolean*⟩ *default:* **true**; *initial:* **false**

Indicates whether or not unstarred sectional units should have thumbtabs. Note that this is not the same as unstarred but unnumbered sectional units in the front matter or back matter. (See `matter-thumbtabs` below, for that.)

This option will have no effect if thumbtabs aren't enabled when the applicable sectional units occur.

**matter-thumbtabs=**⟨*setting*⟩

Indicates if unstarred sectional units outside of the main matter should have thumbtabs. Unstarred commands such as `\chapter` are not usually numbered outside of the main matter but behave slightly differently to the starred version. For example, `\chapter` without a star may not be numbered but may be added to the table of contents. This setting does not affect the starred version.

This setting will have no effect if thumbtabs aren't enabled outside of the main matter.

**matter-thumbtabs=main-only**

Only support thumbtabs for unstarred (numbered) units in the main matter.

`matter-thumbtabs=all`

Support thumbtabs for all unstarred units.

`matter-thumbtabs=not-front`

Only support thumbtabs for unstarred (numbered) units in the main matter and back matter but not in the front matter.

`matter-thumbtabs=not-back`

Only support thumbtabs for unstarred (numbered) units in the main matter and front matter but not in the back matter.

`thumbtab-links=`⟨*setting*⟩

Indicates which thumbtabs should have `\hyperlinks` (if supported). You will need to load hyperref if you want `\hyperlinks`.

`thumbtab-links=toc-only`

Only have `\hyperlinks` in the thumbtab indexes that are shown in the table of contents.

`thumbtab-links=all`

Have `\hyperlinks` in all the thumbtabs.

**thumbtab-links=none**

Don't have `\hyperlinks` in any of the thumbtabs.

**thumbtab-text=**⟨*setting*⟩ *initial:* **rotate**

Indicates how the text should be shown on the thumbtabs.

**thumbtab-text=rotate**

Turn the text sideways. This will either rotate the text to the right or to the left, depending on whether or not a two-sided layout is on and whether or not the current page is odd or even.

**thumbtab-text=rotate-right**

Rotate the text to the right.

**thumbtab-text=rotate-left**

Rotate the text to the left.

**thumbtab-text=stack**

Stacks the letters vertically. This doesn't look very good and is just provided for backward-compatibility.

**thumbtab-text=normal**

No rotation or stacking.

```
thumbtabs=⟨setting⟩
```
preamble only

Indicates whether or not the number or title should be shown in the thumbtabs (if enabled).

```
thumbtabs=title
```

Only show the title.

```
thumbtabs=number
```

Only show the number (if applicable).

```
thumbtabs=both
```

Show both the number (if applicable) and the title

```
thumbtabs=none
```

Don't show either the number or title.

```
sections-extra-option=⟨setting⟩
```

The flowfram package redefines the standard sectioning commands so that they have a second optional argument (which is used for the corresponding thumbtab title, if enabled), but it first saves the original definitions to use as an underlying command. Normally it will only pass the first optional argument to the underlying command.

If you are using a class, such as memoir, that also has a second optional argument, then this option identifies whether or not the second optional argument should also be passed to the underlying command. If flowfram detects that memoir has been loaded, it will automatically implement `sections-extra-option=original-and-thumbtab` otherwise it will implement `sections-extra-option=thumbtab-only`.

**`sections-extra-option=thumbtab-only`**

Only use the second optional argument as the thumbtab title (if applicable) and don't pass it to the underlying command.

**`sections-extra-option=original-and-thumbtab`**

The underlying command has a second optional argument and any second optional argument should be both passed to the underlying command and used as the thumbtab title (if enabled).

**`sections-extra-option=as-original`**

The underlying command has a second optional argument and any second optional argument provided should simply be passed to the underlying command and not used as the thumbtab title.

### 1.2.2. Dynamic Frames

**`dynamic-page-style`**=⟨*setting*⟩ *initial:* **`adjust`**
preamble only

Indicates whether or not `\makedfheaderfooter` should adjust the standard page styles. If this option is required, it should be set before `\makedfheaderfooter` otherwise it will be too late to have an effect.

`dynamic-page-style=adjust`

If `\makedfheaderfooter` is used, it will set (`\let`) the empty, plain, headings and myheadings page styles to ffempty, ffplain, ffheadings and ffmyheadings.

`dynamic-page-style=noadjust`

Don't adjust the page styles.

`dynamic-header-case=`⟨*setting*⟩ *initial:* `no-change`

Indicates whether or not `\chaptermark` (if chapters are defined) or `\sectionmark` (otherwise) with the ffheadings and ffmyheadings styles should change the case of the header text.

`dynamic-header-case=uc`

Convert to uppercase.

`dynamic-header-case=no-change`

Don't change the case.

`dynamic-subheader-case=`⟨*setting*⟩ *initial:* `no-change`

Indicates whether or not `\sectionmark` (if chapters are defined) or `\subsectionmark` (otherwise) with the ffheadings and ffmyheadings styles should change the case of the sub-header text.

`dynamic-subheader-case=uc`

Convert to uppercase.

`dynamic-subheader-case=no-change`

Don't change the case.

`dynamic-page-style-header-font={⟨code⟩}` *initial:* `\emph`

Use the given ⟨*code*⟩ to set the font used by `\chaptermark` (if chapters are available) or `\sectionmark` (otherwise) with ffheadings. The ⟨*code*⟩ may be declarations or the final command in ⟨*code*⟩ may be a text-block command.

If ⟨*code*⟩ is empty (but the equal sign = will still be required) no font change will be applied. For example, you may prefer to use the dynamic frame's style command instead.

`dynamic-page-style-subheader-font={⟨code⟩}`

Use the given ⟨*code*⟩ to set the font used by `\sectionmark` (if chapters are available) or `\subsectionmark` (otherwise) with ffheadings. The ⟨*code*⟩ may be declarations or the final command in ⟨*code*⟩ may be a text-block command. Again the ⟨*code*⟩ may be empty.

The sub-header font is initialised to match the header font, so if you change the header font it will automatically change the sub-header font as well.

`dynamic-empty-page-style={⟨value⟩}` *initial:* **`hide`**

Controls how the ffempty page style behaves with dynamic header and footer frames. Remember that with the default `dynamic-page-style=adjust`, `\makedfheaderfooter` will set the empty page style to ffempty. The initial setting is `dynamic-empty-page-style=empty`, `dynamic-empty-page-style=hide`. This means that `dynamic-empty-page-style=show` will revert back to showing empty frames.

If you have decorated the header or footer dynamic frame, such as setting a border or background colour, then simply setting the header and footer to empty (the usual behaviour of the empty page style), the header and footer frames will still be visible. This may or may not be desired.

**`dynamic-empty-page-style=empty`**

The ffempty page style will set the header and footer text to empty but the frames will still be visible. This option automatically sets `dynamic-empty-page-style=show`.

**`dynamic-empty-page-style=plain`**

The ffempty page style will behave like the ffplain style. The header and footer frames will be visible. This option automatically sets `dynamic-empty-page-style=show`.

**`dynamic-empty-page-style=headings`**

The ffempty page style will behave like the ffheadings style. The header and footer frames will be visible. This option automatically sets `dynamic-empty-page-style=show`.

**dynamic-empty-page-style=myheadings**

The ffempty page style will behave like the ffmyheadings style. The header and footer frames will be visible.

**dynamic-empty-page-style=ignore**

The ffempty page style will do nothing, so the previous page style will remain in effect. The header and footer frames will be visible. This option automatically sets `dynamic-empty-page-style=show`.

**dynamic-empty-page-style=hide**

The header and footer frames will have the hide attribute set by the ffempty page style. No other change will be made, but since the frames will no longer be shown, the frame content won't be visible.

Note that if `\thispagestyle` is used instead of `\pagestyle` then hidethis instead of hide will be set.

**dynamic-empty-page-style=show**

The header and footer frames won't have the hide (or hidethis) attribute set by the ffempty page style. Whatever `dynamic-empty-page-style` setting was in effect before `dynamic-empty-page-style=hide` will continue.

### 1.2.3. Column Command Options

These options govern column commands, such as `\onecolumn`, `\onecolumninarea`, `\twocolumn` and `\twocolumninarea`, which create one or more flow frames arranged as columns (see §5.1). This includes column commands that additionally create a frame to go above or below the column flow frames (see §5.2).

The order that the flow frames are defined matters, as the output routine selects the next flow frame in order of definition (that is, in ascending order of IDN) and cycles back to the start when a page is shipped out. So if the first flow frame to be defined is on the left, the document text will start on the left, but if the first flow frame to be defined is on the right, then the document text will start on the right.

**LR**

preamble only

The column commands, such as `\twocolumn`, will create flow frames from left to right. This is the default.

**RL**

preamble only

The column commands, such as `\twocolumn`, will create flow frames from right to left. The `RL` option may be set in the preamble but should be set before the applicable commands are used (see also §5.3).

**column-changes**=⟨*setting*⟩ *initial:* **ignore**

The flowfram package redefines `\onecolumn` and `\twocolumn` for use in the preamble to define a single flow frame (`\onecolumn`) or two flow frames (`\twocolumn`). However, some commands provided by the kernel or other classes or packages may try to use `\onecolumn` and `\twocolumn` to adjust the page layout. This option indicates what to do if these commands are encountered in the document environment.

**`column-changes=ignore`**

Ignore any instance of `\onecolumn` or `\twocolumn` found in the document environment. In the case of `\twocolumn`, if the optional argument is present, it will simply be added to the page content.

**`column-changes=clearpage`**

Like `column-changes=ignore` but does `\clearpage`.

**`column-changes=switch`**

Switch to the designated frames, which will first need to be identified. If you use `\onecolumn` or `\onecolumninarea` in the preamble (see §5.1), the first instance will automatically identify the flow frame to use for any `\onecolumn` found in the document. Alternatively, you can identify the required flow frame with:

`\SetOneColumnFrame{`⟨*ID*⟩`}` *modifier:* *

The ⟨*ID*⟩ should be the frame's IDN for the unstarred version or the IDL for the starred version.

Similarly if you use `\twocolumn` or `\twocolumninarea`, the first instance will automatically identify the flow frames to use for any `\twocolumn` found in the document. However, this is a little more complicated as a frame for the header and shorter columns are also needed in the event that `\twocolumn` is used with its optional argument. Commands such as `\twocolumnStop` or `\twocolumnDtop` will identify the frames to use in this case. You can also identify the desired frames with:

`\SetTwoColumnFrames[⟨header-type⟩][⟨header-id⟩]{⟨col-1⟩}[⟨header-col1⟩]{⟨col-2⟩}[⟨header-col-2⟩]` *modifier:* *

The starred version references the frames by their IDL. The unstarred version references them by their IDN.

The first non-headed flow frame is identified by ⟨*col1*⟩ and the second non-headed flow frame is identified by ⟨*col2*⟩. If the optional arguments are omitted, then this just establishes the two flow frames to use.

To identify a frame for the header and the two flow frames accompanying it, the optional arguments are required: ⟨*header-type*⟩ is the frame type for the header (`flow`, `dynamic` or `static`). Note that dynamic frames are best for this, although they can cause dependent counters to go out-of-sync. The header frame is identified with ⟨*header-id*⟩. The first headed flow frame is identified with ⟨*header-col1*⟩ and the second headed flow frame is identified with ⟨*header-col2*⟩.

If you don't identify the designated frames to use, you will get a warning.

### 1.2.4. Other

`pages=⟨setting⟩` *initial:* **`relative`**
preamble only

All frames have an associated page list that determines which pages they should be shown on. By default, this uses the page counter, but as that is sometimes reset (for example, when changing from front matter to main matter) this can be ambiguous. Version 1.14 introduced the absolutepage counter which is incremented every time a page is shipped out and should not be reset, which allows for unambiguous references in the page lists.

**pages**=**absolute**

Use the absolutepage counter.

**pages**=**relative**

Use the standard page counter. Note that the page lists always uses the numeric value (1, 2, etc) not the formatted value (i, ii, etc). This is the default setting for backward compatibility.

**adjust-pre-chapter**=⟨*boolean*⟩ *default:* **true**; *initial:* **true**
package option only

If true and `\chapter` is defined, the pre-chapter hook commands `\ffprechapterhook` and `\chapterfirstpagestyle` will be defined (see §1.6).

**verbose**=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

Switches on or off verbose mode. Provided to assist with debugging.

**draft**

Switches on draft mode (see §1.4).

**final**

Switches off draft mode.

**prohibit-frame-rotation**=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

If true, don't rotate frames, regardless of the angle setting.

The remaining options are provided for backward-compatibility.

**norotate**

Equivalent to setting both `prohibit-frame-rotation=true` and `thumbtab-text=stack`. The original version of flowfram was released in 2005 when there was less support for rotation, so the default was no frame rotation and no thumbtab text rotation. This is no longer applicable, so this is no longer the default.

**rotate**=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

If true, this option is equivalent to setting both `prohibit-frame-rotation=false` and `thumbtab-text=rotate-right`. If false, this option is equivalent to `norotate`.

**ttbtitle**

preamble only

Include the title in the thumbtabs (if enabled). This doesn't affect whether or not the number is shown. Use the `thumbtabs` option instead.

**ttbnotitle**

preamble only

Don't include the title in the thumbtabs (if enabled). This doesn't affect whether or not the number is shown. Use the `thumbtabs` option instead.

**ttbnum**

preamble only

Include the number in the thumbtabs (if enabled). This doesn't affect whether or not the title is shown. Use the `thumbtabs` option instead.

**ttbnonum**

preamble only

Don't include the number in the thumbtabs (if enabled). This doesn't affect whether or not the title is shown. Use the `thumbtabs` option instead.

**color**=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

preamble only

If true, enable colour support. Note that the color package is now always loaded regardless of this option but setting this value to false will prevent the textcolor, backcolor and bordercolor frame options from having an effect.

**nocolor**

preamble only

Equivalent to `color=false`.

## 1.3. Floats

The standard figure and table commands will behave as usual in the flow frames, but their starred versions, figure* and table* behave no differently from figure and table. This is because the arbitrary layout of the flow frames makes it difficult to determine where to put them.

If you really need floats to behave normally on a particular page, you can temporarily restore the normal output routine with `\FlowFramRestoreOR` (see §1.5). Note that all the defined frame won't be used until you revert back to flowfram's output routine with `\FlowFramUnrestoreOR`.

Since floats can't go in the content of static frames or dynamic frames, the flowfram package provides two environments that may be used instead. Unlike their figure and table counterparts, they are fixed in place, and so do not take an optional placement specifier.

```
\begin{staticfigure}
⟨content⟩
\end{staticfigure}
```

Simulate a figure in a static or dynamic frame.

```
\begin{statictable}
⟨content⟩
\end{statictable}
```

Simulate a table in a static or dynamic frame.

The `\caption` and `\label` commands can be used within staticfigure and statictable as usual, but remember that if the frame is displayed on multiple pages, you may end up with multiply defined labels. You may want to consider setting the clear attribute to automatically clear the frame contents on every page.

## 1.4. Draft Mode

The flowfram package's `draft` option will draw the bounding boxes for each frame that has been defined. At the bottom right of each bounding box (except for the bounding box denoting the typeblock), a marker will be shown in the form: `[`⟨*T*⟩`:`⟨*IDN*⟩`;`⟨*IDL*⟩`]`, where ⟨*T*⟩ is a single letter denoting the frame type, ⟨*IDN*⟩ is the IDN for the frame and ⟨*IDL*⟩ is the IDL for that frame. Values of ⟨*T*⟩ are: `F` (flow frame), `S` (static frame) or `D` (dynamic frame). Markers of the form: `[M:`⟨*IDN*⟩`]` indicate that the bounding box is the area taken up by the margin for flow frame with IDN ⟨*IDN*⟩.

The bounding box will not be rotated, even if a frame has rotation set.

There are conditionals that govern what types of bounding boxes should be shown. The `draft` option sets all these conditionals to true. You can instead selectively switch on or off the applicable conditions instead of using `draft` or in addition to using `draft`.

`\ifshowtypeblock` ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`

Determines whether or not the bounding box for the typeblock should be shown.

`\showtypeblocktrue`

Show the typeblock bounding box.

`\showtypeblockfalse`

Don't show the typeblock bounding box.

`\ifshowmargins` ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`

Determines whether or not the bounding box for the margins should be shown.

```
\showmarginstrue
```

Show the margin bounding box.

```
\showmarginsfalse
```

Don't show the margin bounding box.

```
\ifshowframebbox ⟨true⟩\else ⟨false⟩\fi        initial: \iffalse
```

Determines whether or not the bounding box for the frames should be shown.

```
\showframebboxtrue
```

Show the bounding box for all frames.

```
\showframebboxfalse
```

Don't show the bounding box for all frames.

You can see the layout for the current page (irrespective of whether or not the `draft` option has been set) using the command:

```
\flowframeshowlayout
```

This finishes the current page, temporarily sets draft mode, and prints an empty page. Only the frames for that page will be shown.

`\flowframeshowlayout` shows the frames for the page *after* the current page.

## 1.5. Output Routine

The flowfram package modifies the output routine to ensure that all frames are placed in the correct location and that `\textwidth` and `\textheight` and related dimensions are correctly set. In particular, it removes the page header and footer code from their normal place to allow for the header and footer to be placed in dynamic frames with arbitrary dimensions and locations.

This can cause a conflict with other packages that rely on an unaltered output routine or expect `\textwidth` and `\textheight` to reflect the one column size. For example, the geometry package's `showframes` option will be confused and may draw the outline of the typeblock shorter or narrower that it ought to be, which will consequently push the footer frame out of place. Therefore, if you want to view the page layout, use flowfram's `draft` option instead.

If you need to use a command or environment that requires the normal output routine, you can restore it with:

```
\FlowFramRestoreOR
```

not preamble

This will finish the current page (with `\finishthispage`) and setup a normal one column page layout with `\textheight` set to `\typeblockheight` and `\textwidth` set to `\typeblockwidth`. The absolutepage counter will still be incremented in the `build/page/after` hook but none of the static frames, flow frames or dynamic frames will appear on the page. The header and footer will be restored to their usual location. The figure* and table* environments and `\onecolumn` and `\twocolumn` commands will also be restored to normal.

You can later revert back to flowfram's output routine with:

```
\FlowFramUnrestoreOR
                                                    not preamble
```

This will start a new page (using `\clearpage`) and select the first flow frame that's valid for the new page. The figure* and table* environments will once again behave like their unstarred counterparts. Likewise `\onecolumn` and `\twocolumn` will return to their document environment behaviour.

## 1.6. Chapters

If the `\chapter` command has been defined, the flowfram package will add the following commands to the `cmd/chapter/before` hook:

```
\ffprechapterhook
\thispagestyle{\chapterfirstpagestyle}
```

Note that these two flowfram commands won't be defined if `\chapter` hasn't been defined or if `adjust-pre-chapter=false` has been used.

These hooks are largely redundant now that there are more convenient hooks provided with the new LaTeX hook management system, but they are retained for backward-compatibility.

```
\ffprechapterhook                                   initial: empty
```

Does nothing by default.

`\chapterfirstpagestyle` *initial:* `plain`

Expands to the page style name to use for the first page of each chapter.

If you want to use a different style, you will need to redefine `\chapterfirstpagestyle` to the name of the relevant page style. Note that some classes provide their own way of adjusting the style of the first page of each chapter.

*These hooks are used at the start of* `\chapter` *before* `\clearpage` *or* `\cleardoublepage` *is called.*

Chapter titles can be placed in a dynamic frame. See §2.4.1 for further details.

# 2. Defining New Frames

*This chapter describes how to define new frames, and how to identify and set frame contents. See also §4.3.*

There are three types of frame: flow frames, static frames and dynamic frames. They each have their own set of commands to define a new frame and to set the frame's attributes. Their identifying labels (IDLs) and numbers (IDNs) are unique to each type of frame. For example, the first flow frame to be defined will have IDN 1, but the first static frame and the first dynamic frame will also have IDN 1.

## 2.1. Flow Frames

The flow frame is the principle type of frame. The text of the document environment will flow from one frame to the next in order of definition. Each flow frame has an associated width, height, position on the page and optionally a border.

`\newflowframe[`⟨*page-list*⟩`]{`⟨*width*⟩`}{`⟨*height*⟩`}{`⟨*x*⟩`}{`⟨*y*⟩`}[`⟨*label*⟩`]`

*modifier:* *

preamble only

Defines a new flow frame, where ⟨*width*⟩ is the width of the frame, ⟨*height*⟩ is the height of the frame, (⟨*x*⟩, ⟨*y*⟩) is the position of the bottom left hand corner of the frame relative to the bottom left hand corner of the typeblock. The starred version will add a plain border to the frame.

See §4.1 if you need to calculate positions from the page edge.

The first optional argument, ⟨*page list*⟩, indicates the list of pages for which this frame is defined. A page list can either be specified by the keywords: `all`, `odd`, `even` or `none`, or by a comma-separated list of either individual page numbers or page ranges. If ⟨*page list*⟩ is omitted, `all` is assumed. A page range can be a closed range (such as, `2-8`) or an open range (such as, `<10` or `>5`). For example: `<3,5,7-11,>15` indicates pages 1, 2, 5, 7, 8, 9, 10, 11 and all pages greater than page 15. These page numbers refer to the integer value of the page counter by default, so if you have a page i and a page 1, they will both have the same layout (unless you change the page list setting somewhere between the two pages).

With the `pages=absolute` package option, the numbers in the page list refer to the absolute page number (as given by the absolutepage counter). In which case page 1 refers to the first page of the document only, regardless of whether there is another page 1 or page i later in the document.

You can't use formatted numbers (such as i or I) in the page list as flowfram needs the integer values in conditional expressions.

Each flow frame has its own unique IDN, corresponding to the order in which it was defined. So the first flow frame to be defined has IDN 1, the second has IDN 2, and so on. This number can then be used to identify the flow frame when you want to modify its settings. Alternatively, you can assign a unique IDL to the flow frame using the final optional argument ⟨*label*⟩. If omitted, the IDL will be the same as its IDN.

```
\getflowlabel{⟨IDN⟩}
```

Expands to the IDL for the flow frame identified by its IDN. An error will occur if there is no flow frame with the given IDN.

There is no equivalent command that will expand to the IDN. Instead you need to fetch the value with:

```
\getflowid{⟨cmd⟩}{⟨IDL⟩}
```

This will define the command ⟨*cmd*⟩ to expand to the IDN of the flow frame identified by its IDL. An error will occur if there is no flow frame with the given IDL.

For example, suppose the first flow frame has been defined with:

```
\newflowframe{0.6\textwidth}{\textheight}{0pt}
{0pt}[main]
```

Then later in the document:

```
The label for the first flow frame
is ``\getflowlabel{1}
''.
The flow frame labelled ``main'' has
IDN \getflowid{\myid}
{main}\myid.
```

The label for the first flow frame is “main”. The flow frame labelled “main” has IDN 1.

If the document continues beyond the last defined flow frame (for example, the flow frames have only been defined on pages 1 to 10, but the document contains 11 pages) then a single flow frame will be defined, emulating one column mode for all subsequent pages.

Flow frame attributes can be changed with `\setflowframe`. See §2.4.4 for further details.

### 2.1.1. Prematurely Ending a Flow Frame

You can force text to move immediately to the next defined flow frame using `\newpage` or `\pagebreak`. These work in an analogous way to the way they work in standard two-column mode.

```
\framebreak[⟨n⟩]
```

This command is required when a paragraph spans two flow frames of different widths, as TeX's output routine does not adjust to the new value of `\hsize` until the last paragraph of the previous column (flow frame in this case) has ended. As a result, the end of the paragraph at the beginning of the new flow frame retains the width of the previous flow frame. The optional argument is as for `\pagebreak`.

The `\framebreak` command is similar to `\newpage` in that it forces a break in the flow of the document text but, unlike `\newpage`, if `\framebreak` is used mid-paragraph it forces a break without given the appearance of a break in the paragraph.

The use of `\framebreak` can lead to unwanted excess space in the paragraph before the break.

If a paragraph spans two flow frames of unequal width without using `\framebreak` a warning will be issued. If a subtle difference in frame widths is caused by rounding errors (for example, if the frames were created using FlowframTk) you can adjust the tolerance to suppress these warnings.

`\fftolerance` *initial:* `2pt`

The tolerance used when determining whether or not to warn when moving between flow frames of different widths is given by the length register `\fftolerance`. If you want to change the tolerance, you need to change the value of this register using an appropriate length command, such as `\setlength`. For example, to suppress warnings where the difference in width is less than 3pt, do:

```
\setlength{\fftolerance}{3pt}
```

If you want to start a new page, rather than simply move to the next flow frame, use the command `\clearpage`, or for two-sided documents, to start on the next odd page do

`\cleardoublepage`.

```
\cleartoevenpage
```

This command is like `\cleardoublepage` but ensures that the next page is even.

```
\finishthispage
```

To finish the entire page (rather than just move to the next column), you can also use `\finishthispage`. This ensures that each remaining valid flow frame on the current page is shipped out with empty content.

## 2.2. Static Frames

A static frame is a rectangular area in which text neither flows into nor flows out of. That is, you have to explicitly set the contents of this frame. A static frame may have non-rectangular content (see §3.2).

A static frame may appear to contain text belonging to another frame if that other frame overlaps it. This doesn't mean that the static frame contains that text.

The contents must be set explicitly, and once set, the contents of the static frame will remain the same on each page until it is explicitly changed. Thus, a static frame can be used, for example, to make a company logo appear in the same place on every page. The clear attribute may be set to automatically clear the contents whenever a page is shipped out.

```
\newstaticframe[⟨page-list⟩]{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}
[⟨label⟩]                                        modifier: *
                                               preamble only
```

Defines a new static frame. The arguments are the same as for `\newflowframe`: ⟨*width*⟩ is the width of the static frame, ⟨*height*⟩ is the height of the frame, (⟨*x*⟩,⟨*y*⟩) is the position of the bottom left hand corner of the frame relative to the bottom left hand corner of the typeblock. The first optional argument, ⟨*page list*⟩, indicates the page list for which this static frame should appear, and the final optional argument, ⟨*label*⟩ is a unique textual IDL which you can use to identify this static frame. If no label is specified, you can refer to this frame by its unique IDN. The first static frame to be defined has IDN 1, the second has IDN 2, and so on. The starred version assigns a plain border to the frame.

```
\getstaticlabel{⟨IDN⟩}
```

Expands to the IDL for the static frame identified by its IDN. An error will occur if there is no static frame with the given IDN.

There is no equivalent command that will expand to the IDN. Instead you need to fetch the value with:

```
\getstaticid{⟨cmd⟩}{⟨IDL⟩}
```

This will define the command ⟨*cmd*⟩ to expand to the IDN of the static frame identified by its IDL. An error will occur if there is no static frame with the given IDL.

For example:

```
The label for the first static frame
is ``\getstaticlabel{1}
''.
The static frame labelled ``backleft'' has IDN
\getstaticid{\myid}
{backleft}\myid.
```

The label for the first static frame is "backleft". The static frame labelled "backleft" has IDN 1.

## 2.3. Setting the Contents

```
\begin{staticcontents}[⟨options⟩]{⟨IDN⟩}
⟨content⟩
\end{staticcontents}
```

The contents of a particular static frame may be set within the staticcontents environment, where the static frame is identified by its IDN.

```
\begin{staticcontents*}[⟨options⟩]{⟨IDL⟩}
⟨content⟩
\end{staticcontents*}
```

The starred version identifies the static frame by its IDN instead. Both environments (as from v2.0) have an optional argument that may be used to locally set the frame's attributes (see §2.4.4).

```
\setstaticcontents[options]{⟨ID⟩}{⟨content⟩}        modifier: *
```

A command alternative to the above environments. The ⟨*ID*⟩ argument is the IDN for the unstarred version and the IDL for the starred version. If the optional argument is provided, this becomes a shortcut for first setting the options and then setting the content. The definition internally uses staticcontents or staticcontents* as applicable.

Frame attributes are global variables.

*For advanced users:* Unlike dynamic frames, which store their contents in a global token list variable, static frames set their contents in a box. A box variable, `\staticframe`, is used for temporary storage when the content is set in an lrbox environment.

flowfram/staticbox/before

The hook flowfram/staticbox/before is used immediately before the `\staticframe` box is set.

flowfram/staticbox/after

The hook flowfram/staticbox/after is used immediately before the `\staticframe` box is set.

The actual box variable associated with the static frame contents is then set after the flowfram/staticbox/after hook. Since this always occurs within the staticcontents or staticcontents* environment (either explicitly or implicitly via `\setstaticcontents`), any changes made by the hooks will be scoped by the usual environment grouping.

### 2.3.1. Continued Text

Although text can't flow into or out of a static frame, it's possible to simulate this effect.

```
\continueonframe[⟨text⟩]{⟨ID⟩}
```

This command may be used when either setting the contents of a static frame or a dynamic frame. When the command is encountered, the content that follows will be placed in the identified frame. The frame type depends on the context. If `\continueonframe` is used while setting the content of a static frame then ⟨*ID*⟩ refers to another static frame. It will match the encapsulating command or environment. (For dynamic frames, see §2.4.3.)

This command forces a break in the text whilst at the same time trying to justify it, so overly large space may occur.

For example, if `\continueonframe` is used within the body of staticcontents then ⟨*ID*⟩ refers to the next static frame's IDN (since staticcontents references the frame by its IDN), but if `\continueonframe` is used within the body of staticcontents* then ⟨*ID*⟩ refers to the next static frame's IDL (since staticcontents* references the frame by its IDL).

This command is actually just a convenient shortcut where the behaviour is determined by the encapsulating command or environment. For example:

```
\begin{staticcontents}{1}
Some text…
\continueonframe[Continued/]{2}
Some more text…
\end{staticcontents}
```

This is simply a shortcut for:

```
\begin{staticcontents}{1}
Some text…
\ffcontinuedtextlayout{Continued/}
\end{staticcontents}
\begin{staticcontents}{2}
\ffstaticpostcontinued{1}{2}
Some more text…
\end{staticcontents}
```

Whereas:

```
\begin{staticcontents*}{leftframe}
Some text…
\continueonframe[Continued/]{rightframe}
Some more text…
\end{staticcontents*}
```

This is a shortcut for:

```
\begin{staticcontents*}{leftframe}
Some text…
\ffcontinuedtextlayout{Continued/}
\end{staticcontents*}
\begin{staticcontents*}{rightframe}
\ffstaticpostcontinued{⟨IDN1⟩}{⟨IDN2⟩}
Some more text…
\end{staticcontents*}
```

where ⟨*IDN1*⟩ is the IDN of `leftframe` and ⟨*IDN2*⟩ is the IDN of `rightframe`.

Bear in mind that this means that `\continueonframe` causes a change in scope so any local changes made before the command won't still be in effect after it.

The optional argument ⟨*text*⟩ specifies some continuation text to place at the end of the first frame. If ⟨*text*⟩ is omitted, then (for static frames) the default is:

`\ffdefaultstaticcontinuetext{`⟨*IDN1*⟩`}{`⟨*IDN2*⟩`}`
*initial:* `\ffdefaultcontinuetext`

where ⟨*IDN1*⟩ is the IDN of the first static frame and ⟨*IDN2*⟩ is the IDN of the continuation frame. Note that these arguments are always numeric, regardless of the encapsulating command or environment.

By default, `\ffdefaultstaticcontinuetext` ignores its arguments and simply expands to:

`\ffdefaultcontinuetext` *initial: empty*

This does nothing by default. It's also used by `\ffdefaultdynamiccontinuetext` so if you do, for example:

```
\renewcommand{\ffdefaultcontinuetext}{Continued/}
```

then this text will be used as the default for both static frames and dynamic frames.

An empty optional argument in `\continueonframe` is not the same as omitting the optional argument. This has changed in version 2.0. In earlier versions, the default value was simply empty.

The formatting of the supplied ⟨*text*⟩ is applied by:

```
\ffcontinuedtextlayout{⟨text⟩}
```

The default definition fills the paragraph so that the end is flush with the right margin and then typesets the ⟨*text*⟩ flush right on the next line encapsulated with:

```
\ffcontinuedtextfont{⟨text⟩}
```

The default definition expands to `\emph{\small ⟨text⟩}`.

This assumes that it should appear as though no paragraph break occurs in the transition between the two frames. If you want a paragraph break you need to explicitly put one before and after `\continueonframe`.

For example:

```
\begin{staticcontents*}{frame1}
Some text in the first frame. (Let's
assume this frame is somewhere on the
left half of the page.)
```

```
\continueonframe[Continued on the right]{frame2}

This is some text in the second frame. (Let's
assume this frame is somewhere on the
right half of the same page.)
\end{staticcontents*}
```

Alternatively, just redefine `\ffcontinuedtextlayout` and `\ffdefaultpostcontinued`. However, there seems little point in using `\continueonframe` in this case. You may just as well use two staticcontents environments to set the contents separately.

The content of the continuation static frame will start with:

```
\ffstaticpostcontinued{⟨IDN1⟩}{⟨IDN2⟩}
                          initial: \ffdefaultpostcontinued
```

The first argument is the IDN of the previous frame (the frame at the point where `\continueonframe` occurs). The second argument is the IDN of the continuation frame.

Both arguments are always numeric irrespective of whether or not the starred or unstarred version of staticcontents or `\setstaticcontents` was used (the second argument is an integer variable whereas the first is the actual number).

The default definition simply ignores its arguments and expands to:

```
\ffdefaultpostcontinued
```

This just does:

```
\par \noindent \ignorespaces
```

which is what suppresses the normal paragraph indentation at the start of the continuation frame content.

The arguments of `\ffstaticpostcontinued` provides a way of referencing back to the previous frame.

Example 3 redefines the default definitions to use `\relativeframelocation` (see §4.3) but first some layout commands described in §5.2 are used to create a flow frame with a static frame above it and another flow frame with a static frame below it. 3

```
\onecolumnStopinarea{1in}
 {0.5\typeblockwidth-10pt}{\typeblockheight}
 {0pt}{0pt}
\setstaticframe{\value{maxstatic}}
 {label=upper,border=plain}

\onecolumnSbottominarea{1in}
 {0.5\typeblockwidth-10pt}{\typeblockheight}
 {0.5\typeblockwidth+10pt}{0pt}
\setstaticframe{\value{maxstatic}}
 {label=lower,border=plain,parindent=1em}
```

I've assigned IDLs to the static frames: `upper` for the first one and `lower` for the second one. I've also given both static frames a border to make them stand out. The second (`lower`) has also been given its own paragraph indentation rather than using the default (given by `\sdfparindent`).

Now the defaults are changed:

```
\renewcommand{\ffdefaultstaticcontinuetext}[2]{%
 Continued
 \relativeframelocation{static}{#2}{static}{#1}%
}
\renewcommand{\ffstaticpostcontinued}[2]{%
 \ffcontinuedtextlayout{%
   Continued from
  \relativeframelocation{static}{#1}{static}{#2}%
 }%
 \ffdefaultpostcontinued
}
```

and then content is added to both static frames but `\continueonframe` is used to transition between the two. The optional argument isn't present so the default will be used:

```
\begin{staticcontents*}{upper}
Some text…
\continueonframe{lower}
resuming where we left off…

Next paragraph.
\end{staticcontents*}
```

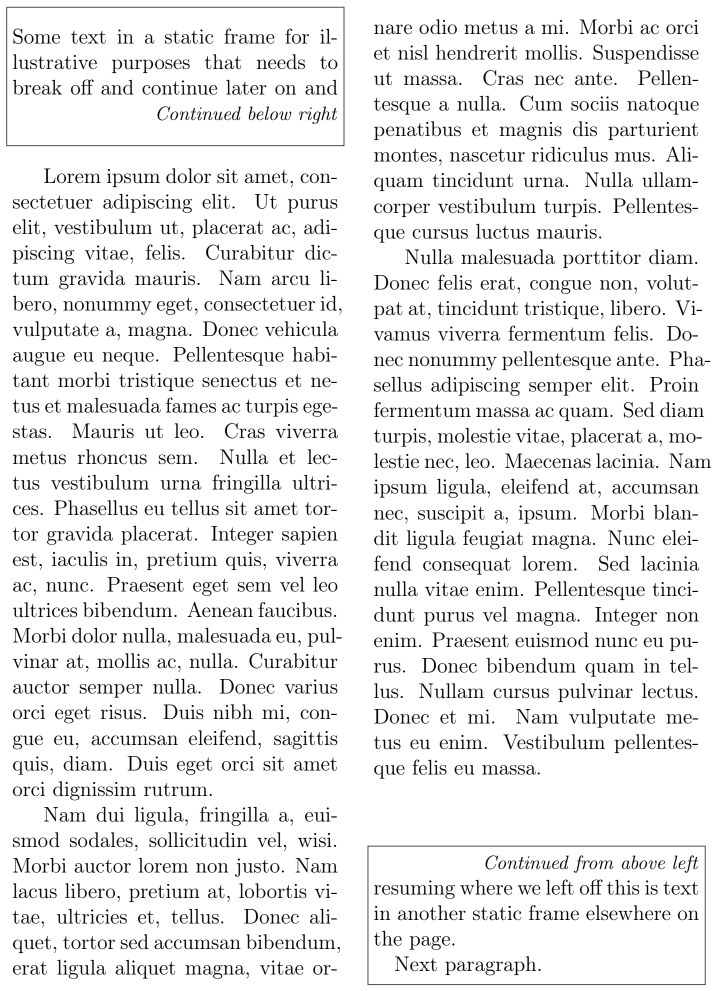
The lipsum package is used for filler text. Note that because of the narrow columns, there are some overfull lines.

### 2.3.2. Important Notes

When you set the contents of a static frame, the contents are immediately typeset and stored in a TEX "box" until it is time to put the contents on the page. This means that if you use any information that varies throughout the document (such as the page number) the value that is current when you set the static frame's contents will be the value used.

However, if `\label` is used inside a static frame, the label information will be written

Example 3: Contination Text in Static Frames



Some text in a static frame for illustrative purposes that needs to break off and continue later on and
*Continued below right*

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

*Continued from above left*
resuming where we left off this is text in another static frame elsewhere on the page.

Next paragraph.

to the auxiliary file each time the static frame is displayed until the contents of that frame have been changed. This means that you may end up with multiply defined labels.

If you want to use cross-referencing commands, it's better to use a dynamic frame instead of a static frame.

## 2.4. Dynamic Frames

A dynamic frame is similar to a static frame except that its contents are re-typeset on each page. As with static frames, a dynamic frame may have non-rectangular content (see §3.2).

A static frame stores its contents in a "box", whereas a dynamic frame stores its contents in a token list variable. This means that verbatim content may be used in the body of the staticcontents environment but not in the body of the dynamiccontents environment (and likewise for the starred versions).

The contents must be set and will remain the same until changed. There are some special types of dynamic frames that have their content automatically set. The clear attribute may be set to automatically clear the contents whenever a page is shipped out but this may interfere with the special dynamic frames.

Special frames have special labels that should not be assigned to non-special dynamic frames: header, footer, evenheader, evenfooter (see §2.4.2), thumbtab⟨*n*⟩, thumbtabindex⟨*n*⟩, eventhumbtab⟨*n*⟩, and eventhumbtabindex⟨*n*⟩ (see §6.1).

\newdynamicframe[⟨*page-list*⟩]{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩]

*modifier:* *
preamble only

Defines a new dynamic frame. The arguments are the same as for `\newflowframe`: ⟨*width*⟩ is the width of the dynamic frame, ⟨*height*⟩ is the height of the frame, (⟨*x*⟩,⟨*y*⟩) is the position of the bottom left hand corner of the frame relative to the bottom left hand corner of the typeblock. The first optional argument, ⟨*page list*⟩, indicates the page list for which this dynamic frame should appear, and the final optional argument, ⟨*label*⟩ is a unique textual IDL which you can use to identify this dynamic frame. If no label is specified, you can refer to this frame by its unique IDN. The first dynamic frame to be defined has IDN 1, the second has IDN 2, and so on. The starred version assigns a plain border to the frame.

```
\getdynamiclabel{⟨IDN⟩}
```

Expands to the IDL for the dynamic frame identified by its IDN. An error will occur if there is no dynamic frame with the given IDN.

There is no equivalent command that will expand to the IDN. Instead you need to fetch the value with:

```
\getdynamicid{⟨cmd⟩}{⟨IDL⟩}
```

This will define the command ⟨*cmd*⟩ to expand to the IDN of the dynamic frame identified by its IDL. An error will occur if there is no dynamic frame with the given IDL.

For example, if the first dynamic frame is defined as:

```
\newdynamicframe{0.38\textwidth}{\textheight}
{0.62\textwidth}{0pt}[chaphead]
```

then later in the document:

```
The label for the first dynamic
frame is ``\getdynamic-
label{1}''.
The dynamic frame labelled ``chaphead'' has IDN
\getdynamicid{\myid}
{chaphead}\myid.
```

The label for the first dynamic frame is “chaphead”. The dynamic frame labelled “chaphead” has IDN 1.

The contents of a dynamic frame may be set with either a command or an environment but, in either case, the content can’t include any verbatim material.

\setdynamiccontents[options]{⟨*ID*⟩}{⟨*content*⟩} *modifier:* *

Sets the contents of the dynamic frame identified by ⟨*ID*⟩. This should be the frame’s IDN for the unstarred versions and the IDL for the starred version. If ⟨*options*⟩ is present, the given attributes will be applied to the frame first.

Frame attributes are global variables.

\begin{**dynamiccontents**}[⟨*options*⟩]{⟨*IDN*⟩}
⟨*content*⟩
\end{**dynamiccontents**}

Sets the contents of the dynamic frame identified by its IDN. If ⟨*options*⟩ is present, the given attributes will be applied to the frame first.

```
\begin{dynamiccontents*}[⟨options⟩]{⟨IDL⟩}
⟨content⟩
\end{dynamiccontents*}
```

The starred version is the same as dynamiccontents except that the frame is identified by its IDL.

Unlike static frames, you can append content to a dynamic frame.

```
\appenddynamiccontents[options]{⟨ID⟩}{⟨content⟩}
                                              modifier: *
```

Appends ⟨*content*⟩ to the current content of the dynamic frame identified by its IDN (unstarred version) or by its IDL (starred version).

The dynamic frame content is stored in a global variable. It's not used until the output routine positions it on the page so there's no point localising any changes.

### 2.4.1. Putting Chapter Titles in a Dynamic Frame

If `\chapter` is defined, you can make the chapter heading appear in a dynamic frame instead of in its usual place in the flow of document text. There is no facility for placing other sectional types in a dynamic frame.

This feature was originally implemented with `\dfchaphead` which hacked the internal commands commonly used in the definition of `\chapter` but this is problematic and doesn't work with some classes so version 2.0 provides a new approach. The old `\dfchaphead` command is still available but its use is now deprecated.

The chapter heading refers to the (typically large, bold) text with the chapter number (if applicable) and title that's normally placed at the start of a new page. If you want thumbtabs, see §6.1. If you want page headers and footers in a dynamic frame, see §2.4.2.

```
\ChapterInDynamic{⟨ID⟩}                                    modifier: *
```

Places the chapter heading in the dynamic frame identified by ⟨*ID*⟩, which is the frame's IDN (unstarred) or IDL (starred). This command is only available if `\chapter` is defined.

This causes each instance of `\chapter` to set the contents of the identified dynamic frame which will replace any previous content.

The flowfram package redefines all the standard sectioning commands (`\chapter` is only redefined if it's already defined). The modified `\chapter` places the original definition in the content of the chosen dynamic frame, but there are some problems with this approach. Normally, the contents of a dynamic frame aren't expanded until the output routine positions the frame on the page, and since dynamic frames are always positioned after flow frames this can cause counters and labels to be out-of-sync.

To avoid this problem, `\ChapterInDynamic` ensures that the chapter heading is first expanded by putting it in a box of the same width as the dynamic frame, but it first needs to locally disable problematic commands, such as `\clearpage`. This means that the page break usually performed by `\chapter` is now implemented with:

```
\dfchapterclearpage
```

This is be defined to use either `\clearpage` or `\cleardoublepage`, depending on the standard `openright` setting. This command is only available if `\chapter` is defined.

There are hooks available in the event that there are other problematic commands that may need adjusting. These hooks are only defined if `\chapter` is defined.

```
flowfram/chaphead/before
```

The flowfram/chaphead/before hook is used before the heading box is set but after the page has been cleared with `\dfchapterclearpage`.

```
flowfram/chaphead/after
```

The flowfram/chaphead/after hook is used after the dynamic frame's content has been set. For example, you can use this hook to automatically append content to the dynamic frame with `\appenddynamiccontents`.

```
flowfram/chaphead/box/before
```

The flowfram/chaphead/box/before hook is used inside the heading box before the heading but after the known set of problematic commands have been adjusted. Since this hook occurs within a `\parbox`, it will be scoped.

For example, this hook can be used to change the text colour of just the heading without affecting the colour of any content that is subsequently added to the frame:

```
\AddToHook{flowfram/chaphead/box/before}{\color
{blue}}
```

flowfram/chaphead/box/after

The flowfram/chaphead/box/after hook is used after the heading at the end of the box. Since this hook occurs within a \parbox, it will be scoped.

For example, this document defines a new dynamic frame called chaphead and uses it for chapter headings:

```
\newdynamicframe{0.38\textwidth}{\textheight}
{0.62\textwidth}{0pt}[chaphead]
\setdynamicframe*{chaphead}{evenx=0pt,clear}
\ChapterInDynamic*{chaphead}
```

Note that this provides a different horizontal position for even pages (specified with evenx) and the frame contents are automatically cleared on each new page (otherwise the chapter heading would show on every page).

The document class is scrbook so the KOMA-Script command that adds extra vertical space is redefined to do nothing (since the dynamic frame has its own custom dimensions):

```
\renewcommand\chapterheadstartvskip{}
```

and the chapter heading text is changed to blue:

```
\addtokomafont{chapter}{\color{blue}}
```

A mini-toc is also added to the same frame (see §6.2):

```
\appenddfminitoc*{chaphead}
```

I then defined a custom command to add a summary after the chapter heading:

```
\newcommand{\chapdesc}[1]{%
 \appenddynamiccontents*
 {chaphead}{\par \normalfont \emph{#1}}%
}
```

For example, this chapter starts with:

```
\chapter{Defining New Frames}\label
{sec:newframes}

\chapdesc
{This chapter describes how to define new frames…}
```

Modern classes, such as memoir and the KOMA-Script classes provide ways of adjusting the style of chapter headings. These can be adjusted, as in the example above. However, if you are using the standard book or report class, flowfram will adjust the definitions to make it easier to change the style.

The following commands are only provided if `\@makechapterhead` matches the definition provided by book or report.

```
\ffchapterpreheadskip
```
only if book or report class

Vertical space before the chapter heading. The default expansion is `\vspace*{50pt}` but this command is set to `\relax` in the flowfram/chaphead/box/before hook as leading vertical space is generally redundant at the start of a dynamic frame that has been specifically dedicated to holding the chapter heading.

```
\ffchapterpostheadskip
```
only if book or report class

Vertical space after the chapter heading. The default expansion is `\vspace*{40pt}`.

```
\ffchapterheadstyle
```
only if book or report class

The paragraph style for the chapter heading. The default definition sets `\parindent` to zero and sets ragged right formatting.

```
\ffchapterdefaultfont
```
only if book or report class

The default font declarations for the chapter heading. This can then be overridden by the following two commands (which both have an argument, unlike `\ffchapterdefaultfont`). The default definition is `\normalfont \bfseries`.

```
\ffchaptertitlefont{⟨text⟩}
```
only if book or report class

The font command for the chapter title (for both numbered and unnumbered chapters).

```
\ffchapternamenumfont{⟨text⟩}
```
only if book or report class

The font command for the chapter name and number (only applicable with numbered chapters). Note that the definition should scope any declarations otherwise they will also be applied to the chapter title.

```
\ffchapternamenum{⟨name⟩}{⟨number⟩}
```
only if book or report class

Typesets the chapter name (`\chaptername` or `\appendixname`) and number. The default definition expands to ⟨*name*⟩ ⟨*number*⟩.

```
\ffchapterpostnamenum
```
only if book or report class

The separator between the chapter number and the title. The default definition inserts a paragraph break and vertical space.

### 2.4.2. Putting Headers and Footers in a Dynamic Frame

Page headers and footers can be turned into dynamic frames using:

```
\makedfheaderfooter
```
preamble only

This command will create two dynamic frames with special labels header and footer. You can then move or resize these using `\setdynamicframe` (see §2.4.4). Additionally, the page style will be set to ffheadings.

If you use FlowframTk (at least v0.8.8), you can additionally have different header and footer frames for even pages. These should be assigned the special labels evenheader and evenfooter with the page list set to "Even". The dynamic frames for odd pages should be assigned the special labels header and footer with the page list set to "Odd". This may be done via the Flow Frame Wizard (see the FlowframTk documentation for further details.)

Below, header frame and footer frame refer to the dynamic frames with the special labels header and footer, but also (if applicable) to the evenheader and evenfooter frames created with FlowframTk.

You can't simply create dynamic frames with `\newdynamicframe` with these special labels for this behaviour to work. You need to use the applicable command or application function. The frame content needs to be set to the appropriate code to ensure that it reflects the current page style. It's important not to use the clear=`true` setting or otherwise clear the content of these frames as this function would then be lost.

The flowfram package provides page styles that are customised for header and footer frames, which are listed below. Remember that you can set the frame style with the style setting, but you may prefer to modify the helper commands instead (which may counteract the style setting):

```
\dfOddFooterStyle{⟨text⟩}
```

This command is used to format the footer for odd pages, where applicable. The default centres ⟨*text*⟩ within the frame.

```
\dfEvenFooterStyle{⟨text⟩}
```

This command is used to format the footer for even pages, where applicable. The default centres ⟨*text*⟩ within the frame.

```
\dfOddHeaderStyle{⟨text⟩}
```

This command is used to format the header for odd pages, where applicable. The default places ⟨*text*⟩ flush right to the edge of the frame.

```
\dfEvenHeaderStyle{⟨text⟩}
```

This command is used to format the header for even pages, where applicable. The default places ⟨*text*⟩ flush left to the edge of the frame.

```
\flowframchapterheader{⟨text⟩}
```

This command is only defined if `\chapter` is defined and picks up the `dynamic-page-style-header-font` and `dynamic-header-case` settings.

```
\flowframsectionheader{⟨text⟩}
```

The definition of this command depends on whether or not `\chapter` is defined. If it isn't defined, the `dynamic-page-style-header-font` and `dynamic-header-case` settings will apply. If `\chapter` is defined, the `dynamic-page-style-subheader-font` and `dynamic-subheader-case` settings will apply.

`\flowframheadersep` *initial: varies*

The separator used between the chapter or section number and the title (where a number is shown). If `\chapter` is defined, this is initialised to `.␣` (period followed by a space) otherwise it is initialised to `\quad`.

`\flowframheaderchapprefix` *initial: varies*

The prefix inserted before the chapter number. If `\chapter` is defined, the initial definition is `\@chapapp␣` (`\chaptername` or `\appendixname` followed by a space) otherwise the initial definition is empty.

ffempty

The default behaviour of the ffempty page style clears the standard header and footer commands (`\@oddhead`, `\@evenhead`, `\@oddfoot` and `\@evenfoot`), as per the empty style, but also hides the special header and footer frames. This action may be changed with the `dynamic-empty-page-style` setting.

For example, suppose you have created fancy header and footer frames with a background and border. It may look strange to have these frames showing with empty content, so the default setting hides them as well. However, if you still want the frames to show, then you would need to change the `dynamic-empty-page-style` option to a setting that doesn't hide them, such as `dynamic-empty-page-style=empty`.

ffplain

The default behaviour of the ffplain style is to have an empty header with `\thepage` centred in the footer. More precisely, this style sets `\@oddfoot` to:

```
\dfOddFooterStyle{\thepage}
```

and sets `\@evenfoot` to:

```
\dfEvenFooterStyle{\thepage}
```

Both `\@oddhead` and `\@evenhead` are set to empty. Additionally, if ffempty is set to hide the special header and footer frames, the ffplain style will ensure that hide and hidethis are set to `false` to counteract any previous instance of ffempty.

The ffheadings and ffmyheadings don't replicate the style of the standard headings and myheadings styles. Instead, the page number is always placed in the footer frame. If you want standard page headers and footers, there's no point switching to dynamic header and footer frames. Any minor stylist changes can be made using packages or document classes that provide support for flexible page styles.

ffheadings

The default behaviour of the ffheadings style depends on whether or not `\chapter` is defined. Note that the page number is still placed in the footer frame, as per the ffplain style. The right and left marks are placed in the applicable header frame.

This style sets `\@oddfoot` to

```
\dfOddFooterStyle{\thepage}
```

\@evenfoot to:

```
\dfEvenFooterStyle{\thepage}
```

\@oddhead to:

```
\dfOddHeaderStyle{\rightmark}
```

and \@evenhead to:

```
\dfOddHeaderStyle{\leftmark}
```

If \chapter is defined, \chaptermark is redefined to encapsulate the mark with \flowframchapterheader and \sectionmark is redefined to encapsulate the mark with \flowframsectionheader.

If \chapter is not defined, \sectionmark is redefined to encapsulate the mark with \flowframsectionheader and \subsectionmark is redefined to encapsulate the mark with \flowframsubsectionheader.

In either case, if ffempty is set to hide the special header and footer frames, the ffheadings style will ensure that hide and hidethis are set to false to counteract any previous instance of ffempty.

ffmyheadings

The ffmyheadings style is similar to the ffheadings style except that it sets the applicable mark commands to discard their argument. So, if \chapter is defined the \chaptermark and \sectionmark will be redefined to do nothing, otherwise \sectionmark and \subsectionmark will be redefined to do nothing.

If the `dynamic-page-style=adjust` setting is on, `\makedfheaderfooter` will redefine the standard page styles (empty, plain, headings and myheadings) to the analogous ffempty, ffplain, ffheadings or ffmyheadings style.

### 2.4.3. Continued Text

As with static frames, in the body of dynamiccontents or dynamiccontents* or within the text of `\setdynamiccontents`, you can move onto another dynamic frame using:

```
\continueonframe[⟨text⟩]{⟨ID⟩}
```

to simulate text flowing from one dynamic frame to another. This command must not be hidden inside the definition of another command.

The `\continueonframe` command forces a break in the text whilst at the same time trying to justify it, so overly large space may occur.

In this case, if `\continueonframe` occurs within dynamiccontents*, ⟨*ID*⟩ refers to the IDL of the other dynamic frame, otherwise it refers to the IDN of the other dynamic frame. It's not possible to continue onto a different type of frame.

For example, suppose I have defined two dynamic frames labelled `frame1` (IDN 1) and `frame2` (IDN 2), then:

```
\begin{dynamiccontents*}{frame1}
Some text in the first frame. (Let's
assume this frame is somewhere on the
left half of the page.)
\continueonframe[Continued on the right]{frame2}
This is some text in the second frame. (Let's
assume this frame is somewhere on the
right half of the same page.)
\end{dynamiccontents*}
```

is essentially the same as:

```
\begin{dynamiccontents*}{frame1}
Some text in the first frame. (Let's
assume this frame is somewhere on the
left half of the page.)
\ffcontinuedtextlayout{Continued on the right}
\end{dynamiccontents*}
\begin{dynamiccontents*}{frame2}
\ffdynamicpostcontinued{1}{2}
This is some text in the second frame. (Let's
assume this frame is somewhere on the
right half of the same page.)
\end{dynamiccontents*}
```

When `\continueonframe` is used with dynamic frames it has analogous commands to those used with static frames. If the optional ⟨*text*⟩ is omitted, the default is obtained with:

`\ffdefaultdynamiccontinuetext{`⟨*IDN1*⟩`}{`⟨*IDN2*⟩`}`
*initial:* `\ffdefaultcontinuetext`

This is like `\ffdefaultstaticcontinuetext` where the first argument is the IDN of the previous dynamic frame and the second argument is the IDN of the continuation dynamic frame. The default definition ignores its arguments and simply expands to `\ffdefaultcontinuetext`.

```
\ffdynamicpostcontinued{⟨IDN1⟩}{⟨IDN2⟩}
                         initial: \ffdefaultpostcontinued
```

This is like `\ffstaticpostcontinued` where the first argument is the IDN of the previous dynamic frame and the second argument is the IDN of the continuation dynamic frame. The default definition ignores its arguments and simply expands to `\ffdefaultpostcontinued`.

## 2.4.4. Important Notes

- Verbatim text can't be used in a dynamic frame. This includes the body of the dynamiccontents and dynamiccontents* environments.
- `\continueonframe` can't be used in the argument of any of the commands that append to the contents of a dynamic frame, such as `\appenddynamiccontents`.
- `\continueonframe` causes a change in scope so any local changes made before the command won't still be in effect after it.
- Dynamic frames are painted on the page after all the static and flow frames. If the location of a dynamic frame overlaps the location of any static or flow frames, the contents of the dynamic frame will obscure the contents of the overlapping frames.

# 3. Modifying Frame Attributes

*This chapter describes how to modify frame attributes, such as the size and location.*

Once you have defined the flow frames, static frames and dynamic frames, their attributes can be changed. The three types of frame mostly have the same set of attributes, but some are specific to a certain type.

Attributes are global. Changes usually come into effect during the output routine. Dimension and text colour are best set in the preamble, especially for flow frames and static frames.

Flow frames attributes are modified using:

```
\setflowframe{⟨ID-list⟩}{⟨key=value list⟩}                    modifier: *
```

Alternatively, you can set the same attributes for all defined flow frames with:

```
\setallflowframes{⟨key=value list⟩}
```

Static frames attributes are modified using:

```
\setstaticframe{⟨ID-list⟩}{⟨key=value list⟩}                  modifier: *
```

Alternatively, you can set the same attributes for all defined static frames with:

```
\setallstaticframes{⟨key=value list⟩}
```

Dynamic frames attributes are modified using:

```
\setdynamicframe{⟨ID-list⟩}{⟨key=value list⟩}          modifier: *
```

Alternatively, you can set the same attributes for all defined dynamic frames with:

```
\setalldynamicframes{⟨key=value list⟩}
```

In each of the above `\set`⟨*type*⟩`frame` commands, if the starred version is used then ⟨*ID-list*⟩ should be a comma-separated list of IDLs that identify the frames for which the attributes need to be applied. Note that you can't use ranges or the keywords described below with the starred versions.

In the case of the unstarred version the ⟨*ID-list*⟩ can either be a list or range of IDNs that identify the required frames or one of the keywords `all`, `odd` or `even`. The `all` keyword is essentially equivalent to using the corresponding `\setall`⟨*type*⟩`frames`. The `odd` keyword indicates that all frames with an odd IDN should be set. The `even` keyword indicates that all frames with an even IDN should be set.

i

The `all`, `odd` and `even` keywords used to identify frames should not be confused with the `all`, `odd` and `even` page list keywords.

The ⟨*key=value list*⟩ argument is a comma-separated list of ⟨*key*⟩=⟨*value*⟩ settings. **Make sure that you group the ⟨*value*⟩ if it contains one or more commas or equal signs.** Available options are listed in §3.1.

## 3.1. Set Frame Options

Some attributes are shared by all types of frame but some are only available for a particular type of frame.

### 3.1.1. Identification and Scope

label=⟨*IDL*⟩

The frame's IDL can be assigned with the label option. This may be needed when a frame is created by a helper command, such as `\Ncolumn` (see §4.3). In general, it's best to set the IDL when the frame is defined. (If you do not specify a label when you first define a frame it will be given a label identical to its IDN.)

pages=⟨*page-list*⟩ | ⟨*keyword*⟩

The frame's page list is set when the frame is defined. The default is `all`. The pages option may be used to change the page list. The value may be either ⟨*page-list*⟩ or ⟨*keyword*⟩.

Changing the page list mid-document can cause unpredictable results. You may lose text if the frame has already had content added to it on the current page. See §3.3 for switching frames on and off mid-document.

Allowed values for ⟨*keyword*⟩:

`none`

The frame should not be displayed on any page.

`all`

The frame should be displayed on all pages, unless exceptions apply.

`even`

The frame should only be displayed on even pages if the document is two-sided, unless exceptions apply.

`odd`

The frame should only be displayed on odd pages if the document is two-sided, unless exceptions apply.

Otherwise the value should be ⟨*page-list*⟩, a list or range of numbers identifying on which page or pages the frame should be shown. Ranges may be closed, such as `3-9` (which indicates pages 3 to 9 inclusive), or open-ended, such as `<3` (which indicates any page less than 3) or `>9` (which indicates any page greater than 9). The ⟨*page-list*⟩ may be a single range or a comma-separated list of individual pages or ranges, such as `<3,5-12,>20`.

Each number in the list must be a plain positive integer (1, 2, etc) not a formatted number.

The page list may also be set using:

`\flowsetpagelist{`⟨*IDN*⟩`}{`⟨*page-list*⟩ | ⟨*keyword*⟩`}`

This sets the page list for a specific flow frame identified by its IDN.

`\staticsetpagelist{`⟨*IDN*⟩`}{`⟨*page-list*⟩ | ⟨*keyword*⟩`}`

This sets the page list for a specific static frame identified by its IDN.

`\dynamicsetpagelist{`⟨*IDN*⟩`}{`⟨*page-list*⟩ | ⟨*keyword*⟩`}`

This sets the page list for a specific dynamic frame identified by its IDN.

There are no starred versions for the \⟨*type*⟩setpagelist commands and ⟨*IDN*⟩ must be a number that identifies a specific frame.

The numbers refer to the counter identified by the pages package option. With pages=relative, the numbers refer to the value of the page counter (not the formatted text obtained with \thepage). This can be ambiguous if the page counter is reset in the document. With pages=absolute, the numbers refer to the value of the absolute-page counter, which should not be reset in the document and is incremented by the output routine whenever a page is shipped out.

With pages=relative, not only can you have multiple pages with the same numeric value (for example, page i and page ii) but there's also no way of determining whether or not the next page will have the number reset. When the flowfram code in the output routine looks ahead to determine which flow frame it needs to switch to, it will assume that the value of the next page is one more than the value of the current page. For example, if the current page is "iv" (4) then the next page is assumed to have the value 5 and flowfram will select the next flow frame on that basis. If it turns out that the page counter is then reset so that \thepage is now 1, if that's outside the range of the selected flow frame then you may lose text.

excludepages=⟨*list*⟩

Certain pages can be excluded from a frame's page list. For example, if a frame's page list is set to all but it has excludepages={2,9} then the frame will be shown on all pages except for page 2 and page 9. Note that in this case the value is just a list of numbers. See also §3.3 for switching frames on and off mid-document.

It's also possible to set the exclusion list the following commands:

```
\flowsetexclusion{⟨IDN⟩}{⟨list⟩}
```

Sets the exclusion list for a flow frame.

```
\flowaddexclusion{⟨IDN⟩}{⟨list⟩}
```

Appends to the existing exclusion list for a flow frame.

```
\staticsetexclusion{⟨IDN⟩}{⟨list⟩}
```

Sets the exclusion list for a static frame.

```
\staticaddexclusion{⟨IDN⟩}{⟨list⟩}
```

Appends to the existing exclusion list for a static frame.

```
\dynamicsetexclusion{⟨IDN⟩}{⟨list⟩}
```

Sets the exclusion list for a dynamic frame.

```
\dynamicaddexclusion{⟨IDN⟩}{⟨list⟩}
```

Appends to the existing exclusion list for a dynamic frame.

There are no starred versions for these commands and the ⟨*IDN*⟩ argument should be a single number.

hide=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**

A frame can also be omitted if the boolean hide option is set. If true, this overrides the page list and omits the frame for all pages.

hidethis=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**

A frame can also be omitted if the boolean hidethis option is set. If true, this overrides the page list and omits the frame for this pages. The value will be reset when the current page is shipped out.

### 3.1.2. Content Attributes

clear=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**
static & dynamic only

This boolean option is only available for static frame and dynamic frame. If true, the frame contents will be cleared on each new page.

If you set clear=`true` for a special dynamic frame, such as header or footer, they will lose their special function.

In this document, I have used `\ChapterInDynamic` to put the chapter titles in a dynamic frame called `chaphead`. This isn't a special frame, but the contents are reset every time `\chapter` is used. I've also used `\appenddfminitoc` which then appends the chapter's mini-toc to that frame. And finally, I have a custom command that appends a brief summary. For example, this chapter is started with:

```
\chapter{Modifying Frame Attributes}\label
{sec:modattr}
\chapdesc
{This chapter describes how to modify frame attributes,
 such as the size and location.}
```

However, I only want this content visible on the first page of the chapter, so I've set the clear option for the `chaphead` frame. This ensures that the content is cleared whenever a new page starts.

I've also used `\makedfheaderfooter` and `\makethumbtabs`. This automatically defines a set of special dynamic frames. In this case, the clear option should not be set as clearing the content will remove the code that ensures the header, footer and thumbtab show the correct information.

parindent=⟨*dim expr*⟩ *initial:* **`\sdfparindent`**

This option is only available for static frames and dynamic frames. The value should be a dimension (either explicit, such as `5pt`, or a variable, such as `\sdfparindent`, or an expression, such as `1em + 0.1\linewidth`). The default is `\sdfparindent`.

The dimension won't be evaluated until the contents of the frame are typeset. For static frames this is when the frame content is set, and for dynamic frames this is every time the frame is placed on the page by the output routine.

style=⟨*value*⟩ *initial:* **`none`**

This option is only available for dynamic frames. The value may be the keyword `none` (equivalent to `@firstofone`) or the name of a declaration or a text-block command (without the leading backslash).

For example:

```
\setdynamicframe{1}{style={emph}}
```

The above will encapsulate the contents of the dynamic frame identified by IDN 1 with \emph.

Bear in mind that text-block commands such as \emph and \textbf introduce grouping which may result in subtle spacing issues. If this is a problem, try using the analogous declaration instead. For example:

```
\setdynamicframe{1}{style={em}}
```

The frame content is already scoped and no extra grouping is introduced with this method for standard rectangular frames. (Extra grouping is added for shaped frames.)

Example 1 provided a custom command:

```
\newcommand{\Hugebf}{\Huge\bfseries}
```

and the sidepanel dynamic frame had style=Hugebf set. This command could have been defined as:

```
\newcommand{\Hugebf}[1]{\textbf{\Huge #1}}
```

but if you try this out you would find that the interline spacing is much smaller. This is because the paragraph doesn't end until after the styling command so the line skip for \Huge is wrong. The text is Huge but the line spacing is normal size.

The FlowframTk dialog windows used to specify frame attributes are different: the "Style" field should be left blank for no styling or may be a command name like `emph` (as for style) but may also be a list of declarations (with optionally a final text-block command). When FlowframTk exports the information, if the "Style" field contains one or more backslashes it will use commands provided by flowframtkutils to define a custom command that can then be used in the style option.

If you are not using FlowframTk and you need more than one command to apply the style then you will need to define your own custom command and use that instead (as in Example 1).

shape=⟨*value*⟩ *initial:* **`\relax`**

This key is only available for static frames and dynamic frames and sets the paragraph shape for the frame contents. See §3.2 for further details.

valign=⟨*value*⟩ *initial:* ***varies***

This key is only available for static frames and dynamic frames and sets the vertical alignment of the content within the frame. The default is `c` for static frames and `t` for dynamic frames.

### 3.1.3. Layout Attributes

margin=⟨*value*⟩ *initial:* **`right`**

This option is only available for flow frames and indicates on which side of the flow frame the margin should be placed. The value may be one of the keywords: `inner`, `outer`, `left` or `right`.

For two-sided documents, `inner` places the margin on the left for odd pages and on the right for even pages, and `outer` places the margin on the right for odd pages and on the left for even pages.

For one-sided documents, `inner` is the same as `left` (place to the left of the frame) and `outer` is the same as `right` (place to the right of the frame).

offset=⟨*dimension*⟩ | **`compute`** *initial:* **`compute`**

Sets the frame's offset to take the border into account. The value may be a dimension or the keyword `compute`, which will compute the offset if the frame has a known border (`\fbox`, `\ovalbox`, `\Ovalbox` and `\doublebox`). If the border is unknown then the offset for `\fbox` will be used.

If the frame has a border set and it takes up extra space (caused by the padding between the border and the text and the size of the frame) then this attribute needs to be set to shift the frame so that the bottom left corner of the text area within the frame is at the designated $x$ and $y$-coordinates.

This setting is not required with FlowframTk as it calculates the offset based on the supplied inner padding.

width=⟨*dimension*⟩

Set the width of the frame to the given ⟨*dimension*⟩. This is available for all types of frame but be careful not to change this value after any content has been added to the frame.

The width can be obtained for a flow frame with:

```
\flowframewidth{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the flow frame's IDN. The width can be obtained for a static frame with:

```
\staticframewidth{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the static frame's IDN. The width can be obtained for a dynamic frame with:

```
\dynamicframewidth{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the dynamic frame's IDN. In each case, the command simply expands to the dimension.

```
height=⟨dimension⟩
```

Set the height of the frame to the given ⟨*dimension*⟩. This is available for all types of frame but be careful not to change this value after any content has been added to the frame.

The height can be obtained for a flow frame with:

```
\flowframeheight{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the flow frame's IDN. The height can be obtained for a static frame with:

```
\staticframeheight{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the static frame's IDN. The height can be obtained for a dynamic frame with:

```
\dynamicframeheight{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the dynamic frame's IDN. In each case, the command simply expands to the dimension.

```
x=⟨dimension⟩
```

Set the $x$-coordinate of the frame for all pages on which it's defined. This is available for all types of frame. Note that x=⟨*position*⟩ is equivalent to setting both oddx=⟨*position*⟩ and evenx=⟨*position*⟩.

```
y=⟨dimension⟩
```

Set the $y$-coordinate of the frame for all pages on which it's defined. This is available for all types of frame. Note that y=⟨*position*⟩ is equivalent to setting both oddy=⟨*position*⟩ and eveny=⟨*position*⟩.

```
evenx=⟨dimension⟩
```

Set the frame's $x$-coordinate for all even pages on which it's defined (only applicable if the document is two-sided). For example:

```
\setflowframe*{main}{evenx={0.4\typeblockwidth}}
\setdynamicframe*{chaphead}{evenx=0pt}
```

This changes the horizontal position of the flow frame labelled `main` and the dynamic frame labelled `chaphead` on even pages (provided the document has the two-sided setting on).

The even-page $x$-coordinate can be obtained for a flow frame with:

```
\flowframeevenx{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the flow frame's IDN. The even-page $x$-coordinate can be obtained for a static frame with:

```
\staticframeevenx{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the static frame's IDN. The even-page $x$-coordinate can be obtained for a dynamic frame with:

```
\dynamicframeevenx{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the dynamic frame's IDN. In each case, the command simply expands to the dimension.

```
eveny=⟨dimension⟩
```

Set the frame's $y$-coordinate for all even pages on which it's defined (only applicable if the document is two-sided).

The even-page $y$ co-ordinate can be obtained for a flow frame with:

```
\flowframeeveny{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the flow frame's IDN. The even-page $y$ co-ordinate can be obtained for a static frame with:

```
\staticframeeveny{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the static frame's IDN. The even-page $y$ co-ordinate can be obtained for a dynamic frame with:

```
\dynamicframeeveny{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the dynamic frame's IDN. In each case, the command simply expands to the dimension.

```
oddx=⟨dimension⟩
```

Set the frame's $x$-coordinate for all odd pages (or all pages for one-sided documents) on which it's defined.

The odd-page $x$-coordinate can be obtained for a flow frame with:

```
\flowframex{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the flow frame's IDN. The odd-page $x$-coordinate can be obtained for a static frame with:

```
\staticframex{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the static frame's IDN. The odd-page $x$-coordinate can be obtained for a dynamic frame with:

```
\dynamicframex{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the dynamic frame's IDN. In each case, the command simply expands to the dimension.

```
oddy=⟨dimension⟩
```

Set the frame's $y$-coordinate for all odd pages (or all pages for one-sided documents) on which it's defined.

The odd-page $y$-coordinate can be obtained for a flow frame with:

```
\flowframey{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the flow frame's IDN. The odd-page $y$-coordinate can be obtained for a static frame with:

```
\staticframey{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the static frame's IDN. The odd-page $y$-coordinate can be obtained for a dynamic frame with:

```
\dynamicframey{⟨IDN⟩}
```

where ⟨*IDN*⟩ is the dynamic frame's IDN. In each case, the command simply expands to the dimension.

There are also commands available to swap the odd and even co-ordinates:

```
\ffswapoddeven{⟨ID-list⟩}                    modifier: *
```

This swaps the odd and even co-ordinates for each flow frame in ⟨*ID-list*⟩. (The ⟨*ID-list*⟩ is as for `\setflowframe`.)

`\sfswapoddeven{⟨ID-list⟩}` *modifier:* `*`

This swaps the odd and even co-ordinates for each static frame in ⟨*ID-list*⟩. (The ⟨*ID-list*⟩ is as for `\setstaticframe`.)

`\dfswapoddeven{⟨ID-list⟩}` *modifier:* `*`

This swaps the odd and even co-ordinates for each dynamic frame in ⟨*ID-list*⟩. (The ⟨*ID-list*⟩ is as for `\setdynamicframe`.)

### 3.1.4. Appearance

angle=⟨*angle*⟩ *initial:* 0

Rotates the frame by ⟨*angle*⟩ degrees. Note the bounding boxes are not rotated in draft mode.

border=⟨*value*⟩ *default:* **`plain`**; *initial:* **`false`**

Sets the frame's border. The ⟨*value*⟩ may be the keyword `plain` (use a plain border, equivalent to border=`fbox`), `false` (no border) or may be the control sequence name (without the leading backslash) of a frame making command. If you set a border for a frame, you may also need to specify the offset if the default computation is incorrect.

The fancybox package provides the commands `\shadowbox`, `\doublebox`, `\ovalbox` and `\Ovalbox`, which may all be used as a border. For example, border=`ovalbox`. Remember to load the fancybox package if you want to use any of these commands as a border.

For example, to make the first static frame have an oval border:

```
\setstaticframe{1}{border=ovalbox}
```

Or you can define your own border, for example with `\fcolorbox`:

```
\newcommand{\greenyellowbox}[1]{\fcolorbox{green}
{yellow}{#1}}
\setstaticframe{1}{border=greenyellowbox}
```

In this case, the border incorporates colour so the bordercolor and backcolor settings should be `none`.

If you use FlowframTk to create the frames, remember to set the “border as shown” option if you want the shape to be the border. FlowframTk will define the border command and deal with the offset, border colour, text colour and background colour.

bordercolor= [ ⟨*colour-space*⟩ ] { ⟨*colour-specs*⟩ } *initial:* **`none`**

Sets the colour of the border. The value may be the keyword `none` (no colour change) or the colour specifications as suitable for `\color`.

This option will have no effect with borders created by FlowframTk as the colour specifications will be included in the custom border command.

border−color *alias:* bordercolor

Synonym for bordercolor.

bordercolour *alias:* bordercolor

Synonym for bordercolor.

border−colour *alias:* bordercolor

Synonym for bordercolor.

backcolor=[⟨*colour-space*⟩]{⟨*colour-specs*⟩} *initial:* **none**

Sets the colour of the frame's background. The value may be the keyword `none` (no colour change) or the colour specifications as suitable for `\color`.

If you are using FlowframTk, set the shape's fill colour instead of using this option or you may end up with blank areas in the border padding.

back−color *alias:* backcolor

Synonym for backcolor.

backcolour *alias:* backcolor

Synonym for backcolor.

back−colour *alias:* backcolor

Synonym for backcolor.

textcolor=[⟨*colour-space*⟩]{⟨*colour-specs*⟩} *initial:* **none**

Sets the colour of the frame's text. The value may be the keyword `none` (no colour change) or the colour specifications as suitable for `\color`.

It's best to keep to the default document colour for all flow frames otherwise inconsistencies can occur when paragraphs span across frames and also the footnotes will now be in the document colour. (This has changed in v2.0 as it requires too much meddling with the output routine to alter the footnote colour.) The default document colour can be set by using `\color`[⟨*color-space*⟩]{⟨*colour-specs*⟩} in the preamble.

text−color *alias:* textcolor

Synonym for textcolor.

textcolour *alias:* textcolor

Synonym for textcolor.

text−colour *alias:* textcolor

Synonym for textcolor.

### 3.1.5. HTML

The need to know where page breaks occur can make it hard for a LaTeX to HTML parser that doesn't use a LaTeX engine to generate HTML output. The arbitrary location of frames can make it hard to support the flowfram package. As from version 2.0, there is now an option specifically to help support parsers that don't have access to the output routine.

html={⟨*key=value list*⟩}

static and dynamic only

When the document is processed by LaTeX, this option simply writes information to the `aux` file. The `aux` command depends on whether or not the html option was in the preamble or in the document environment.

`\flowfram@preamble@htmlopts`{⟨*n*⟩}{⟨*key=value list*⟩}{⟨*type*⟩}{⟨*IDN*⟩}{⟨*page*⟩}{⟨*absolute-page*⟩}

This command will be written to the `aux` file if the html option was set in the preamble.

`\flowfram@doc@htmlopts`{⟨*n*⟩}{⟨*key=value list*⟩}{⟨*type*⟩}{⟨*IDN*⟩}{⟨*page*⟩}{⟨*absolute-page*⟩}

This command will be written to the `aux` file if the html option was set in the document environment.

Both commands have the same syntax. The first argument ⟨*n*⟩ is a number that starts at 1 and is incremented every time that the html option is applied. The ⟨*key=value list*⟩ option is the value of the html option. The ⟨*type*⟩ is the frame type, which will either be static or dynamic. The ⟨*IDN*⟩ argument is the frame's IDN. The final two arguments, ⟨*page*⟩ and ⟨*absolute-page*⟩, are the values of `\thepage` and `\theabsolutepage`.

It's up to the LaTeX to HTML parser to choose what options in ⟨*key=value list*⟩ to implement. The following are options recognised by the TeX Parser Library, which is used by the system that creates the HTML helpset files for some of my Java GUI applications, including FlowframTk:

show=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**

This boolean option indicates that the frame contents should be shown at this point. If this value is true, then the following options govern how the content is shown.

div=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`true`**

If true, the content should be encapsulated in a `div` element.

style=⟨*css*⟩

If div=`true`, then the `style` attribute should be set to ⟨*css*⟩.

class=⟨*css-class*⟩

If div=`true`, then the `class` attribute should be set to ⟨*css-class*⟩.

image=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**

If true, the content should be converted to an image. This typically indicates that the content is too complicated to convert to HTML.

mime−type=⟨*type*⟩

If image=`true`, the image should have the given MIME type.

alt=⟨*text*⟩

If image=`true`, the image's `alt` attribute should be set to ⟨*text*⟩.

## 3.2. Non-Rectangular Frames

As from version 1.03, it is now possible to specify non-rectangular static or dynamic frames (but not flow frames). Note that the bounding box will still appear as a rectangle despite the frame's shape setting. You may use either TEX's `\parshape` command, or the `\shapepar`/`\Shapepar` commands defined in Donald Arseneau's shapepar package (if using `\shapepar` or `\Shapepar`, remember to include the shapepar package.)

FlowframTk can compute the parameters of `\parshape` or `\shapepar`/`\Shapepar` from a shape (limitations may apply). You can either export the parameters to a TEX file that can simply be `\input` at the start of a paragraph or, if you are creating a static or dynamic frame you can set the "Shape" selector to "Parshape" or "Shapepar" as applicable. Use the "Configure TEX Settings" dialog to specify whether you want `\shapepar` or `\Shapepar` for the "Shapepar" setting. See the FlowframTk manual for further details.

The `\shapepar` or `\Shapepar` commands provide greater flexibility in the type of shape that can be used. However, be aware of the advice given in the shapepar documentation.

`\parshape`

With `\parshape` you can not have cut-outs in the middle, top or bottom of a frame, however it is possible to have cut-outs in the left or right side of the frame. When used with the shape key for static or dynamic frames, the effects of `\par` and the sectioning commands (`\section` and below) are modified to allow the paragraph shape to extend beyond a single paragraph, and to allow sectioning commands (but not `\chapter` or `\part`).

`\shapepar`/`\Shapepar`

With `\shapepar` or `\Shapepar` you may have cut-outs, but you may not have any sectioning commands, paragraph breaks, vertical spacing or mathematics. You can simulate a paragraph break using `\FLFsimpar`, but this is not recommended. The size of the shape depends on the amount of text, so the shape will expand or contract as you add or delete text. In general, `\Shapepar` is better suited for use as a frame shape than `\shapepar`. See the shapepar documentation for more details of these commands.

To restore a static or dynamic frame to its default rectangular setting, use shape`\relax`. For those unfamiliar with TEX's `\parshape` command, the syntax is:

```
\parshape=⟨n⟩ ⟨i1⟩ ⟨l1⟩ ⟨i2⟩ ⟨l2⟩ ... ⟨in⟩ ⟨ln⟩
```

where $\langle n\rangle$ is the number of ($\langle i_j\rangle$ $\langle l_j\rangle$) pairs and $\langle i_j\rangle$ specifies the left indentation for the $j$th line and $\langle l_j\rangle$ specifies the length of the $j$th line.

For example, to create a zigzag shaped static frame (whose IDL is `shapedt`):

```
\setstaticframe*{shapedt}{shape={\parshape=20
0.6\linewidth 0.4\linewidth
0.5\linewidth 0.4\linewidth
0.4\linewidth 0.4\linewidth
0.3\linewidth 0.4\linewidth
0.2\linewidth 0.4\linewidth
0.1\linewidth 0.4\linewidth
0pt 0.4\linewidth 0.1\linewidth 0.4\linewidth
0.2\linewidth 0.4\linewidth
0.3\linewidth 0.4\linewidth
0.4\linewidth 0.4\linewidth
0.5\linewidth 0.4\linewidth
0.6\linewidth 0.4\linewidth
0.5\linewidth 0.4\linewidth
0.4\linewidth 0.4\linewidth
0.3\linewidth 0.4\linewidth
0.2\linewidth 0.4\linewidth
0.1\linewidth 0.4\linewidth
0pt 0.4\linewidth 0.1\linewidth 0.4\linewidth
}}
```

This is an example of a static frame with a non-rectangular shape. This zigzag shape was specified using the shape option in \setstaticframe. The \parshape command was used to set the shape. Using the shape key rather than explicitly using \parshape means that I can have a paragraph break whilst retaining the shape **but there are limitations**.

Previously it was possible to have paragraph-breaking content such as displayed maths or the verbatim environment in a shaped paragraph. Due to various underlying changes this is no longer possible. You will now need to insert the \parshape command in the appropriate places in the frame content instead of in the shape option.

The syntax for \shapepar and \Shapepar is more complicated, see the shapepar documentation for more details. In general:

```
\shapepar{⟨shape specs⟩}
```

The shapepar package has some predefined shapes, such as \squareshape, \diamondshape, \heartshape and \nutshape.

For example, to assign a heart shape to the static frame whose IDL is shapedb:

```
\setstaticframe*{shapedb}{shape={\shape-
par\heartshape}}
```

To reset the frame back to its original rectangular shape do:

```
\setstaticframe*{shapedb}{shape=\relax}
```

The flowfram package currently does not support any other paragraph shape making commands. Any other commands would have to be used explicitly within the contents of the frame.

This example has a more complicated shape that can not be generated using TEX's `\parshape` command, so `\shapepar` was used instead. Note that this document must include the shapepar package in this instance, whereas no extra packages are required to use `\parshape`. No sectioning commands are allowed here. The shape will expand as more text is added to it. ♡

## 3.3. Switching Frames On and Off On-The-Fly

Modifying the page list (or the page exclusion list) within the document environment is a risky business. This list must be up-to-date before the output routine looks for the next frame. To make this a little easier, as from version 1.14 there are commands that help you do this.

If you want to use these commands, it's best to use the package option `pages=absolute`.

The commands described in this section update the page lists (and possibly the exclusion list) *when the output routine is next used*. They are designed to switch frames on or off either on the next page or on the next odd page. You therefore need to take care where you place these commands. For example, if you have a two-sided document and you do:

```
\dynamicswitchonnextodd{1}
\mainmatter
\chapter{Introduction}
```

This will set the dynamic frame whose IDN is 1 to be visible for the first page of chapter 1. However, if you do

```
\mainmatter
\dynamicswitchonnextodd{1}
\chapter{Introduction}
```

This will have a different effect as `\mainmatter` issues a `\cleardoublepage` so the command to switch on the dynamic frame is on the same page as the start of chapter 1. This means that the dynamic frame won't appear until the following odd page (page 3).

These commands all have the same syntax with one argument that may be a comma-separated list. The starred version uses the IDLs and the unstarred version uses the IDNs.

`\flowswitchonnext{`⟨*ID-list*⟩`}` *modifier:* *

Switch on the listed flow frames from the following page onwards.

`\flowswitchoffnext{`⟨*ID-list*⟩`}` *modifier:* *

Switch off the listed flow frames from the following page onwards.

`\flowswitchonnextodd{`⟨*ID-list*⟩`}` *modifier:* *

Switch on the listed flow frames from the next odd page onwards.

```
\flowswitchoffnextodd{⟨ID-list⟩}                    modifier: *
```

Switch off the listed flow frames from the next odd page onwards.

```
\flowswitchonnextonly{⟨ID-list⟩}                    modifier: *
```

Switch on the listed flow frames just for the following page.

```
\flowswitchoffnextonly{⟨ID-list⟩}                   modifier: *
```

Switch off the listed flow frames just for the following page.

```
\flowswitchonnextoddonly{⟨ID-list⟩}                 modifier: *
```

Switch on the listed flow frames just for the next odd page.

```
\flowswitchoffnextoddonly{⟨ID-list⟩}                modifier: *
```

Switch off the listed flow frames just for the next odd page.

```
\dynamicswitchonnext{⟨ID-list⟩}                     modifier: *
```

Switch on the listed dynamic frames from the following page onwards.

```
\dynamicswitchoffnext{⟨ID-list⟩}                    modifier: *
```

Switch off the listed dynamic frames from the following page onwards.

`\dynamicswitchonnextodd{⟨ID-list⟩}` *modifier:* *

Switch on the listed dynamic frames from the next odd page onwards.

`\dynamicswitchoffnextodd{⟨ID-list⟩}` *modifier:* *

Switch off the listed dynamic frames from the next odd page onwards.

`\dynamicswitchonnextonly{⟨ID-list⟩}` *modifier:* *

Switch on the listed dynamic frames just for the following page.

`\dynamicswitchoffnextonly{⟨ID-list⟩}` *modifier:* *

Switch off the listed dynamic frames just for the following page.

`\dynamicswitchonnextoddonly{⟨ID-list⟩}` *modifier:* *

Switch on the listed dynamic frames just for the next odd page.

`\dynamicswitchoffnextoddonly{⟨ID-list⟩}` *modifier:* *

Switch off the listed dynamic frames just for the next odd page.

`\staticswitchonnext{⟨ID-list⟩}` *modifier:* *

Switch on the listed static frames from the following page onwards.

`\staticswitchoffnext{⟨ID-list⟩}` *modifier:* *

Switch off the listed static frames from the following page onwards.

`\staticswitchonnextodd{⟨ID-list⟩}` *modifier:* *

Switch on the listed static frames from the next odd page onwards.

`\staticswitchoffnextodd{⟨ID-list⟩}` *modifier:* *

Switch off the listed static frames from the next odd page onwards.

`\staticswitchonnextonly{⟨ID-list⟩}` *modifier:* *

Switch on the listed static frames just for the following page.

`\staticswitchoffnextonly{⟨ID-list⟩}` *modifier:* *

Switch off the listed static frames just for the following page.

`\staticswitchonnextoddonly{⟨ID-list⟩}` *modifier:* *

Switch on the listed static frames just for the next odd page.

`\staticswitchoffnextoddonly{⟨ID-list⟩}` *modifier:* *

Switch off the listed static frames just for the next odd page.

The flowfram package comes with a sample file `sample-pages.tex` that uses some of these commands.

# 4. Locations and Dimensions

*This chapter describes some of the commands provided to determine the locations and dimensions of frames.*

This chapter describes some of the commands available that can be used to determine the locations and dimensions of frames. See the accompanying document `flowfram-code.pdf` for more details of these commands or for other commands not listed here.

## 4.1. Determining the Location of the Typeblock

As mentioned earlier, when you create new frames, you must specify their location relative to the typeblock, but what if you want to position a frame a set distance from the edge of the paper? The flowfram package provides the following commands that compute the distance from the typeblock to the paper boundary. Since odd and even pages may have a different offset if `\oddsidemargin` and `\evensidemargin` have different values, there are separate commands for horizontal measurements on odd and even pages.

```
\computeleftedgeodd{⟨dim var⟩}
```

Computes the position of the left edge of the page, relative to the left side of the typeblock, for odd pages, and stores the result in the dimension variable (length register) ⟨*dim var*⟩.

```
\computeleftedgeeven{⟨dim var⟩}
```

Computes the position of the left edge of the page, relative to the left side of the typeblock, for even pages, and stores the result in the dimension variable (length register) ⟨*dim var*⟩.

```
\computetopedge{⟨dim var⟩}
```

Computes the top edge of the page, relative to the bottom of the typeblock, and stores the result in the dimension variable (length register) ⟨*dim var*⟩.

```
\computebottomedge{⟨dim var⟩}
```

Computes the bottom edge of the page, relative to the bottom of the typeblock, and stores the result in the dimension variable (length register) ⟨*dim var*⟩.

```
\computerightedgeodd{⟨dim var⟩}
```

Computes the position of the right edge of the page, relative to the left side of the typeblock, for odd pages, and store the result in the dimension variable (length register) ⟨*dim var*⟩.

```
\computerightedgeeven{⟨dim var⟩}
```

Computes the position of the right edge of the page, relative to the left side of the typeblock, for even pages, and store the result in the dimension variable (length register) ⟨*dim var*⟩.

For example, if you want to create a frame whose bottom left corner is one inch from the left edge of the page and half an inch from the bottom edge of the page (this assumes odd and even pages have the same margins):

```
% define two new lengths to store the x and y coords
\newlength{\myX}
\newlength{\myY}
% compute the distance from typeblock to the paper edge
\computeleftedgeodd{\myX}
\computebottomedge{\myY}
% Add the absolute co-ordinates to get co-ordinates
% relative to the typeblock
\addtolength{\myX}{1in}
\addtolength{\myY}{0.5in}
```

## 4.2. Determining the Dimensions and Locations of Frames

It is possible to determine the dimensions and locations of a frame using one of the following commands. For each command, the starred version identifies the frame by its IDL and the unstarred version identifies it by its IDN. The results are stored in the length registers `\ffareax` (the $x$-coordinate), `\ffareay` (the $y$-coordinate), `\ffareawidth` (the frame's width) and `\ffareaheight` (the frame's height).

```
\getstaticbounds{⟨ID⟩}                              modifier: *
```

Gets the bounds for the static frame identified by ⟨*ID*⟩.

```
\getflowbounds{⟨ID⟩}                                modifier: *
```

Gets the bounds for the flow frame identified by ⟨*ID*⟩.

```
\getdynamicbounds{⟨ID⟩}                             modifier: *
```

Gets the bounds for the dynamic frame identified by ⟨*ID*⟩.

For other related commands, see the section "Determining Dimensions and Locations" in the accompanying document `flowfram-code.pdf`.

## 4.3. Relative Locations

If you want to a textual description to appear in the document of the relative location of one frame from another you can use:

`\relativeframelocation{⟨type1⟩}{⟨ID1⟩}{⟨type2⟩}{⟨ID2⟩}`
*modifier:* *

This produces a textual description of the relative location of the ⟨*type1*⟩ frame identified by ⟨*ID1*⟩ to the ⟨*type2*⟩ frame identified by ⟨*ID2*⟩ (the starred version references the frames by their IDL and the unstarred version references them by their IDN).

The relative locations are taken for the current page, regardless of whether either of the two frames are displayed on that page. If the current page is odd, then the frame settings for odd pages will be compared, otherwise the frame settings for even pages will be compared. However remember that the first paragraph of each page retains the settings in place at the start of the paragraph, so if the frames have different locations for odd and even pages, then `\relativeframelocation` may use the wrong page settings if it is used at the start of the page. You may prefer to use `\SaveRelativeFrameLocation` instead.

The text produced will be obtained with one of the following commands, which may be added to a language hook if required.

`\FFaboveleft` *initial:* `above left`

Text produced if the first frame is above left of the second frame.

`\FFaboveright` *initial:* `above right`

Text produced if the first frame is above right of the second frame.

```
\FFabove                                        initial: above
```

Text produced if the first frame is above the second frame (but not to the right or left).

```
\FFbelowleft                               initial: below left
```

Text produced if the first frame is below left of the second frame.

```
\FFbelowright                             initial: below right
```

Text produced if the first frame is below right of the second frame.

```
\FFbelow                                        initial: below
```

Text produced if the first frame is below the second frame (but not to the right or left).

```
\FFleft                                   initial: on the left
```

Text produced if the first frame is to the left of the second frame (but not above or below).

```
\FFright                                 initial: on the right
```

Text produced if the first frame is to the right of the second frame (but not above or below).

```
\FFoverlap                                    initial: overlap
```

Text produced if both frames overlap.

Above

A frame is considered to be above another frame if the bottom edge of the first frame is higher than the top edge of the second frame.

Below

A frame is considered to be below another frame if the top edge of the first frame is lower than the bottom edge of the second frame.

Left

A frame is considered to be to the left of another frame if the right edge of the first frame is to the left of the left edge of the second frame.

Right

A frame is considered to be to the right of another frame if the left edge of the first frame is to the right of the right edge of the second frame.

There are some short cut commands for frames of the same type:

`\reldynamicloc{⟨ID1⟩}{⟨ID2⟩}` *modifier:* *

A shortcut for `\relativeframelocation` where both frame types are dynamic. The starred version references the dynamic frames by their IDL and the unstarred version references them by their IDN.

`\relstaticloc{⟨ID1⟩}{⟨ID2⟩}` *modifier:* *

A shortcut for `\relativeframelocation` where both frame types are static. The starred version references the static frames by their IDL and the unstarred version references them by their IDN.

```
\relflowloc{⟨ID1⟩}{⟨ID2⟩}                          modifier: *
```

A shortcut for `\relativeframelocation` where both frame types are flow. The starred version references the flow frames by their IDL and the unstarred version references them by their IDN.

These commands may be used in the optional argument of `\continueonframe` for frames that are on the same page. For example:

```
\begin{dynamiccontents}{1}
Some text in the first dynamic frame that goes on for
quite a bit longer than this example.
\continueonframe[continued \reldynamicloc{2}{1}]{2}
This text is in the second dynamic frame which is
somewhere on the same page.
\end{dynamiccontents}
```

Alternatively, they may be used in the default continued text, as in Example 3.

An alternative approach is to save the query in the `aux` file to ensure that the page counter is correct:

```
\SaveRelativeFrameLocation{⟨label⟩}{⟨type1⟩}{⟨ID1⟩}
{⟨type2⟩}{⟨ID2⟩}                                   modifier: *
```

This saves the information and defers the calculation until the start of the next LaTeX run. This means that the page number for the location in the document where `\SaveRelativeFrameLocation` was used will now be correct. The ⟨*label*⟩ argument is a label that may be used to reference the information and has the usual restrictions that apply to labels. The remaining arguments are as for `\relativeframelocation`.

On the next run, the information can be retrieved with:

```
\RefSavedRelativeLocation{⟨label⟩}
```

where ⟨*label*⟩ should match the label argument supplied to `\SaveRelativeFrameLocation`. This will produce the usual undefined reference double question mark `??` and a warning on the first run.

For example, this document defined a flow frame labelled `main` and a dynamic frame labelled `chaphead` which is used to display the chapter headings. The following code:

```
The dynamic frame is
\SaveRelativeFrameLocation*
 {chapheadmain}% label
 {dynamic}{chaphead}{flow}{main}% frames
\RefSavedRelativeLocation{chapheadmain}
of the flow frame.
```

produces:

The dynamic frame is on the left of the flow frame.

The result from the calculation is saved as one of the placeholder commands, `\FFabove` etc. You can test for any of these commands with:

```
\IfSavedRelativeLocationEq{⟨label⟩}{⟨cmd⟩}{⟨true⟩}{⟨false⟩}
```

This will expand to ⟨*true*⟩ if the given command matches or false if it doesn't match or if the label doesn't exist. In this case, no warning occurs for an undefined label. The second argument must be the actual placeholder command, not an expansion of it.

You will only be able to find out an exact match with `\IfSavedRelativeLocationEq`. For example, `\FFabove` will only match if the first frame is above the second frame but not to the right or to the left (that is, their horizontal positions overlap).

If you simply need to know more generally, you can use the following commands:

```
\IfSavedRelativeLocationAbove{⟨label⟩}{⟨true⟩}{⟨false⟩}
```

Expands to ⟨*true*⟩, if the first frame is above the second frame without regard to their horizontal positions.

```
\IfSavedRelativeLocationBelow{⟨label⟩}{⟨true⟩}{⟨false⟩}
```

Expands to ⟨*true*⟩, if the first frame is below the second frame without regard to their horizontal positions.

```
\IfSavedRelativeLocationLeft{⟨label⟩}{⟨true⟩}{⟨false⟩}
```

Expands to ⟨*true*⟩, if the first frame is left of the second frame without regard to their vertical positions.

```
\IfSavedRelativeLocationRight{⟨label⟩}{⟨true⟩}{⟨false⟩}
```

Expands to ⟨*true*⟩, if the first frame is right of the second frame without regard to their vertical positions.

For additional commands that determine the relative location of one frame from another, see the section “Determining the relative location of one frame from another” in the accompanying document `flowfram-code.pdf`.

# 5. Predefined Layouts

*This chapter describes commands that create frames arranged in a predefined layout.*

The flowfram package has a number of commands which create frames in a predefined layout. These commands may only be used in the preamble. The distance between the flow frames created with these commands is given by the standard `\columnsep` length register.

```
\adjustcolsep
```

If you have more than two columns and you want space for any marginal notes (created with `\marginpar`) that may occur in the middle columns, you can use `\adjustcolsep` to automatically adjust the value of `\columnsep` to ensure that there's sufficient room. This is simply a shortcut that sets `\columnsep` to twice its existing value plus the value of `\marginparwidth`.

If you want to change the value of `\columnsep`, make sure you do it before using the column commands. Changing it afterwards will have no effect on any flow frames that have already been created.

## 5.1. Column Styles

The standard LaTeX commands `\onecolumn` and `\twocolumn` are redefined to create one or two flow frames that fill the entire typeblock separated from each other (in the case of `\twocolumn`) by a gap of width `\columnsep`.

The final ⟨*label*⟩ optional argument the commands described below is retained for backward-compatibility. In early versions of flowfram that option occurred by chance rather than design as the some of the commands happened to end with `\newflow-frame` which picked up a following optional argument.

The syntax for these commands is slightly different:

```
\onecolumn[⟨page-list⟩][⟨label⟩]
```
redefined by flowfram

Creates a flow frame that spans the typeblock with the given label and page list. If you want to specify a label, you will need to also supply the page list. The default is `all` for ⟨*page-list*⟩ and the frame's IDN for ⟨*label*⟩.

```
\twocolumn[⟨page-list⟩][⟨label⟩]
```
redefined by flowfram

Creates two flow frames that fit in the typeblock separated by the distance given by `\columnsep`. The first argument ⟨*page-list*⟩ is as for `\onecolumn`. The second argument may either be a single label, in which case it applies to the second column, or (v2.0+) two comma-separated labels where the first one should apply to the first flow frame and the second should apply to the second flow frame. For example:

```
\twocolumn[all][frame1,frame2]
```

The `RL` package option will define the columns starting from the right, so the right-hand column will be filled first. The default `LR` package option will define the columns starting from the left, so the left-hand column will be filled first. Likewise for all the other commands described in this section that create more than one flow frame.

The height of these flow frames may not be exactly as high as the typeblock, as their height is adjusted to make them an integer multiple of `\baselineskip`.

4 Example 4 shows a two-column document in draft mode where the flow frames were created with \twocolumn. Remember that the page geometry needs to be set first. The lipsum package is used to provide dummy text to pad out the document.

```
\documentclass[12pt]{article}
\usepackage[a4paper,margin=1in]{geometry}
\usepackage[draft]{flowfram}
\usepackage{lipsum}
\twocolumn% create two flow frames
\setflowframe{2}{border=plain}
% add border to second frame
\begin{document}
\lipsum[1-3]
\end{document}
```

Draft mode draws grey rectangles to show the location of the typeblock, frames and the margin area (where the text of \marginpar is placed). Note that the top border of both flow frames is slightly below the top border of the typeblock. This is due to the automatic vertical adjustment.

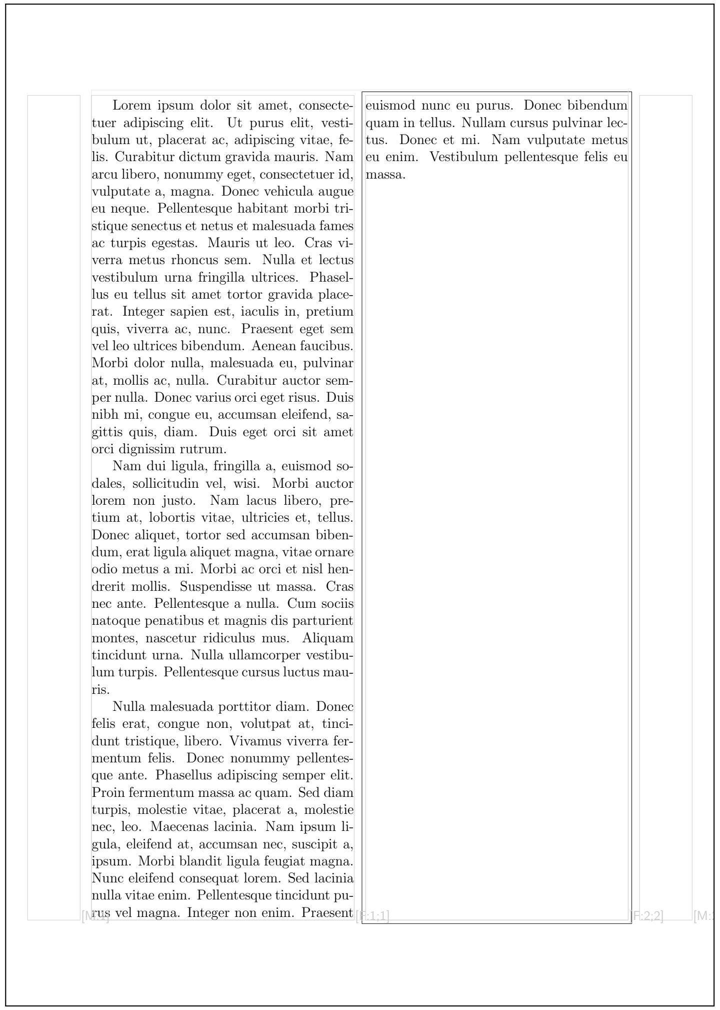
The plain border around the second flow frame is wider than the actual frame area. In this case, the flowfram package knows the appropriate offset for the plain border. Custom borders may require the use of the offset option. If you are creating the frames with FlowframTk then there's an area in which you can specify the inner padding between the border and the area for the text.

You can switch off the automatic height adjustment using the command:

```
\ffvadjustfalse
```

or switch it back on again with:

Example 4: Two Columns in Draft Mode



Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

```
\ffvadjusttrue
```

You can test the current setting with the conditional:

```
\ifffvadjust ⟨true⟩\else ⟨false⟩\fi                initial: \iftrue
```

This setting applies to all the column commands described in this section.

```
\Ncolumn[⟨page-list⟩]{⟨n⟩}[⟨label⟩]
                                                         preamble only
```

Creates ⟨*n*⟩ column flow frames, each separated by a distance of `\columnsep`, adjusting their height according to the `\ifffvadjust` setting. This is similar to `\twocolumn` but the final optional argument is a single label for the last flow frame to be created.

```
\onecolumninarea[⟨page-list⟩]{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}
[⟨label⟩]
                                                         preamble only
```

This creates a single flow frame to fill the given area, adjusting its height according to the `\ifffvadjust` setting.

```
\twocolumninarea[⟨page-list⟩]{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}
[⟨labels⟩]
                                                         preamble only
```

Creates two column flow frames separated by a distance of `\columnsep` filling the area specified, adjusting their height according to the `\ifffvadjust` setting.

`\Ncolumninarea[⟨page-list⟩]{⟨n⟩}{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨labels⟩]`

preamble only

This is a more general form of `\twocolumninarea` making ⟨*n*⟩ flow frames instead of two. The final optional argument is a single label for the last flow frame to be created.

## 5.2. Column Styles with an Additional Frame

As well as the column-style flow frames defined in §5.1, it is also possible to define ⟨*n*⟩ columns with an additional frame spanning either above or below them. There will be a vertical gap of approximately `\vcolumnsep` between the columns and the extra frame.

The height of the vertical gap may not be exactly `\vcolumnsep`, as the flow frames will be adjusted (if the default `\ffvadjusttrue` setting is in effect) so that their height is an integer multiple of `\baselineskip`, which may increase the gap.

In each of the following definitions, the argument ⟨*page-list*⟩ is the page list for which the frames are defined, ⟨*n*⟩ is the number of columns required, ⟨*type*⟩ is the type of frame to go above or below the columns (this may be one of: `flow`, `static` or `dynamic`). The area in which the new frames should fill is defined by ⟨*width*⟩, ⟨*height*⟩ (the width and height of the area) and ⟨*x*⟩, ⟨*y*⟩ (the position of the bottom left hand corner of the area relative to the bottom left hand corner of the typeblock.)

The height of the additional frame at the top or bottom of the columns is given by ⟨*top-height*⟩ and ⟨*bottom-height*⟩, respectively.

```
\onecolumntopinarea[⟨page-list⟩]{⟨type⟩}{⟨top-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                                    preamble only
```

Creates one flow frame with a ⟨*type*⟩ frame above it, filling the area specified. If the ⟨*label*⟩ option is supplied, it will be set as the label of the flow frame.

```
\twocolumntopinarea[⟨page-list⟩]{⟨type⟩}{⟨top-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                                    preamble only
```

Creates two column-style flow frames with a ⟨*type*⟩ frame above them, filling the area specified. If the ⟨*label*⟩ option is supplied, it will be set as the label of the last flow frame to be created.

```
\Ncolumntopinarea[⟨page-list⟩]{⟨type⟩}{⟨n⟩}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                                    preamble only
```

Creates ⟨*n*⟩ column-style flow frames with a ⟨*type*⟩ frame above them, filling the area specified. If the ⟨*label*⟩ option is supplied, it will be set as the label of the last flow frame to be created.

```
\onecolumnbottominarea[⟨page-list⟩]{⟨type⟩}{⟨bottom-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                                    preamble only
```

Creates one flow frame with a ⟨*type*⟩ frame underneath it, filling the area specified. If the ⟨*label*⟩ option is supplied, it will be set as the label of the flow frame.

```
\twocolumnbottominarea[⟨page-list⟩]{⟨type⟩}{⟨bottom-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                                preamble only
```

Creates two column-style flow frames with a ⟨*type*⟩ frame below them, filling the area specified. If the ⟨*label*⟩ option is supplied, it will be set as the label of the last flow frame to be created.

```
\Ncolumnbottominarea[⟨page-list⟩]{⟨type⟩}{⟨n⟩}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                                preamble only
```

Creates ⟨*n*⟩ column-style flow frames with a ⟨*type*⟩ frame below them, filling the area specified. If the ⟨*label*⟩ option is supplied, it will be set as the label of the last flow frame to be created.

The following commands are special cases of the above:

```
\onecolumnStopinarea[⟨page-list⟩]{⟨top-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                                preamble only
```

This is equivalent to:

```
\onecolumntopinarea[⟨page-list⟩]{static}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\onecolumnDtopinarea[⟨page-list⟩]{⟨top-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\onecolumntopinarea[⟨page-list⟩]{dynamic}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\onecolumntop[⟨page-list⟩]{⟨type⟩}{⟨top-height⟩}[⟨label⟩]
```
preamble only

As `\onecolumntopinarea` where the area is given by the typeblock.

```
\onecolumnStop[⟨page-list⟩]{⟨top-height⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\onecolumntop[⟨page-list⟩]{static}{⟨top-height⟩}[⟨label⟩]
```

```
\onecolumnDtop[⟨page-list⟩]{⟨top-height⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\onecolumntop[⟨page-list⟩]{dynamic}{⟨top-height⟩}[⟨label⟩]
```

```
\twocolumnStopinarea[⟨page-list⟩]{⟨top-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

preamble only

This is equivalent to:

```
\twocolumntopinarea[⟨page-list⟩]{static}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\twocolumnDtopinarea[⟨page-list⟩]{⟨top-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

preamble only

This is equivalent to:

```
\twocolumntopinarea[⟨page-list⟩]{dynamic}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\twocolumntop[⟨page-list⟩]{⟨type⟩}{⟨top-height⟩}[⟨label⟩]
```

preamble only

As `\twocolumntopinarea` where the area is the entire typeblock.

```
\twocolumnStop[⟨page-list⟩]{⟨top-height⟩}[⟨label⟩]
                                                  preamble only
```

This is equivalent to:

```
\twocolumntop[⟨page-list⟩]{static}{⟨top-height⟩}[⟨label⟩]
```

```
\twocolumnDtop[⟨page-list⟩]{⟨top-height⟩}[⟨label⟩]
                                                  preamble only
```

This is equivalent to:

```
\twocolumntop[⟨page-list⟩]{dynamic}{⟨top-height⟩}[⟨label⟩]
```

```
\NcolumnStopinarea[⟨page-list⟩]{⟨n⟩}{⟨top-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                                  preamble only
```

This is equivalent to:

```
\Ncolumntopinarea[⟨page-list⟩]{static}{⟨n⟩}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\NcolumnDtopinarea[⟨page-list⟩]{⟨n⟩}{⟨top-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\Ncolumntopinarea[⟨page-list⟩]{dynamic}{⟨n⟩}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\Ncolumntop[⟨page-list⟩]{⟨type⟩}{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
```
preamble only

As `\Ncolumntopinarea` where the area is the entire typeblock.

```
\NcolumnStop[⟨page-list⟩]{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\Ncolumntop[⟨page-list⟩]{static}{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
```

```
\NcolumnDtop[⟨page-list⟩]{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\Ncolumntop[⟨page-list⟩]{dynamic}{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
```

```
\onecolumnSbottominarea[⟨page-list⟩]{⟨bottom-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\onecolumnbottominarea[⟨page-list⟩]{static}{⟨bottom-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\onecolumnDbottominarea[⟨page-list⟩]{⟨bottom-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\onecolumnbottominarea[⟨page-list⟩]{dynamic}{⟨bottom-
height⟩}{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\onecolumnbottom[⟨page-list⟩]{⟨type⟩}{⟨bottom-height⟩}[⟨label⟩]
```
preamble only

As `\onecolumnbottominarea` where the area is the entire typeblock.

```
\onecolumnSbottom[⟨page-list⟩]{⟨bottom-height⟩}[⟨label⟩]
                                              preamble only
```

This is equivalent to:

```
\onecolumnbottom[⟨page-list⟩]{static}{⟨bottom-height⟩}
[⟨label⟩]
```

```
\onecolumnDbottom[⟨page-list⟩]{⟨bottom-height⟩}[⟨label⟩]
                                              preamble only
```

This is equivalent to:

```
\onecolumnbottom[⟨page-list⟩]{dynamic}{⟨bottom-height⟩}
[⟨label⟩]
```

```
\twocolumnSbottominarea[⟨page-list⟩]{⟨bottom-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
                                              preamble only
```

This is equivalent to:

```
\twocolumnbottominarea[⟨page-list⟩]{static}{⟨bottom-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\twocolumnDbottominarea[⟨page-list⟩]{⟨bottom-height⟩}{⟨width⟩}
{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\twocolumnbottominarea[⟨page-list⟩]{dynamic}{⟨bottom-
height⟩}{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\twocolumnbottom[⟨page-list⟩]{⟨type⟩}{⟨bottom-height⟩}[⟨label⟩]
```
preamble only

As `\twocolumnbottominarea` where the area is the entire typeblock.

```
\twocolumnSbottom[⟨page-list⟩]{⟨bottom-height⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\twocolumnbottom[⟨page-list⟩]{static}{⟨bottom-height⟩}
[⟨label⟩]
```

```
\twocolumnDbottom[⟨page-list⟩]{⟨bottom-height⟩}[⟨label⟩]
```
preamble only

This is equivalent to:

```
\twocolumnbottom[⟨page-list⟩]{dynamic}{⟨bottom-height⟩}
[⟨label⟩]
```

```
\NcolumnSbottominarea[⟨page-list⟩]{⟨n⟩}{⟨bottom-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

preamble only

This is equivalent to:

```
\Ncolumnbottominarea[⟨page-list⟩]{static}{⟨n⟩}{⟨top-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\NcolumnDbottominarea[⟨page-list⟩]{⟨n⟩}{⟨bottom-height⟩}
{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

preamble only

This is equivalent to:

```
\Ncolumnbottominarea[⟨page-list⟩]{dynamic}{⟨n⟩}{⟨top-
height⟩}{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}[⟨label⟩]
```

```
\Ncolumnbottom[⟨page-list⟩]{⟨type⟩}{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
```

preamble only

As `\Ncolumnbottominarea` where the area is the entire typeblock.

```
\NcolumnSbottom[⟨page-list⟩]{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
                                                    preamble only
```

This is equivalent to:

```
\Ncolumnbottom[⟨page-list⟩]{static}{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
```

```
\NcolumnDbottom[⟨page-list⟩]{⟨n⟩}{⟨top-height⟩}[⟨label⟩]
                                                    preamble only
```

This is equivalent to:

```
\Ncolumnbottom[⟨page-list⟩]{dynamic}{⟨n⟩}{⟨top-height⟩}
[⟨label⟩]
```

The top/bottom frame for each of the above commands is defined with:

```
\newframe[⟨page-list⟩]{⟨frame-type⟩}{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}
[⟨label⟩]                                              modifier: *
                                                    preamble only
```

This is essentially the same as:

```
\new⟨frame-type⟩frame[⟨page-list⟩]{⟨width⟩}{⟨height⟩}{⟨x⟩}{⟨y⟩}
[⟨label⟩]
```

## 5.3. Right to Left Columns

The default behaviour for the column style commands defined in §§5.1 & 5.2 above is to create the flow frames from left to right. However if you are typesetting from right to left, you will probably prefer to define the flow frames in the reverse order. This can be done via the package option RL. Alternatively you can use the command:

```
\lefttorightcolumnsfalse
```

before using commands such as \twocolumn or \Ncolumn. To switch back to left-to-right columns use the LR option or:

```
\lefttorightcolumnstrue
```

You can test the current setting with the conditional:

```
\iflefttorightcolumns ⟨true⟩\else ⟨false⟩\fi
                                        initial: \iftrue
```

Changing the setting only affects new flow frames created with subsequent column commands and does not affect any flow frames that have already been created or any flow frames created explicitly with \newflowframe.

Remember that it's the order of definition that determines the next flow frame to be selected by the output routine (that's valid for the current page). This setting simply switches round the order of definition in the column-style shortcut commands.

## 5.4. Backdrop Effects

Static frames can be used to produce a backdrop, and there are a number of commands that define static frames and automatically set the backcolor attribute to create a rectangular patch of colour.

In the following definitions, ⟨*page-list*⟩ is the page list for which those static frames are defined (`all` is the default). The arguments ⟨*L1*⟩, ..., ⟨*Ln*⟩ are the labels to assign to the 1st, ..., ⟨*n*⟩th new frame that will be created by the applicable command.

The arguments ⟨*C1*⟩, ..., ⟨*Cn*⟩ are the colour specifications for the ⟨*L1*⟩, ..., ⟨*Ln*⟩ frame. The colour specification must be completely enclosed in the ⟨*C1*⟩ etc argument braces. For example `{[rgb]{1,0,1}}` (one argument) not `[rgb]{1,0,1}` (optional argument followed by mandatory argument).

For the horizontal stripes: ⟨*x-offset*⟩ is the amount by which the frames should be shifted horizontally (0pt by default), ⟨*W1*⟩ is the width of frame ⟨*L1*⟩, and so on up to ⟨*Wn*⟩, the width of frame ⟨*Ln*⟩.

For the vertical stripes: ⟨*y-offset*⟩ is the amount by which the frames should be shifted vertically (0pt by default), ⟨*H1*⟩ is the height of frame ⟨*L1*⟩, and so on up to ⟨*Hn*⟩, the height of frame ⟨*Ln*⟩.

i

Unlike the earlier commands, these commands are all relative to the actual page, not the typeblock. So an ⟨*x-offset*⟩ of 0pt indicates the first vertical frame is flush with the left hand edge of the page, and a ⟨*y-offset*⟩ of 0pt indicates the first horizontal frame is flush with the bottom edge of the page. This is because backdrops tend to span the entire page, not just the typeblock.

### 5.4.1. Vertical stripe effects

\vtwotone[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*W2*⟩}{⟨*C2*⟩}{⟨*L2*⟩}

Defines two static frames to create a two-tone vertical stripe effect. (This command was used to create the coloured background on the title page of this document.)

\vNtone[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*n*⟩}{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}...{⟨*Wn*⟩}{⟨*Cn*⟩}{⟨*Ln*⟩}

This is similar to `\vtwotone` but with ⟨*n*⟩ static frames instead of two.

\vtwotonebottom[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*H*⟩}{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*W2*⟩}{⟨*C2*⟩}{⟨*L2*⟩}

This is similar to `\vtwotone` but the static frames are only ⟨*H*⟩ high, instead of the entire height of the page. The frames are aligned along the bottom edge of the page.

\vtwotonetop[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*H*⟩}{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*W2*⟩}{⟨*C2*⟩}{⟨*L2*⟩}

This is similar to `\vtwotone` but the static frames are only ⟨*H*⟩ high, instead of the entire height of the page. The frames are aligned along the top edge of the page. (This command was used to create the border effect along the top of every page in this document. Two `\vtwotonetop` commands were used, one for the even pages, and the other for the odd pages.)

`\vNtonebottom[⟨page-list⟩][⟨x-offset⟩]{⟨n⟩}{⟨H⟩}{⟨W1⟩}{⟨C1⟩}{⟨L1⟩}...{⟨Wn⟩}{⟨Cn⟩}{⟨Ln⟩}`

This is a general version of `\vtwotonebottom` for ⟨*n*⟩ frames instead of two.

`\vNtonetop[⟨page-list⟩][⟨x-offset⟩]{⟨n⟩}{⟨H⟩}{⟨W1⟩}{⟨C1⟩}{⟨L1⟩}...{⟨Wn⟩}{⟨Cn⟩}{⟨Ln⟩}`

This is a general version of `\vtwotonetop` for ⟨*n*⟩ frames instead of two.

### 5.4.2. Horizontal stripe effect

`\htwotone[⟨page-list⟩][⟨y-offset⟩]{⟨H1⟩}{⟨C1⟩}{⟨L1⟩}{⟨H2⟩}{⟨C2⟩}{⟨L2⟩}`

This defines two static frames to create a two-tone horizontal stripe effect.

`\hNtone[⟨page-list⟩][⟨y-offset⟩]{⟨n⟩}{⟨H1⟩}{⟨C1⟩}{⟨L1⟩}...{⟨Hn⟩}{⟨Cn⟩}{⟨Ln⟩}`

This is similar to `\htwotone` but with ⟨*n*⟩ static frames instead of two.

`\htwotoneleft[⟨page-list⟩][⟨y-offset⟩]{⟨W⟩}{⟨H1⟩}{⟨C1⟩}{⟨L1⟩}{⟨H2⟩}{⟨C2⟩}{⟨L2⟩}`

This is similar to `\htwotone` but the statics are only ⟨*W*⟩ wide, instead of the entire width of the page. The frames are aligned along the left edge of the page.

`\htwotoneright[⟨page-list⟩][⟨y-offset⟩]{⟨W⟩}{⟨H1⟩}{⟨C1⟩}{⟨L1⟩}{⟨H2⟩}{⟨C2⟩}{⟨L2⟩}`

This is similar to `\htwotone` but the statics are only ⟨*W*⟩ wide, instead of the entire width of the page. The frames are aligned along the right edge of the page.

`\hNtoneleft[⟨page-list⟩][⟨y-offset⟩]{⟨n⟩}{⟨W⟩}{⟨H1⟩}{⟨C1⟩}{⟨L1⟩}...{⟨Hn⟩}{⟨Cn⟩}{⟨Ln⟩}`

This is a general version of `\htwotoneleft` for ⟨*n*⟩ frames instead of two.

`\hNtoneright[⟨page-list⟩][⟨y-offset⟩]{⟨n⟩}{⟨W⟩}{⟨H1⟩}{⟨C1⟩}{⟨L1⟩}...{⟨Hn⟩}{⟨Cn⟩}{⟨Ln⟩}`

This is a general version of `\htwotoneright` for ⟨*n*⟩ frames instead of two.

### 5.4.3. Background Frame

`\makebackgroundframe[⟨page-list⟩][⟨label⟩]`

Creates a single static frame covering the entire page. The new frame will be given the IDL ⟨*label*⟩, if provided.

If this frame is given a background colour, it should be created before any other static frames otherwise it will obscure all other static frames created before it.

### 5.4.4. Vertical and Horizontal Rules

You can create vertical or horizontal rules between two frames using the commands described below. The arguments ⟨*type-1*⟩ and ⟨*type-2*⟩ identify the frame type (`static`, `flow` or `dynamic`). For the unstarred version, the frames should be identified by their IDN and for the starred version they should be identified by their IDL.

`\insertvrule[`⟨*y-top*⟩`][`⟨*y-bottom*⟩`]{`⟨*type-1*⟩`}{`⟨*ID1*⟩`}{`⟨*type-2*⟩`}{`⟨*ID2*⟩`}` *modifier:* *

Creates a new static frame which fits between ⟨*type-1*⟩ frame identified by ⟨*ID1*⟩ and ⟨*type-2*⟩ frame identified by ⟨*ID2*⟩.

The new frame's y position and height are calculated from the lowest point of the lowest frame (of ⟨*ID1*⟩ and ⟨*ID2*⟩) to the highest point of the highest of the reference frames. The first optional argument ⟨*y-top*⟩ (default 0pt) extends the height of the new frame by that much above the highest point.

The second optional argument ⟨*y-bottom*⟩ (default 0pt) increases the height of the new frame by that much but also lowers the y position by that amount. If either of the optional arguments are negative, the frame will be shortened instead of extended.

The contents of this new frame are set to a vertical rule that extends the full height of the frame. The width of the rule is given by:

`\ffcolumnseprule` *initial:* `2pt`

This has changed as from version 1.09: versions prior to 1.09 used `\columnseprule` (provided by the LaTeX kernel).

The vertical rule drawn by `\insertvrule` is created using the command:

```
\ffvrule{⟨offset⟩}{⟨width⟩}{⟨height⟩}
```

This can be redefined if a more ornate rule is required (see Example 5).

The command:

```
\ffruledeclarations                                  initial: empty
```

can be redefined to set declarations that affect how the rule is drawn. The most likely use of this command is to set the rule colour. Any redefinition of `\ffruledeclarations` must come before the static frame is created. (Remember that the contents of a static frame are set in a box so the declaration is used at that point.)

```
\inserthrule[⟨x-left⟩][⟨x-right⟩]{⟨type-1⟩}{⟨ID1⟩}{⟨type-2⟩}
{⟨ID2⟩}                                                  modifier: *
```

This creates a new static frame which fits between ⟨*type-1*⟩ frame identified by ⟨*ID1*⟩ and ⟨*type-2*⟩ frame identified by ⟨*ID2*⟩.

The new frame's `x` position and width are calculated from the leftmost point of the furthest left frame (of ⟨*ID1*⟩ and ⟨*ID2*⟩) to the rightmost point of the furthest right of the reference frames. The first optional argument ⟨*x-top*⟩ (default 0pt) extends the width of the new frame by that much above the rightmost point.

The second optional argument ⟨*x-bottom*⟩ (default 0pt) increases the width of the new frame by that much but also lowers the `x` position by that amount. If either of the optional arguments are negative, the frame will be shortened instead of extended.

The contents of this new frame are set to a horizontal rule that extends the full width of the frame. The height of the rule is given by `\ffcolumnseprule`.

The horizontal rule drawn by `\inserthrule` is created using the command:

```
\ffhrule{⟨offset⟩}{⟨width⟩}{⟨height⟩}
```

This can be redefined if a more ornate rule is required (see Example 5). The horizontal rule is also governed by `\ffruledeclarations`. As with `\insertvrule`, any redefinition of these commands must be made before `\inserthrule` to have an effect.

Example 5 creates two flow frames with a static frame above (with `\twocolumnStop`) and the creates a vertical rule between the two flow frames and a horizontal rule between the top static frame and the two flow frames, bit first the rules are redefined to use a zigzag pattern (achieved with the tikz package). 5

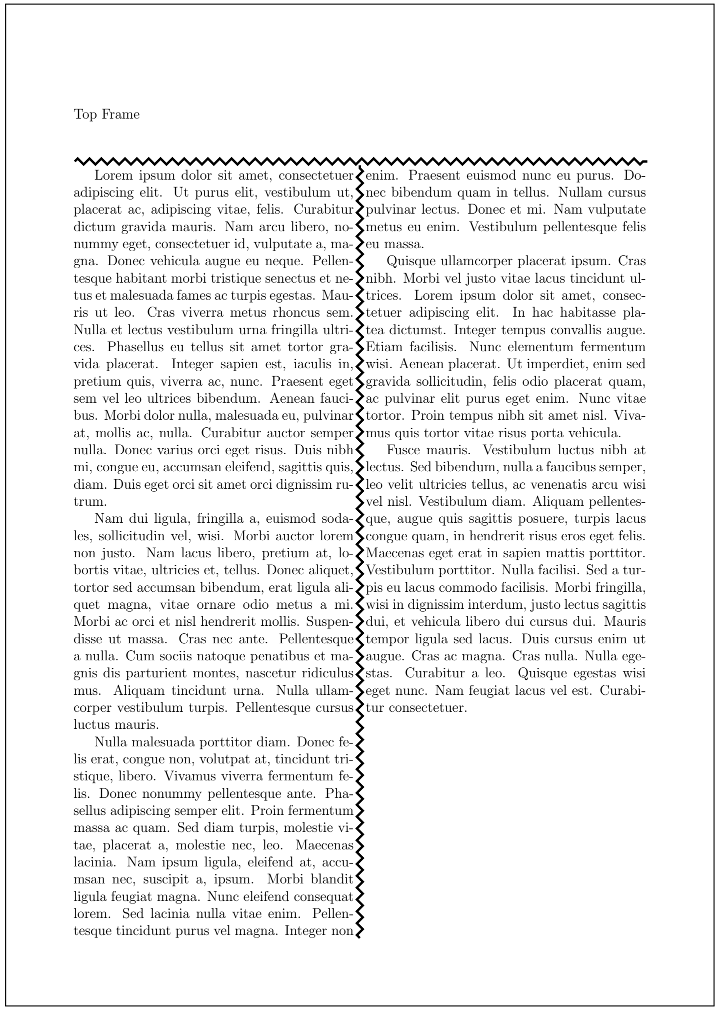
```
\usepackage{flowfram}
\usepackage{tikz}
\usetikzlibrary{decorations.pathmorphing}
\twocolumnStop{1in}
\renewcommand{\ffvrule}[3]{%
 \hfill \tikz{%
 \draw[decorate,decoration=zigzag,
   line width=#2,yshift=-#1]
  (0,0) -- (0pt,#3); }%
 \hfill \mbox{}}
\insertvrule{flow}{1}{flow}{2}

\renewcommand{\ffhrule}[3]{%
 \tikz{%
  \draw[decorate,decoration=zigzag,
   line width=#3,xshift=-#1] (0,0) -- (#2,0pt);
  }}
\inserthrule{static}{1}{flow}{1}
```

The lipsum package is used for filler text. The document body sets the content for the first static frame (the top one):

```
\setstaticcontents{1}{Top Frame}\lipsum[1-5]
```

Remember that because the static frame contents are set in a box, any change to `\ffvrule` or `\ffhrule` must occur before the corresponding `\insertvrule` or `\inserthrule` for the change to have an effect.

↑ Example 5: Rules



Top Frame

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

# 6. Thumbtabs and Minitocs

*This chapter describes how to create thumbtabs and minitocs, such as those used in this document.*

## 6.1. Thumbtabs

On the right hand side of the odd pages and on the left hand side of even pages in this document, there is a blue rectangle with the chapter number in it. This is a thumbtab, and it gives you a rough idea whereabouts in the document you are when you quickly flick through the pages (of a hard copy). Each thumbtab is in fact a dynamic frame.

There are two dynamic frames per thumbtab: the thumbtab index shown in the table of contents and the main thumbtab shown in the applicable part of the document. These frames are given special labels: thumbtabindex⟨*n*⟩ for the ⟨*n*⟩th thumbtab index and thumbtab⟨*n*⟩ for the ⟨*n*⟩th thumbtab.

With FlowframTk you can also define separate thumbtabs for even pages in two-sided documents. These have special labels eventhumbtabindex⟨*n*⟩ for the ⟨*n*⟩ thumbtab index frame shown on even pages and eventhumbtab⟨*n*⟩ for the ⟨*n*⟩ thumbtab frame shown on even pages.

Options that govern the thumbtab settings are listed in §1.2.1.

`\makethumbtabs[`⟨*y-offset*⟩`]{`⟨*height*⟩`}[`⟨*section-type*⟩`]`

preamble only

If you want thumbtabs in your document, you need to add `\makethumbtabs` in the preamble. This reads information in from the `ttb` file created during the previous LaTeX run. As with the table of contents, you need a rerun to ensure that the thumbtab information is correct.

Each dynamic frame corresponding to a thumbtab and thumbtab index is then defined by `\makethumbtabs` with the contents set according to the information read from the `ttb` file. If a dynamic frame with the corresponding special label has already been defined, that will be used instead of defining a new frame. (This will be the case with any package or class files that have been exported from FlowframTk where thumbtab frames have been setup.)

If you have used FlowframTk and you haven't defined enough thumbtab frames, `\makethumbtabs` will fall back on the default thumbtab frame style for the missing frames.

When `\makethumbtabs` creates a dynamic frame, the ⟨*height*⟩ argument is the height of each thumbtab frame. The default $y$ position puts the first thumbtab level with the top of the typeblock and each following thumbtab is shifted below the last by ⟨*height*⟩. The first optional argument ⟨*y-offset*⟩ (a dimension) will shift all the thumbtabs vertically by ⟨*y-offset*⟩. The width of each thumbtab is given by:

`\thumbtabwidth` *initial:* `1cm`

This value may be changed with `\setlength` but only changes that are made before `\makethumbtabs` will have any effect. The evenx position will automatically be calculated. If the frame was already defined, no change will be made to its attributes.

The dimension settings will be ignored if the thumbtab frames have already been defined.

The final optional argument ⟨*section-type*⟩ indicates what section unit should be used for the thumbtabs. If omitted, ⟨*section-type*⟩ obtained by expanding:

`\defaultthumbtabtype` *initial: varies*

This command should expand to the control sequence name (no backslash) of the applicable section type. The default definition is `chapter` if `\chapter` is defined, otherwise it's `section`.

Make sure that you have set all your preferred thumbtab settings before using `\makethumbtabs`. For example:

```
\flowframsetup{
 thumbtabs=number,
 thumbtab-text=normal
}
\makethumbtabs{0.75in}
```

Once the thumbtab dynamic frames have been defined and setup, the thumbtab indexing must be enabled.

```
\enablethumbtabs
```

This should be placed at the point where you want the thumbtab indexing to start.

There will be no thumbtabs if you don't have *both* `\makethumbtabs` and `\enablethumbtabs` in your document.

You may put `\enablethumbtabs` in the preamble (after `\makethumbtabs`) in which case any front matter sectional units may have associated thumbtabs, depending on the `matter-thumbtabs` option. Alternatively, place `\enablethumbtabs` just before the first sectional unit that should appear as a thumbtab. For example:

```
\tableofcontents
\mainmatter
\enablethumbtabs
\chapter{Introduction}
```

You can stop indexing thumbtabs with:

```
\disablethumbtabs
```

This should be placed before the sectional unit that shouldn't have a thumbtab. For example:

```
End of last chapter.
\appendix
\disablethumbtabs
\chapter{Supplementary Material}
```

The thumbtab index frames will automatically be shown in the table of contents unless you have used `adjust-toc=off`, which prevents flowfram from redefining `\tableofcontents`. If you have done this and want the thumbtab indexes to appear elsewhere, you can switch them on with:

```
\thumbtabindex[⟨page-list⟩]
```

The optional argument ⟨*page-list*⟩ indicates which pages the thumbtab frames should be shown on.

There are commands available to customized the format of the thumbtab content.

`\thumbtabhyperlinkformat{⟨anchor⟩}{⟨text⟩}{⟨height⟩}`

This command is used when the content should have a hyperlink to the start of the associated sectional unit. The hyperref package needs to be loaded to support this. If hyperref is loaded after flowfram, `\thumbtabhyperlinkformat` will automatically be redefined to use `\hyperlink`.

Regardless of whether or not hyperref has been loaded, `\thumbtabhyperlinkformat` will format the text with:

`\thumbtabformat{⟨text⟩}{⟨height⟩}`

The behaviour of this command varies according to the `thumbtab-text` option. You can redefine `\thumbtabformat` if you prefer, but the `thumbtab-text` options will no longer have an effect after that.

`\thumbtabindexformat{⟨anchor⟩}{⟨text⟩}{⟨height⟩}`

This command is used to format the content of the thumbtab index frames. The default definition just uses `\thumbtabhyperlinkformat`. The `\thumbtabindexformat` command will be redefined by the `thumbtab-links` option.

`\thumbtabregularformat{⟨anchor⟩}{⟨text⟩}{⟨height⟩}`

This command is used to format the content of the thumbtab (non-index) frames. The default definition just uses `\thumbtabformat`. The `\thumbtabregularformat` command will be redefined by the `thumbtab-links` option.

Since thumbtab frames are just dynamic frames, it is possible to change their attributes with `\setdynamicframe`. However, the dynamic frames won't be defined until

`\makethumbtabs` finds the relevant information in the `ttb` file. This means that you will get an error on the first run when the frames are not defined.

Instead, the thumbtab frame attributes can be changed with:

```
\setthumbtab{⟨index⟩|all}{⟨key=value list⟩}
```

This sets the attributes ⟨*key=value list*⟩ for the dynamic frames associated with the thumbtab with the given ⟨*index*⟩ (starting from 1 for the first thumbtab). Only a warning occurs if the dynamic frames haven't been defined.

For example:

```
\setthumbtab{1}{backcolor=[rgb]{0.15,0.15,1}}
```

This is essentially equivalent to:

```
\setdynamicframe*
 {thumbtabindex1,thumbtab1}
 {backcolor=[rgb]{0.15,0.15,1}}
```

Note that ⟨*index*⟩ isn't the same as the frame's IDN. The first argument may be the keyword `all`, which indicates all thumbtabs frames.

To just change the settings for the thumbtab index, use:

```
\setthumbtabindex{⟨index⟩|all}{⟨key=value list⟩}
```

This is like `\setthumbtab` but only changes the thumbtab index frames. For example:

```
\setthumbtabindex{1}{backcolor=[rgb]{0.15,0.15,1}
}
```

This is essentially equivalent to:

```
\setdynamicframe*
 {thumbtabindex1}
 {backcolor=[rgb]{0.15,0.15,1}}
```

By default, the thumbtabs have a grey background (unless you have created them with FlowframTk). In this document, I set the background colours with:

```
\setthumbtab{1}{backcolor=[rgb]{0.15,0.15,1}}
\setthumbtab{2}{backcolor=[rgb]{0.2,0.2,1}}
\setthumbtab{3}{backcolor=[rgb]{0.25,0.25,1}}
\setthumbtab{4}{backcolor=[rgb]{0.3,0.3,1}}
\setthumbtab{5}{backcolor=[rgb]{0.35,0.35,1}}
\setthumbtab{6}{backcolor=[rgb]{0.4,0.4,1}}
\setthumbtab{7}{backcolor=[rgb]{0.45,0.45,1}}
\setthumbtab{8}{backcolor=[rgb]{0.5,0.5,1}}
\setthumbtab{9}{backcolor=[rgb]{0.55,0.55,1}}
```

I then defined my own custom command for the frame style:

```
\newcommand{\thumbtabstyle}{%
 \centering\bfseries\large\sffamily
}
```

```
\setthumbtab{all}{valign=c,style=
thumbtabstyle,textcolor=white}
```

## 6.2. Minitocs

In this document, after each chapter heading, there is a mini table of contents (mini-toc) for that chapter. To enable mini-tocs, use the command:

`\enableminitoc[⟨section-type⟩]`

preamble only

This enables a mini-toc at the start of the given sectional unit. If present, the optional argument ⟨*section-type*⟩ should be the control sequence name (no backslash) of the section command. If omitted, the default is to use the same as that used by thumbtabs.

The mini-toc will normally appear immediately after the corresponding section heading. If you prefer the mini-toc to appear in a dynamic frame, you can use:

`\appenddfminitoc{⟨ID⟩}`

*modifier:* *
preamble only

This appends the mini-toc to the dynamic frame identified by ⟨*ID*⟩ (the IDN for the unstarred version or the IDL for the starred version). The mini-toc support needs to be enabled first with `\enableminitoc`.

For example, in this document, I have:

```
\appenddfminitoc*{chaphead}
```

Recall from §2.4.1 that `\ChapterInDynamic` causes the corresponding sectioning command to reset the contents of the identified dynamic frame. The `\appenddfminitoc` command will cause the corresponding sectioning command to append to the contents of the associated dynamic frame.

This means that if you have something like:

```
\ChapterInDynamic*{⟨label⟩}
\enableminitoc
\appenddfminitoc*{⟨label⟩}
```

for the same dynamic frame ⟨*label*⟩, then `\chapter` will first set the contents with the section heading and will then append the mini-toc. If `\ChapterInDynamic` isn't used or has a different frame, then the behaviour of `\appenddfminitoc` will depend on whether or not the clear attribute has been set on the mini-toc frame. If the clear attribute is not set then the mini-toc code will only be appended once. (Otherwise the frame would end up accumulating all the mini-tocs.)

The style of the mini-toc text is governed by:

```
\minitocstyle{⟨content⟩}
```

The default behaviour is to simply reset the font and text colour to normal. The argument is the mini-toc content. Note that this style isn't applied to the dynamic frame's style as the reason for appending the mini-toc content is because there may be other content in the frame, such as the chapter heading. The `\minitocstyle` command is placed inside a group so any declarations will be scoped.

The mini-toc has vertical spacing before and after it.

```
\beforeminitocskip                                    initial: 0pt
```

The gap before the mini-toc.

```
\afterminitocskip                              initial: \baselineskip
```

The gap after the mini-toc. These are both lengths and can be changed with `\setlength`.

# 7. Global Settings

*This section describes style macros, lengths and counters used by the* flowfram *package.*

## 7.1. Macros

The following macros can be changed using `\renewcommand`:

```
\setffdraftcolor
```

Sets the colour of the bounding boxes when displayed in draft mode (see §1.4). The default definition is:

```
\color[gray]{0.8}
```

For example, if you want a darker grey, do:

```
\renewcommand{\setffdraftcolor}{\color[gray]{0.3}
}
```

```
\setffdrafttypeblockcolor
```

Sets the colour of the typeblock outline when displayed in draft mode. The default definition is:

```
\color[gray]{0.9}
```

```
\fflabelfont
```

Sets the font for the annotation labels shown in draft mode. The default definition is:

```
\small\sffamily
```

## 7.2. Lengths

The following are lengths, which can be changed using `\setlength`:

```
\fflabelsep                                   initial: 1pt
```

The distance from the right hand side of the bounding box at which to place the annotation.

```
\flowframesep
```

The gap between the text of the frame and its border, for the standard border types. This is initialised to `\fboxsep` when flowfram loads. It won't be automatically updated if `\fboxsep` is later changed.

```
\flowframerule
```

The width of the frame's border, if using a border given by a frame making command that uses `\fboxsep` to set its border width (for example, `\fbox`).

```
\sdfparindent                                 initial: 0pt
```

The default paragraph indentation within static or dynamic frames. This may be overridden for individual frames with the parindent attribute.

`\vcolumnsep` *initial:* `\columnsep`

The approximate vertical distance between the top frame and the column frames when using `\Ncolumntop` etc. (The height of the flow frame may be adjusted if the ffvadjust setting is on.) This length should be changed before the frames are defined for the change to have effect.

`\columnsep`

The `\columnsep` length is provided by the LaTeX kernel and is used in LaTeX's normal two-column mode. With flowfram it's used for the horizontal distance between the column frames that are created with commands such as `\Ncolumn`. This length should be changed before the frames are defined for the change to have effect.

## 7.3. Counters

The following are counters that can be accessed via `\value{`⟨*counter−name*⟩`}` or `\the`⟨*counter−name*⟩.

The values of these counters may be referenced but should not be modified. They are automatically incremented by the commands.

№ **`maxstatic`**

The total number of static frames that have been defined so far. You can use this value to reference the most recently defined static frame.

№

**maxflow**

The total number of flow frames that have been defined so far. You can use this value to reference the most recently defined flow frame.

№

**maxdynamic**

The total number of dynamic frames that have been defined so far. You can use this value to reference the most recently defined dynamic frame.

№

**maxthumbtabs**

The total number of thumbtabs.

№

**absolutepage**

The absolute page number.

№

**thisframe**

Stores the IDN of the current flow frame. You can label and reference the IDN using:

```
\labelflowidn{⟨label⟩}
```

This is analogous to the standard `\label` command, but will always refer to the IDN of the current flow frame. It can then be referenced using `\ref`.

Avoid using \labelflowidn or thisframe in the contents of a static or dynamic frame.

**displayedframe**

This counter is reset at the start of each page and is incremented each time a flow frame is selected by the output routine. If all flow frames are displayed on the current page then this will have the same value as thisframe. However, if some flow frames are invalid for the current page, the values of displayedframe and thisframe may be different. You can label the displayedframe counter using:

```
\labelflow{⟨label⟩}
```

The label may then be referenced with \ref. For example, if your document has a column layout:

```
This text is about hippos\labelflow{hippos}.
…
% Somewhere else in the document:
See column~\ref{hippos} on page~\pageref{hippos}
for information on hippos.
```

Avoid using \labelflow or displayedframe in the contents of a static or dynamic frame.

# 8. FlowframTk

FlowframTk is a Java graphical application that may be used to create images from shapes that are defined by control points (vector graphics). Bitmaps and text can also be added. The latest stable version can be found on CTAN[1] but experimental releases are also available on GitHub.[2] The application can be invoked from the command line with:

```
flowframtk
```

Depending on how it has been installed, you may also be able to start it from your operating system's application menu or from a desktop shortcut.

## 8.1. Export to Image

An image created in FlowframTk can be exported as a LaTeX (`tex`) file that contains a pgfpicture environment (provided by the pgf package) with the basic layer commands that replicate (as near as possible) the image created in FlowframTk. By way of example, figure 1.1 was created using FlowframTk (with the isometric grid).

Any document that inputs one of these exported files will need to include the pgf package, but the exported file may also contain some bespoke commands needed to emulate certain effects, such as outlined text. These are provided by the flowframtkutils package (see §8.4) which automatically loads pgf.

If all you want is to include the exported image in your document, then you don't need to load the flowfram package in your document (but you will need to ensure that flowfram is installed in order to load the supplementary flowframtkutils package, if required). The flowfram package is too fragile and has the potential to conflict with too many other classes or packages to load it unnecessarily.

[1] `ctan.org/pkg/flowframtk`
[2] `https://github.com/nlct/flowframtk/releases`

## 8.2. Compute Parameters for `\parshape` or `\shapepar`

The shape attribute is only available for static or dynamic frames. It can be set automatically if you want to use FlowframTk's export to document class or package function (§8.3) but if you want a shaped paragraph in a flow frame or if you don't need to use the flowfram package but would like a shaped paragraph in an ordinary document, then you can create the desired shape in FlowframTk and use the "Parshape" or "Shapepar" function (be sure to set the normal font size first in FlowframTk's TeX configuration settings). This will calculate the parameters for the applicable command and create a file that contains the command and arguments needed to shape a paragraph.

Simply `\input` the file (that was created by FlowframTk's Parshape or Shapepar function) at the start of the paragraph that should be shaped. Take care not to insert a paragraph break between inputting the file and the start of the text. See the FlowframTk user manual for further details.

Since `\parindent` is a TeX primitive and shapepar provides generic code, this function may be used with other TeX formats, such as plain TeX.

## 8.3. Export to Document Class or Package

Whilst creating images is something that can also be done by other vector graphics applications (some of which may create more readable tikz code), FlowframTk can also create a LaTeX package (`sty`) or class file (`cls`) that uses the flowfram package. In this case, instead of reproducing the image, the shapes may be used to identify flow frames, static frames or dynamic frames. The definitions (`\newflowframe`, `\newstaticframe` and `\newdynamicframe`) will be written to the exported `sty` or `cls` file. You may find that using this graphical interface is an easier approach to defining frames than trying to calculate the locations and dimensions using LaTeX commands. When you use FlowframTk's export function to create a document class or package, the created file will contain the code to automatically load both flowfram and flowframtkutils (but make sure that you have at least version 0.8.8 of FlowframTk).

When using FlowframTk to construct flowfram data, you need to first establish the typeblock and then identify the shapes that should represent frames. Any shapes that aren't identified as such won't be exported. (This is useful if, for example, you have shapes that provide guides for the positioning of other shapes.) You can choose whether the shape itself should be used as a border (or background, if it's filled) for the corresponding frame or whether to simply use the shape's bounding box as the location and size of the frame.

> When using the FlowframTk export function, don't set the backcolor or bordercolor attributes. Instead, use the line paint and fill paint settings in FlowframTk.

You may also define dynamic frames with special labels but make sure you have an up-to-date version of both flowfram and FlowframTk to ensure they are properly implemented. With at least FlowframTk version 0.8.8, you can use the Flow Frame Wizard to ensure that the correct labels are used.

## 8.4. The flowframtkutils Package

```
\usepackage[⟨options⟩]{flowframtkutils}
```

The flowframtkutils package is provided with flowfram v2.0+ to assist with files that have been exported from FlowframTk. It provides the bespoke commands that may be written to those files. Ensure that you have at least version 0.8.8 of FlowframTk to support newer commands.

The flowframtkutils package doesn't load the flowfram package as it may be simply required when an image is exported as pgf code. However, flowframtkutils does automatically load the pgf package as it's likely to be needed in either the image export or the class/package export.

### 8.4.1. Options

All options that can be passed to flowfram can also be passed to a package (`sty`) or class (`cls`) created by FlowframTk. However, flowframtkutils itself only has the following options, which can only be set when the package is loaded:

**`textpath`**=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**

If true, when flowframtkutils is loaded, it will automatically load the decorations.text PGF library after loading the pgf package. Note that this library doesn't support all of FlowframTk's text-path settings. See the FlowframTk documentation for further details.

**`outline`**=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**

If true, when flowframtkutils is loaded, it will try to provide the appropriate support for outline text and will redefine `\jdroutline` as appropriate.

If pdfLaTeX or LuaLaTeX is used, this option will input the `pdf-trans.tex` file, which contains generic pdfTeX code to render text as an outline. This code includes pdfTeX primitives, so if LuaLaTeX is used, the luatex85 package will automatically be loaded with this option to ensure they are defined.

If a different engine is used, this option will load the pst−char package instead, which will use PSTricks to render the outline.

### 8.4.2. Provided Commands

Many of the flowfram-related commands provided by the flowframtkutils package aren't intended for general use, as they are provided to simplify the code written by FlowframTk to ensure that the bespoke files are correctly integrated with the flowfram package with the class/package export function.

Most of the user-level commands described here are those used when exporting an image to a file containing the pgfpicture environment. (The "jdr" prefix is a throwback to FlowframTk's original name, JpgfDraw.)

If you export to a complete document, FlowframTk version 0.8.8 and above will search for the flowframtkutils package using `kpsewhich`. If it's found, FlowframTk will assume it's available for use. If not, or if an older version of FlowframTk is used, it will provide the definition of the "jdr" commands if they are required.

```
\includeteximage[⟨key=value list⟩]{⟨filename⟩}
```

The `\includeteximage` command is provided with `\ProvideDocumentCommand` rather than defined as a new command, as it may be defined in a similar manner by other packages. This command isn't written to any exported file by FlowframTk but is provided by flowframtkutils to include an exported image (`tex`) file in a document. You can simply input the file with commands like `\input`, but `\includeteximage` provides some extra features.

The filename identified by ⟨*filename*⟩ is input but the command first locally sets the input search path to match the graphics search path used by `\includegraphics`. This means that you can save your exported image `tex` file to the same location as your other image files.

If the optional argument is present, a transformation can be applied to the image. For example, if it turns out to be too large for the page, you could go back to FlowframTk and shrink it or you can use the scaling options in `\includeteximage`. The recognised keys are just a subset of those recognised by graphicx: `scale`, `angle`, `width`, `height` and `alt`. Bear in mind that scaling the image will also scale any text within the image.

```
\jdroutline{⟨pdf-trans code⟩}{⟨pst-char code⟩}{⟨text⟩}
```

If enabled by the `outline` option, `\jdroutline` will attempt to render ⟨*text*⟩ as an outline using ⟨*pdf-trans code*⟩ (if `pdf-trans.tex` was loaded) or ⟨*pst-char code*⟩ (if pst-char was loaded). If not enabled, this command will generate a warning and just do ⟨*text*⟩.

For example, large bold “ABC” rendered with a green outline that’s filled with red:

```
\jdroutline
  {2 Tr 0 1 0 rg 1 0 0 RG}
  {fillstyle=solid,linecolor=green,fillcolor=red}
  {\bfseries\Huge ABC}
```

```
\jdrimagebox{⟨image⟩}
```

Used for encapsulated images, `\jdrimagebox` is designed to prevent problems if numerical rounding errors cause the image to be larger than the available area. It’s used when FlowframTk exports to a complete document that encapsulates the image. (Rather than exporting to a `tex` file that can be input from an existing document.)

The following commands are only written to exported files by FlowframTk v0.8.8+ if it’s detected that flowframtkutils is available.

```
\flowframtkimageinfo{⟨key=value list⟩}
```

If FlowframTk was instructed to include the title and creation date, this command will be written to the exported (`tex`) document file. This is only applicable when the export to a complete document is used, rather than export to a file just containing the image code. Available options:

```
title={⟨text⟩}
```

Set the title with `\flowframtkSetTitle{⟨text⟩}`. (The title is the image's description.)

```
creationdate={⟨PDF date⟩}
```

Set the creation date with `\flowframtkSetCreationDate{⟨PDF date⟩}`.

```
\flowframtkSetCreationDate{⟨PDF date⟩}
```

If hyperref has been loaded, this will use `\hypersetup` to set the PDF creation date, otherwise if `\pdfinfo` has been defined that will be used instead.

```
\flowframtkSetTitle{⟨title⟩}
```

If hyperref has been loaded, this will use `\hypersetup` to set the document title meta data, otherwise if `\pdfinfo` has been defined that will be used instead. Awkward characters within the title will be marked up with:

```
\flowframtkimgtitlechar{⟨char⟩}{⟨PDF replacement⟩}
```

This normally just expands to ⟨*char*⟩ but if `\pdfinfo` is required, then this will expand to the detokenized ⟨*PDF replacement*⟩. For example, if the image's description was set to "A House (with a pond)" in FlowframTk, and the image is exported as a complete document with meta data, then the document preamble would contain (for the given timestamp):

```
\flowframtkimageinfo{
 title={A House \flowframtkimgtitlechar{(}{\(}
with a pond\flowframtkimgtitlechar{)}{\)}},
 creationdate={D:20250815192214+01'00'}
}
```

If you provide a description for an object, it will be included as a comment. This can help to identify the relevant parts of the code but you may prefer to have hooks to allow you to add content from the document (rather than editing the exported image file). For example, to add tagging for accessibility or to uncover elements of the image in beamer. With FlowframTk v0.8.8+, you can specify what type of hook you want: paired, encapsulated or no hook.

```
\flowframtkstartobject{⟨n⟩}{⟨Java class
name⟩}{⟨description⟩}{⟨tag⟩}{⟨pgf point⟩}{⟨width⟩}{⟨height⟩}
```

If this hook is added to the image, it will always be paired with a matching closing hook:

```
\flowframtkendobject{⟨n⟩}{⟨Java class
name⟩}{⟨description⟩}{⟨tag⟩}{⟨pgf point⟩}{⟨width⟩}{⟨height⟩}
```

By default, the above pair do nothing.

Alternatively, you can choose to encapsulate each object, which will instead use the following hook:

```
\flowframtkencapobject{⟨n⟩}{⟨Java class name⟩}{⟨description⟩}{⟨tag⟩}{⟨pgf point⟩}{⟨width⟩}{⟨height⟩}{⟨object⟩}
```

By default, this simply does the final argument ⟨*object*⟩, which is the code to draw the object.

The common arguments in these hooks are:

⟨*n*⟩

A number that's unique for each object in a given image. This argument in `\flowframtkendobject` must always be the same as its matching `\flowframtkstartobject`. The numbering starts at 0 for the entire image and then increments for each object within the image, descending down groups.

⟨*Java class name*⟩

The Java class name of the object. For example, `JDRPath` for a path (which may consist of lines, curves and moves), `JDRText` for text or `JDRBitmap` for bitmaps.

This is the class responsible for writing the PGF code to render (as closely as possible) the object. That is, the final argument for `\flowframtkencapobject` or the content between `\flowframtkstartobject` and `\flowframtkendobject`. This is true, even if the object needs to be temporarily converted for the export. For example, a text area with gradient paint may be converted to a filled path if the "To Path" export setting has been applied. The PGF code will then be path construction code not `\pgftext`, but the class name will still be `JDRText`.

⟨*description*⟩

The description assigned to the object or empty if no description was set.

⟨*tag*⟩

The tag assigned to the object or empty if no tag was set.

⟨*pgf point*⟩

The lower left point of the object's bounding box, in the form `\pgfpoint{⟨x⟩}{⟨y⟩}`.

⟨*width*⟩

The width of the object's bounding box.

⟨*height*⟩

The height of the object's bounding box.

A simplistic way of adding overlays would be to export using the paired commands and add `\pause`:

```
\renewcommand{\flowframtkendobject}[7]{\pause}
```

The flowframtkutils package provides a function that may be used in the definition of `\flowframtkencapobject`. To use this function, first make sure that you export to an image with the "Encapsulated" setting on. Then anywhere before the image is included in the document, use:

```
\FlowFramTkUtilsSetOverlayEncap[⟨key=value list⟩]
```

This redefines `\flowframtkencapobject` to use the low-level function:

```
\flowframtkutils_uncover_encap:nnnnnnnn {⟨n⟩}
{⟨Java class name⟩} {⟨description⟩} {⟨tag⟩} {⟨pgf point⟩} {⟨width⟩}
{⟨height⟩} {⟨object⟩}
```

(There shouldn't be any need to use this function directly.)

This recognises the settings that may be set in the optional argument of `\FlowFramTkUtilsSetOverlayEncap` to govern the way that objects are uncovered.

The default is to use beamer's `\uncover` command but two helper functions are used (which both required LaTeX3 syntax to be enabled if you need to redefine them).

```
\flowframtkutils_uncover:nn {⟨number⟩} {⟨content⟩}
```

Uncovers for a single slide. The default definition is:

```
\uncover<⟨number⟩>{⟨content⟩}
```

```
\flowframtkutils_uncover_from:nn {⟨number⟩} {⟨content⟩}
```

Uncovers from a slide. The default definition is:

```
\uncover<⟨number⟩->{⟨content⟩}
```

If you later want to change the settings you can use:

```
\FlowFramTkUtilsOverlayEncapSetup{⟨key=value list⟩}
```

This should be placed before the image is input into the document but bear in mind that the settings are local and there may be interference if they are changed within the same frame that has the overlays.

The \flowframtkutils_uncover_encap:nnnnnnnn function is still a little experimental, but it basically works as follows:

- When ⟨*n*⟩ is 0 (that is, the entire image), the bounding box is saved and some other initialisation is performed.
- When ⟨*n*⟩ is greater than 0, the overlay setting is searched for:
  - With the auto-overlay=true option, an integer variable is incremented for all object types except groups. This value is used as the overlay. Each object will be uncovered according to the order in which it was written to the exported file.
  - With the auto-overlay=false option, the tag-overlay setting is checked. If that has an overlay identified for the object's tag then that will be used. If not, the object-overlay setting is checked. If that has an overlay identified for the object's type then that will be used. If not, the fallback-overlay setting is checked. If that has been set then that will be used. If not, the current object will be visible with no overlay specification.
- When ⟨*n*⟩ is greater than 0 and the annote setting is not false, then after the object has been draw according to the above overlay settings an annotation will be added if applicable. The annotation consists of text with optionally a line from the object's bounding box to the text, which is offset according to the dx and dy settings. By default the line has an arrow head at the start. This may be changed with the annote-arrow setting. Whether or not the object has an annotation is determined by annote-text. The annotation overlay is determined by the annote setting.

For example, assuming that the file image.tex was created by FlowframTk with the "Object Markup" option set to "Encapsulated":

```
\documentclass{beamer}
\usepackage{flowframtkutils}
\FlowFramTkUtilsSetOverlayEncap
 [
   object-overlay={path = 1, text = 2},
   annote,
   annote-text=description-or-tag
 ]
\begin{document}
\begin{frame}
\frametitle{An Image}
\centering
\input{image}
\end{frame}
\end{document}
```

There are two ways of adding annotations. You can either add the text and, if applicable, a line with an arrow head to the image in FlowframTk and tag them. Then you can use the `tag-overlay` option to uncover them. Alternatively, you can use the `annote` option to automatically add annotations based on each object's tag or description. This second method is less precise as the annotation is placed relative to the object's bounding box, which may be at a distance from a non-rectangular shape, but may be sufficient for simple images.

Available options are listed below.

**`uncover`**=⟨*value*⟩ *initial:* **`from`**

Determines whether or not overlays should be used for image objects and, if so, should the object just be shown on a single overlay or from a certain number.

`uncover=single`

Uncover for a single slide.

`uncover=from`

Uncover from a slide.

`uncover=false`

Don't have overlays for image objects (but annotations may still be overlaid).

`annote=`⟨*value*⟩ *default:* `true`; *initial:* `false`

Determines whether or not annotations should be added and, if so, what overlays should be used for annotations. If this setting is on but not annotation text is available for a given object then the annotation will be skipped. The annotation text, if available, is offset from the object's bounding box. Optionally a line may be drawn from the bounding box to the text. The "annotation" refers to the text and, if present, the line.

`annote=single`

Uncover annotation, if applicable, for a single slide. That is, use `\uncover<`⟨*num*⟩`>{`⟨*annotation*⟩`}`.

`annote=from`

Uncover annotation, if applicable, from a slide. That is, use `\uncover<`⟨*num*⟩`->{`⟨*annotation*⟩`}`.

**annote=match**

Match the `uncover` setting.

**annote=true**

Enable annotations without changing the overlay setting.

**annote=false**

Don't show annotations.

**annote-position=⟨*value*⟩** *initial:* **auto**

If annotations should be added, this setting indicates in which direction the annotation should be.

**annote-position=auto**

Automatically determine the direction based on the object's relative position within the image.

**annote-position=north**

Place the annotation above the object.

**annote-position=northeast**

Place the annotation above right of the object.

**annote-position=east**

Place the annotation right of the object.

**annote-position=southeast**

Place the annotation below right of the object.

**annote-position=south**

Place the annotation below the object.

**annote-position=southwest**

Place the annotation below left of the object.

**annote-position=west**

Place the annotation left of the object.

**annote-position=northwest**

Place the annotation above left of the object.

**annote-arrow**={⟨*key=value list*⟩} *default:* **show**

If annotations should be added, this setting governs the annotation arrow. Available settings are described in §8.4.2.2.

**annote-text**={⟨*value*⟩} *default:* **description**

The text used for the annotation. If the text is empty or no mapping has been provided then there won't be an annotation for the object.

**annote-text**=**hide**

No annotation text (which means no arrow either, regardless of the `annote-arrow` setting).

**annote-text**=**description**

If the object has a description, use that as the annotation text.

**annote-text**=**tag**

If the object has a tag, use that as the annotation text.

**annote-text**=**description-or-tag**

If the object has a description, use that as the annotation text, otherwise if the object has a tag use that.

**annote-text**=**tag-map**

If the object has a tag and a mapping has been set with `tag-text`, use the mapping as the annotation text.

**tag-text**={⟨*key=value list*⟩}

Sets up the mappings for use with `annote-text=tag-map`, where the key in ⟨*key=value list*⟩ is the tag and the value is the text to use for that tag. For example, if the image has some objects that have the tag "window" and some have the tag "door":

```
tag-text={
  window = {A Window},
  door = {A Door}
}
```

**auto-overlay**={⟨*boolean*⟩} *default:* **true**; *initial:* **false**

If true, auto-increment the overlay for each object that's not a group. (Groups are excluded as their content is nested.)

**tag-overlay**={⟨*key=value list*⟩}

If `auto-overlay=false`, set up the tag overlays, where the key in ⟨*key=value list*⟩ is the tag and the value is the overlay number. For example, if the image has some objects that have the tag "building", some have the tag "window" and some have the tag "door":

```
tag-overlay={
  building = 1,
  window = 2,
  door = 3
}
```

Within ⟨*key=value list*⟩, the value part should be a number but may also be empty (the equal sign is still needed). This means that there shouldn't be an overlay for an object with that tag. This is different from omitting the tag from the list as it prevents the `object-overlay` and `fallback-overlay` checks.

If the ⟨*key=value list*⟩ is not empty, this option automatically sets `auto-overlay=false`.

**`object-overlay`**`={`⟨*key=value list*⟩`}`

If `auto-overlay=false`, set up the object overlays, where the key in ⟨*key=value list*⟩ identifies the object and the value is the overlay number.

If you set an overlay for groups but objects within the group also have an overlay, this will cause nested `\uncover`.

The key may be any of: `group`, `path`, `text`, `text-path`, `symmetric-path`, `scaled-pattern`, `spiral-pattern`, `rotational-pattern` or `bitmap`. For example:

```
object-overlay={
  bitmap = 1,
  path = 2,
  text = 3
}
```

Within ⟨*key=value list*⟩, the value part should be a number but may also be empty (the equal sign is still needed). This means that there shouldn't be an overlay for an object of that type. This is different from omitting the object type from the list as it prevents the `fallback-overlay` check.

If the ⟨*key=value list*⟩ is not empty, this option automatically sets `auto-overlay=false`.

**`fallback-overlay`**={⟨*key=value list*⟩}

If `auto-overlay=false`, and if an object doesn't have an overlay identified by either `tag-overlay` or `object-overlay`, then use this fallback. Available keys for ⟨*key=value list*⟩ are listed in §8.4.2.1.

#### 8.4.2.1. Fallback Overlay Options

These are available settings for use in the `fallback-overlay` option value. They can also be set with `\keys_set:nn` where the module is `flowframtkutils / overlay / fallback`.

**`enable`**=⟨*boolean*⟩ *default:* **`true`**; *initial:* **`false`**

If true, enables fallback overlay setting. This means that if an overlay is identified by either `tag-overlay` or `object-overlay` then the overlay will be set to the default value (for non-group objects).

**`fixed`**={⟨*number*⟩}

If fallback overlay setting enabled, set the default overlay to ⟨*number*⟩.

**`increment`**

If fallback overlay setting enabled, set the default overlay to a number that's incremented for each use of the fallback.

### 8.4.2.2. Annotation Arrow Options

These are available settings for use in the `annote-arrow` option value. They can also be set with `\keys_set:nn` where the module is `flowframtkutils / overlay / arrow`.

**show**=⟨*boolean*⟩ *default:* **true**; *initial:* **true**

If annotations should be added, show an arrow in addition to the text. For example:

```
annote-arrow={show=true}
```

If this is the only `annote-arrow` setting, then it can be written more compactly:

```
annote-arrow=show
```

**hide**=⟨*boolean*⟩ *default:* **true**; *initial:* **false**

The inverse of `show`. If annotations should be added, don't show an arrow (only show the text).

**dx**={⟨*dim*⟩} *default:* **1cm**

If annotations should be added, the horizontal displacement from the start of the arrow. This is also the horizontal displacement for the annotation text, regardless of whether or not the arrow is shown.

**dy**={⟨*dim*⟩} *default:* **1cm**

If annotations should be added, the vertical displacement from the start of the arrow. This is also the vertical displacement for the annotation text, regardless of whether or not the arrow is shown.

**color**={⟨*colour-name*⟩} *initial:* **black**

The colour to use for the annotation arrow, if shown. The value may be empty to indicate the current stroke colour should be used, otherwise it should be a colour name suitable for use in `\pgfsetstrokecolor`.

**start**={⟨*arrow-name*⟩} *initial:* **<**

The start arrow marker to use for the annotation arrow, if shown. The value may be empty to indicate there should be no start arrow, otherwise it should be suitable for use in `\pgfsetarrowsstart`. The line starts at the object's bounding box so that's where the start arrow will be placed. A non-rectangular shape may be some distance away from this point.

Remember to include the arrows.meta PGF library if you want to use any of the meta arrows. For example with:

```
annote-arrow={
 start={Latex[length=5pt]},
}
```

the preamble requires:

```
\usepgflibrary{arrows.meta}
```

**end**={⟨*arrow-name*⟩} *initial:* ***empty***

The end arrow marker to use for the annotation arrow, if shown. The value may be empty to indicate there should be no end arrow, otherwise it should be suitable for use in `\pgfsetarrowsend`. The line ends on the annotation text's boundary, the location determined according to the `\pgftext` anchor.

For example:

```
annote-arrow={
 show,
 dx=2cm,
 dy=1cm,
 start={},
 end={>}
}
```

# 9. Troubleshooting

*This chapter should be consulted if you experience any problems using the flowfram package.*

For an up-to-date list of frequently asked questions, see the flowfram FAQ.[1]

## 9.1. General Queries

1. If all my flow frames are only defined on, say, pages `1-10`, what happens if I then add some extra text so that the document exceeds 10 pages?

   The output routine will create a new flow frame the size of the typeblock and use that.

2. Can I use the formatted page number in page lists?

   No.

3. Why not?

   When the output routine finishes with one flow frame it looks for the next flow frame defined on that page. If there are none left, it then searches through the page list of all the defined flow frames to see if the next page lies in that range. If there are none defined on that page, it ships out that page, and tries the next page. This gives rise to two problems:

   a) LaTeX is not clairvoyant. If it is currently on page 14, and on the next page the page numbering changes to A, it has no way of knowing this until it has reached that point, which it hasn't yet. So it is looking for a flow frame defined on page 15, not on page A.

   b) How does LaTeX tell if page C lies between pages A and D? It would require an algorithm that can convert from a formatted number back to an integer. Given that there are many different ways of formatting the value of a counter (besides the standard Roman and alphabetical formats) it would be impossible to write an algorithm to do this for some arbitrary format.

[1] `dickimaw-books.com/faq.php?category=flowfram`

4. Can I have an arbitrarily shaped frame?

   You can assign certain irregular shapes to static or dynamic frames, using the shape key (see §3.2). Note that the bounding box will still appear as a rectangle with the dimensions of the given frame which may not correspond to the assigned shape. This function is not available for flow frames.

5. Why has the text from my flow frame appeared in a static or dynamic frame?

   Assuming you haven't inadvertently set that text as the contents of the static or dynamic frame, the frames are most likely overlapping (see §1.1). In an attempt to clarify what's going on, suppose you have defined a static frame, a dynamic frame and two flow frames. The following is an approximate analogy (not strictly accurate as TeX may make several attempts to fill in the flow frames due to penalties etc):

   TeX has a sheet of paper on the table, and has pencilled in a rectangle denoting the typeblock. The paper is put to one side for now. TeX also has four rectangular sheets of transparent paper. The first (which I shall call sheet 1) represents the static frame, the next two (which I shall call sheets 2 and 3) represent the flow frames, and the last one (which I shall call sheet 4) represents the dynamic frame.

   TeX starts work on filling sheet 2 with the document text. Once it has put as much text on that sheet as it considers possible (according to its views on aesthetics), it puts sheet 2 into the "in tray", and then continues on sheet 3. While it's filling in sheets 2 and 3, if it encounters a command or environment that tells it what to put in the static frame, it fills in sheet 1 and then puts sheet 1 into the "in tray" and resumes where it left off on sheet 2 or 3. Similarly, if it encounters a command that tells it what to put in the dynamic frame, it stops what it's doing, fills in sheet 4, then puts sheet 4 into the "in tray", and resumes where it left off.

   Only when it has finished sheet 3 (the last flow frame defined on that page), will it gather together all the transparent sheets, and fix them onto the page starting with sheet 1 through to sheet 4, measuring the bottom left hand corner of each transparent sheet relative to the bottom left hand corner of the typeblock. TeX will then put that page aside, and start work on the next page. If two or more of the transparent sheets overlap, you will see through the top one into the one below (unless of course the top one has been painted over, either by setting a background colour, or by adding an image that has a non-transparent background).

Note that it's also possible that the overlap is caused by an overfull hbox that's causing the text to poke out the side of the flow frame into a neighbouring frame. Check the log file for warnings or use the `draft` option to the document class which will place a filled rectangle on the end of overfull lines.

6. Why do I get lots of overfull hbox messages?

   Probably because you have narrow frames. It's better to use ragged right formatting when the frames are narrow.

7. Why do I keep getting multiply-defined warnings?

   Probably because you have used `\label` in a static or dynamic frame that is displayed on more than one page. Try using the clear key to ensure that the frame is always cleared at the end of each page.

8. What happens if I use a command or environment that switches to two-column mode (e.g. theindex)?

   The behaviour depends on the `column-changes` setting (see §1.2.3).

9. How do I change the vertical alignment of material inside a static or dynamic frame?

   Use the valign attribute.

10. How do I compute the distance from the edge of the page instead of the typeblock?

    See §4.1.

11. Is there a GUI I can use to make it easier to create the frames?

    Yes, FlowframTk (see §7.3).

## 9.2. Unexpected Output

1. The lines of text at the beginning of my flow frames are the wrong width.

   This is a problem that will occur if you have flow frames with different widths, as the change in `\hsize` does not come into effect until a paragraph break. So if you have a paragraph that spans two flow frames, the end of the paragraph at the beginning of the second flow frame will retain the width it had at the start of the paragraph at the bottom of the previous flow frame. You can fix the problem by inserting `\framebreak` at the point where the frame break occurs (see §2.1.1).

2. My frames shift to the right when I add a border.

   This may occur if you use a border that is not recognised by the flowfram package. You will need to set the offset using the offset key (see §2.4.4).

3. I have a vertical white strip along the right hand side of every page.

   This can happen if you have, say, an A4 document, and `ghostscript` has letter as the default paper size. You can change the default paper size by editing the file `gs_init.ps`. Change:

   ```
   % Optionally choose a default paper size other than U.S. letter.
   % (a4)
   ```

   to:

   ```
   % Optionally choose a default paper size other than U.S. letter.
   (a4)
   ```

4. I don't have any output.

   All your flow frames are empty. TEX doesn't put the frames onto the page until it has finished putting text into the flow frames. So if there is no text to go in the flow frames it won't output the page. If you only want the static or dynamic frames filled in, and nothing outside of them, just do `\mbox{}\clearpage`. This will put an invisible something with zero area into your flow frame, but it's enough to convince TEX that the document contains some text.

5. The last page hasn't appeared.

   See the previous answer.

6. There is no paragraph indentation inside my static or dynamic frames.

   The paragraph indentation in static or dynamic frames is governed by the length `\sdfparindent` which is set to 0pt by default. To make the indentation the same as that used by flow frames place the following in the preamble:

   ```
   \setlength{\sdfparindent}{\parindent}
   ```

   Alternatively, you can use the parindent attribute if different indentation is required for different frames.

7. My section numbering is in the wrong order.

   Remember that the contents of the dynamic frames are not set until the page is shipped out, and the contents will be set in the order of IDN, so if you have any sectioning commands occurring within dynamic frames, they may not be set in the same order as they are in your input file.

8. The contents of my static or dynamic frame have shifted to the left when I used `\parshape`.

   This will happen if your `\parshape` specification exceeds the line width. For example:

```
\parshape=1 0.4\linewidth 0.7\linewidth
```

This specifies a line with overall length `1.1\linewidth` which is too long.

## 9.3. Error Messages

1. Illegal unit of measure (pt inserted)

   All lengths must have units. Remember to include the units when defining new frames. The following frame options require lengths: width, height, x, y, oddx, oddy, evenx, eveny, parindent and offset (although offset can also have the value `compute`).

2. Missing number, treated as zero

   LaTeX is expecting a number. There are a number of possible causes:

   a) You have used an IDL instead of an IDN. If you want to refer to a frame by its label, you need to remember to use the starred versions of the `\set`⟨*type*⟩`frame` commands, or when setting the contents of static or dynamic frames.

   b) When specifying page lists, you have mixed keywords with page ranges. For example: `1,even` is invalid.

3. Flow frame IDL '⟨*label*⟩' already defined

   All IDLs within each frame type must be unique. There are similar error messages for duplicate IDLs for static frames and dynamic frames.

4. Can't find flow frame id

   You have specified a non-existent flow frame IDL. There are similar error messages for static frames and dynamic frames. Check to make sure you have spelt the label correctly, and check you are using the correct frame type command. (For example, if a static frame has the IDL "`mylabel`", and you attempt to do `\setflowframe*{mylabel}{`⟨*options*⟩`}`, then you will get this error, because "`mylabel`" refers to a static frames not a flow frame.)

5. Key 'clear' is boolean

   The clear key can only have the values `true` or `false`.

6. Key 'clear' not available

   The clear key is only available for static or dynamic frames.

7. Key 'style' not available

   The style key is only available for dynamic frames.

8. Key 'margin' not available

   The margin key is only available for flow frames.

9. Key 'shape' not available

   The shape key is only available for static or dynamic frames.

10. Dynamic frame style '⟨*style*⟩' not defined

    The specified style ⟨*style*⟩ must be the name of a command without the preceding backslash. It is possible that you have mis-spelt the name, or you have forgotten to define the command.

11. Argument of `\fbox` has an extra `}`

    This error will occur if you do, say, border=`\fbox` instead of border=`fbox`. Remember not to include the initial backslash.

12. Not in outer par mode

    You can not have floats (such as figures, tables or marginal notes) in static or dynamic frames. If you want a figure or table within a static or dynamic frame use staticfigure or statictable.

13. Something's wrong—maybe missing `\item`

    Assuming that all your list type of environments start with `\item`, this may be caused by something going wrong with the `toc` (table of contents), `ttb` (thumbtab) or `aux` (auxiliary) files in the previous run. Try deleting them, and try again.

14. `\verb` illegal in command argument

    As a general rule, you can't use verbatim in a command argument. This rule applies to all the commands defined by the flowfram package. See also below.

15. I get "`\verb` illegal in command argument" when using a command inside the dynamiccontents environment.

    You can not use commands or environments inside either the dynamiccontents or dynamiccontents* environment.

# A. Glossary

## A.1. Terms

### Bounding box

The bounding box of a frame is the area allocated for the contents of that frame. However the text may not completely fill that area, and it is possible that the text may overflow that area.

### Dynamic frame

Frames in which text is fixed in place, but the contents are re-typeset each time the frame is displayed.

### Flow frame

The frames in a document such that the contents of the document environment flow from one frame to the next in the order that they were defined. There must be at least one flow frame on every page.

### FlowframTk

A Java application with a graphical user interface which can be used to construct frames for use with the flowfram package. It can also be used to create vector graphics images which can be exported as `tex` files containing a pgfpicture environment (defined by the pgf package). The supplementary package flowframtkutils provided with flowfram will need to be loaded by any document that inputs these files. The flowframtkutils package will also be automatically added to any `cls` or `sty` files created by `flowframtk` (version 0.8.8 onwards). The latest stable release is available on CTAN[1] or experimental releases can be found on GitHub.[2] FlowFramTk was originally called JpgfDraw (a Java pgf-code generating drawing application). Ensure you have at least version 0.8.8 to use the features mentioned in this manual.

[1] `ctan.org/pkg/flowframtk`
[2] `https://github.com/nlct/flowframtk/releases`

### Footer frame

A special dynamic frame with the label footer created either with `\makedfheader-footer` or via FlowframTk's export function, or the special dynamic frame with the label evenfooter that may be created with FlowframTk. If evenfooter doesn't exist, the footer frame will be used for the page footer on both odd and even pages. If evenfooter does exist, it will be used as the page footer for even pages if the document two-sided setting is on. This special behaviour must be enabled either with `\makedfheaderfooter` or via FlowframTk.

### Frame

A rectangular area of the page in which text can be placed (not to be confused with a frame making command). There are three types: flow, static and dynamic.

### Frame making command

A LaTeX command which places some kind of border around its argument. For example: `\fbox`.

### Header frame

A special dynamic frame with the label header created either with `\makedfheader-footer` or via FlowframTk's export function, or the special dynamic frame with the label evenheader that may be created with FlowframTk. If evenheader doesn't exist, the header frame will be used for the page header on both odd and even pages. If evenheader does exist, it will be used as the page header for even pages if the document two-sided setting is on. This special behaviour must be enabled either with `\makedfheaderfooter` or via FlowframTk.

### Identification label (IDL)

A unique label which can be assigned to a frame, enabling you to refer to the frame by label instead of by its IDN.

### Identification number (IDN)

A unique number assigned to each frame, which you can use to identify the frame when modifying its appearance. Example: if you have defined 3 flow frames, 2 static frames and

1 dynamic frame, the flow frames will have IDNs 1, 2 and 3, the static frames will have IDNs 1 and 2, and the dynamic frame will have IDN 1.

### Page list

A list of pages. This can either be a single keyword: `all`, `odd`, `even` or `none`, or it can be a comma-separated list of individual page numbers or page ranges. For example: `<3,5,7-11,>15` indicates pages 1,2,5,7,8,9,10,11 and all pages after page 15. These numbers refer to the decimal value of the page counter by default. To make them refer to the absolute page number use the package option `pages=absolute`.

### Page range

Page ranges can be closed (for example `5-10`) or open (for example, `<7` or `>9`).

### Static frame

Frames in which text is fixed in place. The contents are fixed until explicitly changed or cleared via the clear key in `\setstaticcontents`.

### Typeblock

The area of the page where the main body of the text goes. The width and height of this area are given by `\typeblockwidth` and `\typeblockheight`. This lengths are initialised to `\textwidth` and `\textheight`, and may be adjusted at the start of the document environment.

## A.2. Symbols

| Symbol | Description |
|---|---|
| № | A counter is being described. |
| 📌 | The syntax and usage of a command, environment or option etc. |
| 🗑 | A command, environment or option that is now deprecated. |
| i | An important message. |
| ⓘ | Prominent information. |
| 📄 | LaTeX code to insert into your document. |
| 🏷 | The definition of an option value. |
| 📄 | How the example code should appear in the PDF. |
| ☰ | An option that takes a value. |
| >_ | A command-line application invocation that needs to be entered into a terminal or command prompt. |
| ○ | A boolean option that is initially false. |
| ● | A boolean option that is initially true. |
| ≡ | An option that doesn't take a value. |
| ⚠ | A warning. |

# B. Alphabetical Summaries

## B.1. Command Summary

### A

**`\adjustcolsep`** flowfram

Adjust the value of `\columnsep` so that it's equal to twice its original value plus `\marginparwidth`. **§5**; 111

**`\afterminitocskip`** *initial:* `\baselineskip` flowfram

The vertical space after mini-tocs. **§6.2**; 146

**`\appenddfminitoc`**`{`⟨*ID*⟩`}` *modifier:* `*` flowfram
(preamble only)

Appends the mini-toc to the dynamic frame identified by ⟨*ID*⟩ (the IDN for the unstarred version or the IDL for the starred version). The mini-toc support needs to be enabled with `\enableminitoc`. **§6.2**; 144

**`\appenddynamiccontents`**`[options]{`⟨*ID*⟩`}{`⟨*content*⟩`}` *modifier:* `*` flowfram

Appends the given content to the dynamic frames identified by ⟨*ID*⟩, which is the frame's IDN (unstarred) or IDL (starred). If ⟨*options*⟩ is present, they will be applied to the frame at the same time. **§2.4**; 53

## B

**`\beforeminitocskip`** *initial:* `0pt` flowfram

The vertical space before mini-tocs. **§6.2**; 145

## C

**`\chapter`**[⟨*toc*⟩][⟨*thumbtab-title*⟩]{⟨*title*⟩} *modifier:* `*`

Modified by flowfram, this has an additional optional argument for the thumbtab titles (see `sections-extra-option` if you are using a class that also has a second optional argument). Note that the starred version also has both optional arguments.

**`\chapterfirstpagestyle`** *initial:* `plain` flowfram

The page style for the first page of a chapter. **§1.6**; 33

**`\ChapterInDynamic`**{⟨*ID*⟩} *modifier:* `*` flowfram v2.0+

Adapts `\chapter` to ensure that chapter titles are placed in the identified dynamic frame. **§2.4.1**; 54

**`\cleartoevenpage`** flowfram v2.0+

Similar to `\cleardoublepage` (which clears to the next odd page) this clears to the next even page for two-sided documents. For one-sided documents this just does `\clearpage`. **§2.1.1**; 39

**`\computebottomedge`**{⟨*dim var*⟩} flowfram

Computes the position of the bottom edge of the page, relative to the bottom of the typeblock. **§4.1**; 102

**`\computeflowframearea`**{⟨*ID list*⟩} *modifier:* * flowfram

Computes the minimum area surrounding the listed flow frames (identified by their IDL for the starred version and their IDN for the unstarred version) and stores the results in `\ffareax`, `\ffareay`, `\ffareawidth` and `\ffareaheight`.

**`\computeleftedgeeven`**{⟨*dim var*⟩} flowfram

Computes the position of the leftmost edge of the page, relative to the left side of the typeblock for even pages. **§4.1**; 101

**`\computeleftedgeodd`**{⟨*dim var*⟩} flowfram

Computes the position of the leftmost edge of the page, relative to the left side of the typeblock for odd pages. **§4.1**; 101

**`\computerightedgeeven`**{⟨*dim var*⟩} flowfram

Computes the position of the rightmost edge of the page, relative to the left side of the typeblock for even pages. **§4.1**; 102

**`\computerightedgeodd`**{⟨*dim var*⟩} flowfram

Computes the position of the rightmost edge of the page, relative to the left side of the typeblock for odd pages. **§4.1**; 102

**`\computetopedge`**{⟨*dim var*⟩} flowfram

Computes the position of the top edge of the page, relative to the bottom of the typeblock. **§4.1**; 101

**`\continueonframe`**[⟨*text*⟩]{⟨*ID*⟩} flowfram

Ends the current staticcontents or dynamiccontents and starts a new environment of the same type for the frame given by ⟨*ID*⟩. If the starred version of the environment was used then ⟨*ID*⟩ refers to the next frame's IDL otherwise it refers to its IDN. If ⟨*text*⟩ is omitted then `\ffdefaultstaticcontinuetext` or `\ffdefault-staticcontinuetext` will be used (with the appropriate arguments), as applicable. **§2.3.1**; 43

## D

**`\defaultthumbtabtype`** *initial: varies* flowfram

Expands to the default section type to be used by `\makethumbtabs` if the ⟨*section-type*⟩ argument is missing. The default definition is either `chapter` or `section`, depending on whether or not `\chapter` is defined. **§6.1**; 138

🗑 **`\dfchaphead`**{⟨*ID*⟩} *modifier:* * flowfram

Old method for putting chapter title in a dynamic frame.

**`\dfchapterclearpage`** flowfram v2.0+

Used with `\ChapterInDynamic` to start a new page. **§2.4.1**; 54

**`\dfEvenFooterStyle`**{⟨*text*⟩} flowfram v2.0

Used to format the page footer in the header and footer frame page styles for even pages. **§2.4.2**; 61

**`\dfEvenHeaderStyle`**{⟨*text*⟩} flowfram v2.0

Used to format the page header in the header and footer frame page styles for even pages. **§2.4.2**; 61

**`\dfOddFooterStyle`**{⟨*text*⟩} flowfram v2.0

Used to format the page footer in the header and footer frame page styles for odd pages. **§2.4.2**; 61

**`\dfOddHeaderStyle`**{⟨*text*⟩} flowfram v2.0

Used to format the page header in the header and footer frame page styles for odd pages. **§2.4.2**; 61

**`\dfswapoddeven`**{⟨*ID-list*⟩} *modifier:* * flowfram

Swap the odd and even co-ordinates for all listed dynamic frames. **§3.1.3**; 85

**`\disablethumbtabs`** flowfram

The thumbtab frames created by `\makethumbtabs` will have the hide attribute switched on and thumbtab information will stop being written to the `ttb`. **§6.1**; 140

**`\dynamicaddexclusion`**{⟨*IDN*⟩}{⟨*list*⟩} flowfram v1.14+

Appends ⟨*list*⟩ to the exclusion list for the dynamic frame identified by its IDN. **§3.1.1**; 74

```
\dynamicframeevenx{⟨IDN⟩}                                flowfram
```

Expands to the even-page $x$-coordinate of the dynamic frame identified by its IDN. §3.1.3; 82

```
\dynamicframeeveny{⟨IDN⟩}                                flowfram
```

Expands to the even-page $y$-coordinate of the dynamic frame identified by its IDN. §3.1.3; 83

```
\dynamicframeheight{⟨IDN⟩}                               flowfram
```

Expands to the height of the dynamic frame identified by its IDN. §3.1.3; 81

```
\dynamicframewidth{⟨IDN⟩}                                flowfram
```

Expands to the width of the dynamic frame identified by its IDN. §3.1.3; 80

```
\dynamicframex{⟨IDN⟩}                                    flowfram
```

Expands to the $x$-coordinate of the dynamic frame identified by its IDN. §3.1.3; 83

```
\dynamicframey{⟨IDN⟩}                                    flowfram
```

Expands to the $y$-coordinate of the dynamic frame identified by its IDN. §3.1.3; 84

**`\dynamicsetexclusion`**{⟨*IDN*⟩}{⟨*list*⟩} flowfram v1.14+

Sets the exclusion list for the dynamic frame identified by its IDN. **§3.1.1**; 74

**`\dynamicsetpagelist`**{⟨*IDN*⟩}{⟨*page-list*⟩ | ⟨*keyword*⟩} flowfram v1.14+

Sets the page list for the dynamic frame identified by its IDN. **§3.1.1**; 72

**`\dynamicswitchoffnext`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch off the listed dynamic frames from the next page onwards. **§3.3**; 96

**`\dynamicswitchoffnextodd`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch off the listed dynamic frames from the next odd page onwards (after that they will be off for both even and odd pages). **§3.3**; 97

**`\dynamicswitchoffnextoddonly`**{⟨*ID-list*⟩} *modifier:* *
flowfram v1.14+

Switch off the listed dynamic frames for the next odd page only. **§3.3**; 97

**`\dynamicswitchoffnextonly`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch off the listed dynamic frames for the next page only. **§3.3**; 97

**`\dynamicswitchonnext`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed dynamic frames from the next page onwards. **§3.3**; 96

**`\dynamicswitchonnextodd`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed dynamic frames from the next odd page onwards (after that they will show on both even and odd pages). **§3.3**; 97

**`\dynamicswitchonnextoddonly`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed dynamic frames for the next odd page only. **§3.3**; 97

**`\dynamicswitchonnextonly`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed dynamic frames for the next page only. **§3.3**; 97

## E

```
\enableminitoc[⟨section-type⟩]                                flowfram
(preamble only)
```

Enables mini-tocs at the start of the given sectional unit. **§6.2**; 144

```
\enablethumbtabs                                               flowfram
```

The thumbtab frames created by `\makethumbtabs` will have the hide attribute switched off and thumbtab information can start being written to the `ttb` file. Any thumbtab sectional units that occur before this command won't have a corresponding thumbtab. **§6.1**; 139

## F

```
\FFabove                         initial: above         flowfram v1.11+
```

Text used to describe one frame is above another. **§4.3**; 105

```
\FFaboveleft                     initial: above left    flowfram v1.11+
```

Text used to describe one frame above left from another. **§4.3**; 104

```
\FFaboveright                    initial: above right   flowfram v1.11+
```

Text used to describe one frame above right from another. **§4.3**; 104

**\ffareaheight** flowfram

Used by commands like `\computeflowframearea` to store the calculated height.

**\ffareawidth** flowfram

Used by commands like `\computeflowframearea` to store the calculated width.

**\ffareax** flowfram

Used by commands like `\computeflowframearea` to store the calculated $x$ position.

**\ffareay** flowfram

Used by commands like `\computeflowframearea` to store the calculated $y$ position.

**\FFbelow** *initial:* `below` flowfram v1.11+

Text used to describe one frame is below another. **§4.3**; 105

**\FFbelowleft** *initial:* `below left` flowfram v1.11+

Text used to describe one frame below left from another. **§4.3**; 105

**\FFbelowright** *initial:* `below right` flowfram v1.11+

Text used to describe one frame below right from another. **§4.3**; 105

**\ffchapterdefaultfont** flowfram v2.0+
(only if book or report class)

The default font declarations for the chapter heading. **§2.4.1**; 58

**\ffchapterheadstyle** flowfram v2.0+
(only if book or report class)

The paragraph style for the chapter header. **§2.4.1**; 58

**\ffchapternamenum**{⟨*name*⟩}{⟨*number*⟩} flowfram v2.0+
(only if book or report class)

Typesets the chapter name (Chapter/Appendix) and number. **§2.4.1**; 59

**\ffchapternamenumfont**{⟨*text*⟩} flowfram v2.0+
(only if book or report class)

The font command for the chapter name and number. **§2.4.1**; 59

**\ffchapterpostheadskip** flowfram v2.0+
(only if book or report class)

Vertical space after chapter header. **§2.4.1**; 58

**\ffchapterpostnamenum** flowfram v2.0+
(only if book or report class)

Separator between the chapter number and title. **§2.4.1**; 59

**\ffchapterpreheadskip** flowfram v2.0+
(only if book or report class)

Vertical space before chapter header. **§2.4.1**; 58

**\ffchaptertitlefont**{⟨*text*⟩} flowfram v2.0+
(only if book or report class)

The font command for the chapter title. **§2.4.1**; 59

**\ffcolumnseprule** *initial:* `2pt` flowfram v1.09+

The width/height of the rules created by `\insertvrule`/`\inserthrule`. **§5.4.4**; 132

**\ffcontinuedtextfont**{⟨*text*⟩} flowfram

Used \ffcontinuedtextlayout to apply the font for the continuation text. §2.3.1; 46

**\ffcontinuedtextlayout**{⟨*text*⟩} flowfram

Used by \continueonframe to adjust the current paragraph so that it ends flushed with the right margin and adds the continuation text flush right on the next line. §2.3.1; 46

**\ffdefaultcontinuetext** *initial: empty* flowfram v2.0+

Default used by both \ffdefaultstaticcontinuetext and \ffdefault-dynamiccontinuetext. §2.3.1; 45

**\ffdefaultdynamiccontinuetext**{⟨*IDN1*⟩}{⟨*IDN2*⟩}
*initial:* \ffdefaultcontinuetext flowfram v2.0+

Default continuation text if the optional argument of \continueonframe is omitted in dynamic frames. The arguments are the IDN of the current frame and of the continuation frame (not the IDL, regardless of the encapsulating command or environment). §2.4.3; 66

**\ffdefaultpostcontinued** flowfram v2.0+

Default behaviour of \ffstaticpostcontinued and \ffdynamicpost-continued. §2.3.1; 47

**\ffdefaultstaticcontinuetext**{⟨*IDN1*⟩}{⟨*IDN2*⟩}
*initial:* `\ffdefaultcontinuetext` flowfram v2.0+

Default continuation text if the optional argument of `\continueonframe` is omitted in static frames. The arguments are the IDN of the current frame and of the continuation frame (not the IDL, regardless of the encapsulating command or environment). **§2.3.1**; 45

**\ffdynamicpostcontinued**{⟨*IDN1*⟩}{⟨*IDN2*⟩}
*initial:* `\ffdefaultpostcontinued` flowfram v2.0+

Placed at the start of a dynamic frame with continued content where ⟨*IDN1*⟩ is the IDN of the previous frame and ⟨*IDN2*⟩ is the IDN of the continuation frame. **§2.4.3**; 67

**\ffhrule**{⟨*offset*⟩}{⟨*width*⟩}{⟨*height*⟩} flowfram v1.11+

Used to draw the rule for `\inserthrule`. **§5.4.4**; 134

**\fflabelfont** flowfram

Font declarations used for the annotation labels shown in draft mode. **§7.1**; 147

**\fflabelsep** *initial:* `1pt` flowfram

When drawing the bounding boxes in draft mode (but not for the typeblock), this is the distance between the bounding box outline and the label. **§7.2**; 148

**\FFleft** *initial:* `on the left` flowfram v1.11+

Text used to describe one frame is to the left of another. **§4.3**; 105

**\FFoverlap** *initial:* `overlap` flowfram v1.11+

Text used to describe overlapping frames. **§4.3**; 105

**\ffprechapterhook** *initial:* *empty* flowfram v1.14+

Hook added to the start of `\chapter`. **§1.6**; 32

**\FFright** *initial:* `on the right` flowfram v1.11+

Text used to describe one frame is to the right of another. **§4.3**; 105

**\ffruledeclarations** *initial:* *empty* flowfram v1.11+

Used by `\insertvrule` and `\inserthrule` to apply any declarations that may alter the way that the rule appears. **§5.4.4**; 133

**\ffstaticpostcontinued**{⟨*IDN1*⟩}{⟨*IDN2*⟩}
*initial:* `\ffdefaultpostcontinued` flowfram v2.0+

Placed at the start of a static frame with continued content where ⟨*IDN1*⟩ is the IDN of the previous frame and ⟨*IDN2*⟩ is the IDN of the continuation frame. **§2.3.1**; 47

**`\ffswapoddeven`**{⟨*ID-list*⟩} *modifier:* `*` flowfram

Swap the odd and even co-ordinates for all listed flow frames. **§3.1.3**; 84

**`\fftolerance`** *initial:* `2pt` flowfram v1.14+

The tolerance used when determining whether or not to warn when moving between flow frames of different widths. **§2.1.1**; 38

**`\ffvadjustfalse`** flowfram

Sets the `\ifffvadjust` conditional to false. **§5.1**; 113

**`\ffvadjusttrue`** flowfram

Sets the `\ifffvadjust` conditional to true. **§5.1**; 114

**`\ffvrule`**{⟨*offset*⟩}{⟨*width*⟩}{⟨*height*⟩} flowfram v1.11+

Used to draw the rule for `\insertvrule`. **§5.4.4**; 133

**`\finishthispage`** flowfram

Similar to the way `\clearpage` works for normal two-column mode, this command issues the correct number of `\newpage` needed to finish the current page to allow a page break even if the current flow frame isn't the last available for the current page. **§2.1.1**; 39

**`\FLFsimpar`** flowfram v2.0+

Simulate a paragraph break inside `\shapepar` or `\Shapepar`.

**`\flowaddexclusion`**{⟨*IDN*⟩}{⟨*list*⟩} flowfram v1.14+

Appends ⟨*list*⟩ to the exclusion list for the flow frame identified by its IDN. **§3.1.1**; 74

**`\flowfram@doc@htmlopts`**{⟨*n*⟩}{⟨*key=value list*⟩}{⟨*type*⟩}{⟨*IDN*⟩}{⟨*page*⟩}{⟨*absolute-page*⟩} flowfram v2.0+

Does nothing. This command is written to the `aux` file if the html option is used in the document environment. It's provided for the benefit of LaTeX to HTML parsers. **§3.1.5**; 89

**`\flowfram@preamble@htmlopts`**{⟨*n*⟩}{⟨*key=value list*⟩}{⟨*type*⟩}{⟨*IDN*⟩}{⟨*page*⟩}{⟨*absolute-page*⟩} flowfram v2.0+

Does nothing. This command is written to the `aux` file if the html option is used in the preamble. It's provided for the benefit of LaTeX to HTML parsers. **§3.1.5**; 89

**`\flowframchapterheader`**{⟨*text*⟩} flowfram v2.0

Used by the ffheadings page style to format the chapter mark. **§2.4.2**; 61

**`\flowframeevenx`**{⟨*IDN*⟩} flowfram

Expands to the even-page $x$-coordinate of the flow frame identified by its IDN. **§3.1.3**; 82

**`\flowframeeveny`**{⟨*IDN*⟩} flowfram

Expands to the even-page $y$-coordinate of the flow frame identified by its IDN. **§3.1.3**; 82

**`\flowframeheight`**{⟨*IDN*⟩} flowfram

Expands to the height of the flow frame identified by its IDN. **§3.1.3**; 80

**`\flowframerule`** flowfram

The width of the rule for the border of frames, if the standard border is used. This is initialised to `\fboxrule` when flowfram loads. **§7.2**; 148

**`\flowframesep`** flowfram

The gap between the text and the border of frames, if the standard border is used. This is initialised to `\fboxsep` when flowfram loads. **§7.2**; 148

`\flowframeshowlayout` flowfram

Finishes the current page, temporarily sets draft mode, and prints an empty page. **§1.4**; 30

`\flowframewidth{⟨IDN⟩}` flowfram

Expands to the width of the flow frame identified by its IDN. **§3.1.3**; 80

`\flowframex{⟨IDN⟩}` flowfram

Expands to the $x$-coordinate of the flow frame identified by its IDN. **§3.1.3**; 83

`\flowframey{⟨IDN⟩}` flowfram

Expands to the $y$-coordinate of the flow frame identified by its IDN. **§3.1.3**; 84

`\flowframheaderchapprefix` *initial: varies* flowfram v2.0

Prefix inserted before the chapter number. **§2.4.2**; 62

`\flowframheadersep` *initial: varies* flowfram v2.0

Separator used between the chapter or section number and the title (when the number is shown). **§2.4.2**; 62

**`\FlowFramRestoreOR`** flowfram v2.0+
(not preamble)

Restore the original output routine (the current page will be finished and output first). **§1.5**; 31

**`\flowframsectionheader`**{⟨*text*⟩} flowfram v2.0

Used by the ffheadings page style to format the section mark. **§2.4.2**; 61

**`\flowframsetup`**{⟨*key=value list*⟩} flowfram v2.0+

Set any options that may be adjusted after the package has loaded. **§1.2**; 10

**`\flowframsubsectionheader`**{⟨*text*⟩} flowfram v2.0

Used by the ffheadings page style to format the subsection mark.

**`\flowframtkencapobject`**{⟨*n*⟩}{⟨*Java class name*⟩}{⟨*description*⟩}{⟨*tag*⟩}{⟨*pgf point*⟩}{⟨*width*⟩}{⟨*height*⟩}{⟨*object*⟩}
flowframtkutils v2.0+

Encapsulates an object that forms part of an image. **§8.4.2**; 161

**\flowframtkendobject**{⟨*n*⟩}{⟨*Java class name*⟩}{⟨*description*⟩}{⟨*tag*⟩}{⟨*pgf point*⟩}{⟨*width*⟩}{⟨*height*⟩}
flowframtkutils v2.0+

Marks the end of an object that forms part of the image. Should always be paired with a matching `\flowframtkstartobject`. **§8.4.2**; 160

**\flowframtkimageinfo**{⟨*key=value list*⟩} flowframtkutils v2.0+

Set the document title and creation date. **§8.4.2**; 159

**\flowframtkimgtitlechar**{⟨*char*⟩}{⟨*PDF replacement*⟩}
flowframtkutils v2.0+

Normally just expands to ⟨*char*⟩. **§8.4.2**; 160

**\flowframtkSetCreationDate**{⟨*PDF date*⟩} flowframtkutils v2.0+

Set the PDF creation date. **§8.4.2**; 159

**\flowframtkSetTitle**{⟨*title*⟩} flowframtkutils v2.0+

Set the document title. **§8.4.2**; 159

**\flowframtkstartobject**{⟨*n*⟩}{⟨*Java class name*⟩}{⟨*description*⟩}{⟨*tag*⟩}{⟨*pgf point*⟩}{⟨*width*⟩}{⟨*height*⟩}
flowframtkutils v2.0+

Marks the start of an object that forms part of the image. Should always be paired with a matching `\flowframtkendobject`. **§8.4.2**; 160

**\FlowFramTkUtilsOverlayEncapSetup**{⟨*key=value list*⟩}
flowframtkutils v2.0+

Set the options for `\FlowFramTkUtilsSetOverlayEncap`. **§8.4.2**; 163

**\FlowFramTkUtilsSetOverlayEncap**[⟨*key=value list*⟩]
flowframtkutils v2.0+

Redefines `\flowframtkencapobject` to use a low-level command (`\flowframtkutils_uncover_encap:nnnnnnnn`) that uncovers objects according to the settings given in the optional argument. The settings may be changed later with `\FlowFramTkUtilsOverlayEncapSetup`. **§8.4.2**; 162

**\flowframtkutils_uncover_encap:nnnnnnnn** {⟨*n*⟩} {⟨*Java class name*⟩} {⟨*description*⟩} {⟨*tag*⟩} {⟨*pgf point*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*object*⟩}
flowframtkutils v2.0+

The low-level function used by `\flowframtkencapobject` when enabled with `\FlowFramTkUtilsSetOverlayEncap`. **§8.4.2**; 163

**`\flowframtkutils_uncover_from:nn`** { ⟨*number*⟩ } { ⟨*content*⟩ }
flowframtkutils v2.0+

Used by `\flowframtkutils_uncover_encap:nnnnnnnn` to uncover an object from ⟨*number*⟩. **§8.4.2**; 163

**`\flowframtkutils_uncover:nn`** { ⟨*number*⟩ } { ⟨*content*⟩ }
flowframtkutils v2.0+

Used by `\flowframtkutils_uncover_encap:nnnnnnnn` to uncover an object on a single overlay. **§8.4.2**; 163

**`\FlowFramUnrestoreOR`** flowfram v2.0+
(not preamble)

Restore flowfram's modified output routine (the current page will be finished and output first). **§1.5**; 32

**`\flowsetexclusion`**{⟨*IDN*⟩}{⟨*list*⟩} flowfram v1.14+

Sets the exclusion list for the flow frame identified by its IDN. **§3.1.1**; 74

**`\flowsetpagelist`**{⟨*IDN*⟩}{⟨*page-list*⟩ | ⟨*keyword*⟩} flowfram v1.14+

Sets the page list for the flow frame identified by its IDN. **§3.1.1**; 72

**\flowswitchoffnext**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch off the listed flow frames from the next page onwards. §3.3; 95

**\flowswitchoffnextodd**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch off the listed flow frames from the next odd page onwards (after that they will be off for both even and odd pages). §3.3; 96

**\flowswitchoffnextoddonly**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch off the listed flow frames for the next odd page only. §3.3; 96

**\flowswitchoffnextonly**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch off the listed flow frames for the next page only. §3.3; 96

**\flowswitchonnext**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed flow frames from the next page onwards. §3.3; 95

**\flowswitchonnextodd**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed flow frames from the next odd page onwards (after that they will show on both even and odd pages). **§3.3**; 95

**`\flowswitchonnextoddonly`**`{`⟨*ID-list*⟩`}` *modifier:* `*` flowfram v1.14+

Switch on the listed flow frames for the next odd page only. **§3.3**; 96

**`\flowswitchonnextonly`**`{`⟨*ID-list*⟩`}` *modifier:* `*` flowfram v1.14+

Switch on the listed flow frames for the next page only. **§3.3**; 96

**`\framebreak`**`[`⟨*n*⟩`]` flowfram

Forces a break from one frame to another. This creates a paragraph break in order to ensure the output routine adjusts to the new frame width but gives the effect of a continuous paragraph (although this may cause excess spacing). **§2.1.1**; 37

## G

**`\getdynamicbounds`**`{`⟨*ID*⟩`}` *modifier:* `*` flowfram

Gets the bounds for the dynamic frame identified by ⟨*ID*⟩ (the IDL for the starred version and the IDN for the unstarred version) and stores the results in `\ffareax`, `\ffareay`, `\ffareawidth` and `\ffareaheight`. **§4.2**; 103

**\getdynamicid**{⟨*cmd*⟩}{⟨*IDL*⟩} flowfram v1.11+

Defines the command ⟨*cmd*⟩ to expand to the IDN of the dynamic frame identified by its IDL. **§2.4**; 51

**\getdynamiclabel**{⟨*IDN*⟩} flowfram v1.11+

Expands to the IDL of the dynamic frame identified by its IDN. **§2.4**; 51

**\getflowbounds**{⟨*ID*⟩} *modifier:* * flowfram

Gets the bounds for the flow frame identified by ⟨*ID*⟩ (the IDL for the starred version and the IDN for the unstarred version) and stores the results in `\ffareax`, `\ffareay`, `\ffareawidth` and `\ffareaheight`. **§4.2**; 103

**\getflowid**{⟨*cmd*⟩}{⟨*IDL*⟩} flowfram v1.11+

Defines the command ⟨*cmd*⟩ to expand to the IDN of the flow frame identified by its IDL. **§2.1**; 36

**\getflowlabel**{⟨*IDN*⟩} flowfram v1.11+

Expands to the IDL of the flow frame identified by its IDN. **§2.1**; 36

**\getstaticbounds**{⟨*ID*⟩} *modifier:* * flowfram

Gets the bounds for the static frame identified by ⟨*ID*⟩ (the IDL for the starred version and the IDN for the unstarred version) and stores the results in `\ffareax`, `\ffareay`, `\ffareawidth` and `\ffareaheight`. **§4.2**; 103

**\getstaticid**{⟨*cmd*⟩}{⟨*IDL*⟩} flowfram v1.11+

Defines the command ⟨*cmd*⟩ to expand to the IDN of the static frame identified by its IDL. **§2.2**; 40

**\getstaticlabel**{⟨*IDN*⟩} flowfram v1.11+

Expands to the IDL of the static frame identified by its IDN. **§2.2**; 40

## H

**\hNtone**[⟨*page-list*⟩][⟨*y-offset*⟩]{⟨*n*⟩}{⟨*H1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}...{⟨*Hn*⟩}{⟨*Cn*⟩}{⟨*Ln*⟩} flowfram

As `\htwotone` but creates ⟨*n*⟩ static frames. **§5.4.2**; 130

**\hNtoneleft**[⟨*page-list*⟩][⟨*y-offset*⟩]{⟨*n*⟩}{⟨*W*⟩}{⟨*H1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}...{⟨*Hn*⟩}{⟨*Cn*⟩}{⟨*Ln*⟩} flowfram

As `\htwotoneleft` but creates ⟨*n*⟩ static frames. **§5.4.2**; 131

**\hNtoneright**[⟨*page-list*⟩][⟨*y-offset*⟩]{⟨*n*⟩}{⟨*W*⟩}{⟨*H1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}...{⟨*Hn*⟩}{⟨*Cn*⟩}{⟨*Ln*⟩} flowfram

As `\htwotoneright` but creates ⟨*n*⟩ static frames. **§5.4.2**; 131

**\htwotone**[⟨*page-list*⟩][⟨*y-offset*⟩]{⟨*H1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*H2*⟩}{⟨*C2*⟩}{⟨*L2*⟩} flowfram

Defines two vertically stacked static frames where the first frame has height ⟨*H1*⟩, background colour ⟨*C1*⟩ and label ⟨*L1*⟩, and the second frame has height ⟨*H2*⟩, background colour ⟨*C1*⟩ and label ⟨*L2*⟩. Both frames have the width set to `\paperwidth` and are positioned so that their left edge is at the left side of the page, with the first frame offset from the bottom edge of the page by ⟨*y-offset*⟩. Note that the colour specifications ⟨*C1*⟩ and ⟨*C2*⟩ should be enclosed in braces, for example `{[gray]{0.5}}` not `[gray]{0.5}`. **§5.4.2**; 130

**\htwotoneleft**[⟨*page-list*⟩][⟨*y-offset*⟩]{⟨*W*⟩}{⟨*H1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*H2*⟩}{⟨*C2*⟩}{⟨*L2*⟩} flowfram

As `\htwotone` but instead of being the full width of the page, the width for both frames is set to ⟨*W*⟩. **§5.4.2**; 130

**\htwotoneright**[⟨*page-list*⟩][⟨*y-offset*⟩]{⟨*W*⟩}{⟨*H1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*H2*⟩}{⟨*C2*⟩}{⟨*L2*⟩} flowfram

As `\htwotoneleft` but horizontally offset so that the right edge of the frames is along the right edge of the page. **§5.4.2**; 131

## I

**\ifffvadjust** ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iftrue` flowfram

If true, the column commands, such as `\onecolumninarea` will adjust the specified height to ensure that it is an integer multiple of `\baselineskip`. **§5.1**; 114

**\iflefttorightcolumns** ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iftrue` flowfram v1.12+

Set to `\iftrue` by the `LR` package option and to `\iffalse` by the `RL` option, this conditional determines the order of frames created by the column commands, such as `\twocolumn`. **§5.3**; 127

**\IfSavedRelativeLocationAbove**{⟨*label*⟩}{⟨*true*⟩}{⟨*false*⟩} flowfram v2.0+

Tests if the saved result from `\SaveRelativeFrameLocation` indicates that the first frame is above the second frame, without regard to their horizontal positions. **§4.3**; 109

**`\IfSavedRelativeLocationBelow`**{⟨*label*⟩}{⟨*true*⟩}{⟨*false*⟩}
flowfram v2.0+

Tests if the saved result from `\SaveRelativeFrameLocation` indicates that the first frame is below the second frame, without regard to their horizontal positions. **§4.3**; 109

**`\IfSavedRelativeLocationEq`**{⟨*label*⟩}{⟨*cmd*⟩}{⟨*true*⟩}{⟨*false*⟩}
flowfram v2.0+

Tests if the saved result from `\SaveRelativeFrameLocation` matches ⟨*cmd*⟩, which should be one of the relative placeholder commands such as `\FFabove`. **§4.3**; 108

**`\IfSavedRelativeLocationLeft`**{⟨*label*⟩}{⟨*true*⟩}{⟨*false*⟩}
flowfram v2.0+

Tests if the saved result from `\SaveRelativeFrameLocation` indicates that the first frame is left of the second frame, without regard to their vertical positions. **§4.3**; 109

**`\IfSavedRelativeLocationRight`**{⟨*label*⟩}{⟨*true*⟩}{⟨*false*⟩}
flowfram v2.0+

Tests if the saved result from `\SaveRelativeFrameLocation` indicates that the first frame is right of the second frame, without regard to their vertical positions. **§4.3**; 109

**\ifshowframebbox** ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`
flowfram

If true, the bounding boxes of all frames will be drawn in grey on the page. **§1.4**; 30

**\ifshowmargins** ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`
flowfram

If true, the flow frame margins will be drawn in grey on the page. **§1.4**; 29

**\ifshowtypeblock** ⟨*true*⟩`\else` ⟨*false*⟩`\fi` *initial:* `\iffalse`
flowfram

If true, the typeblock will be drawn in grey on the page. **§1.4**; 29

**\includeteximage** [⟨*key=value list*⟩] {⟨*filename*⟩} flowframtkutils v2.0+

If the optional argument is omitted, this just inputs ⟨*filename*⟩, otherwise it applies a transformation according to the options. **§8.4.2**; 157

**\inserthrule** [⟨*x-left*⟩] [⟨*x-right*⟩] {⟨*type-1*⟩} {⟨*ID1*⟩} {⟨*type-2*⟩} {⟨*ID2*⟩} *modifier:* * flowfram

Creates a horizontal rule between two frames. The starred version references the frames by their IDL otherwise the frames are referenced by their IDN. The rule will extend from the leftmost point of the two frames minus ⟨*x-left*⟩ to the rightmost point of the two frames plus ⟨*x-right*⟩. **§5.4.4**; 133

**\insertvrule**[⟨*y-top*⟩][⟨*y-bottom*⟩]{⟨*type-1*⟩}{⟨*ID1*⟩}{⟨*type-2*⟩}{⟨*ID2*⟩} *modifier:* * flowfram

Creates a vertical rule between two frames. The starred version references the frames by their IDL otherwise the frames are referenced by their IDN. The rule will extend from the highest point of the two frames plus ⟨*y-top*⟩ to the lowest point of the two frames minus ⟨*y-bottom*⟩. **§5.4.4**; 132

## J

**\jdrimagebox**{⟨*image*⟩} flowframtkutils

Used for encapsulated images, this is designed to prevent problems if numerical rounding errors cause the image to be larger than the available area. **§8.4.2**; 158

**\jdroutline**{⟨*pdf-trans code*⟩}{⟨*pst-char code*⟩}{⟨*text*⟩} flowframtkutils

If enabled by the `outline` option, this will attempt to render ⟨*text*⟩ as an outline using ⟨*pdf-trans code*⟩ (if `pdf-trans.tex` was loaded) or ⟨*pst-char code*⟩ (if pst−char was loaded). If not enabled, this command will generate a warning and just do ⟨*text*⟩. **§8.4.2**; 158

## L

**\labelflow**{⟨*label*⟩} flowfram v1.11+

Labels the current flow frame with `\label` so that its displayed index (displayedframe) can be referenced with `\ref`. **§7.3**; 151

**\labelflowidn**{⟨*label*⟩} flowfram v1.11+

Labels the current flow frame with `\label` so that its IDN can be referenced with `\ref`. **§7.3**; 150

**\lefttorightcolumnsfalse** flowfram v1.12+

Sets the `\iflefttorightcolumns` conditional to false. **§5.3**; 127

**\lefttorightcolumnstrue** flowfram v1.12+

Sets the `\iflefttorightcolumns` conditional to true. **§5.3**; 127

## M

**\makebackgroundframe**[⟨*page-list*⟩] [⟨*label*⟩] flowfram

Creates a single static frame that covers the entire page. This command, if required, should come before any other static frames that are shown on any of the pages in ⟨*page-list*⟩ otherwise it will obscure them. **§5.4.3**; 131

**`\makedfheaderfooter`** flowfram
(preamble only)

Creates the header and footer dynamic frames to use for the page header and footer. They can then be moved, resized or styled as required. This command will set the ffheadings package style and, depending on the `dynamic-page-style` setting, may adjust the standard page styles. **§2.4.2**; 60

**`\makethumbtabs`**[⟨*y-offset*⟩]{⟨*height*⟩}[⟨*section-type*⟩] flowfram
(preamble only)

Create thumbtabs for the given section type with the given height, where the first thumbtab is offset from the top of the typeblock by ⟨*y-offset*⟩. This command first inputs the `ttb` file to read the information from the previous run and the opens the file to allow information from this run to be written to it. Note that the thumbtab dynamic frames will have the hide attribute switched on. This will be switched off by `\enablethumbtabs`. **§6.1**; 137

**`\minitocstyle`**{⟨*content*⟩} flowfram

The style used to format the mini-toc. **§6.2**; 145

## N

**`\Ncolumn`**[⟨*page-list*⟩]{⟨*n*⟩}[⟨*label*⟩] (preamble only)

Creates ⟨*n*⟩ flow frames that fit in the typeblock separated by the distance given by `\column-`

`sep`, with the height adjusted if the conditional `\ifffvadjust` is true. **§5.1**; 114

**`\Ncolumnbottom`**[⟨*page-list*⟩]{⟨*type*⟩}{⟨*n*⟩}{⟨*top-height*⟩}[⟨*label*⟩]
flowfram
(preamble only)

As `\Ncolumnbottominarea` where the area is given by the typeblock. **§5.2**; 125

**`\Ncolumnbottominarea`**[⟨*page-list*⟩]{⟨*type*⟩}{⟨*n*⟩}{⟨*top-height*⟩}{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩] flowfram
(preamble only)

Creates a frame of the given type (flow, static or dynamic) and height ⟨*top-height*⟩ and ⟨*n*⟩ side-by-side flow frames that all fit in the given area such that the first frame is below the other ⟨*n*⟩ frames, separated by `\vcolumnsep`. The height of the ⟨*n*⟩ side-by-side flow frames is adjusted if `\ifffvadjust` is true. The horizontal gap between them is given by `\columnsep`. **§5.2**; 117

**`\NcolumnDbottom`**[⟨*page-list*⟩]{⟨*n*⟩}{⟨*top-height*⟩}[⟨*label*⟩] flowfram
(preamble only)

As `\Ncolumnbottom` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 126

**\NcolumnDbottominarea** [⟨*page-list*⟩] {⟨*n*⟩} {⟨*bottom-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\Ncolumnbottominarea` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 125

**\NcolumnDtop** [⟨*page-list*⟩] {⟨*n*⟩} {⟨*top-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\Ncolumntop` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 121

**\NcolumnDtopinarea** [⟨*page-list*⟩] {⟨*n*⟩} {⟨*top-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\Ncolumntopinarea` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 121

**\Ncolumninarea** [⟨*page-list*⟩] {⟨*n*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*labels*⟩] flowfram
(preamble only)

Creates ⟨*n*⟩ flow frames that span the given area separated by the distance given by `\columnsep`, with the height adjusted if the conditional `\ifffvadjust` is true. **§5.1**; 115

**\NcolumnSbottom** [⟨*page-list*⟩] {⟨*n*⟩} {⟨*top-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As \Ncolumnbottom with the ⟨*type*⟩ set to static. §5.2; 126

**\NcolumnSbottominarea** [⟨*page-list*⟩] {⟨*n*⟩} {⟨*bottom-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As \Ncolumnbottominarea with the ⟨*type*⟩ set to static. §5.2; 125

**\NcolumnStop** [⟨*page-list*⟩] {⟨*n*⟩} {⟨*top-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As \Ncolumntop with the ⟨*type*⟩ set to static. §5.2; 121

**\NcolumnStopinarea** [⟨*page-list*⟩] {⟨*n*⟩} {⟨*top-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As \Ncolumntopinarea with the ⟨*type*⟩ set to static. §5.2; 120

**\Ncolumntop** [⟨*page-list*⟩] {⟨*type*⟩} {⟨*n*⟩} {⟨*top-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As \Ncolumntopinarea where the area is given by the typeblock. §5.2; 121

**`\Ncolumntopinarea`** [⟨*page-list*⟩] {⟨*type*⟩} {⟨*n*⟩} {⟨*top-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

Creates a frame of the given type (flow, static or dynamic) and height ⟨*top-height*⟩ and ⟨*n*⟩ side-by-side flow frames that all fit in the given area such that the first frame is above the other ⟨*n*⟩ frames, separated by `\vcolumnsep`. The height of the ⟨*n*⟩ side-by-side flow frames is adjusted if `\ifffvadjust` is true. The horizontal gap between them is given by `\columnsep`. **§5.2**; 116

**`\newdynamicframe`** [⟨*page-list*⟩] {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] *modifier:* * flowfram
(preamble only)

Defines a new dynamic frame with the given dimensions and location. If ⟨*label*⟩ is provided, that will be set as the frame's IDL otherwise the IDL will be the same as the IDN. **§2.4**; 50

**`\newflowframe`** [⟨*page-list*⟩] {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] *modifier:* * flowfram
(preamble only)

Defines a new flow frame with the given dimensions and location. If ⟨*label*⟩ is provided, that will be set as the frame's IDL otherwise the IDL will be the same as the IDN. **§2.1**; 35

**\newframe** [⟨*page-list*⟩] {⟨*frame-type*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩}
[⟨*label*⟩] *modifier:* * flowfram
(preamble only)

Defines a new frame of the given ⟨*frame-type*⟩ with the given dimensions and location. If ⟨*label*⟩ is provided, that will be set as the frame's IDL otherwise the IDL will be the same as the IDN. **§5.2**; 126

**\newstaticframe** [⟨*page-list*⟩] {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩}
[⟨*label*⟩] *modifier:* * flowfram
(preamble only)

Defines a new static frame with the given dimensions and location. If ⟨*label*⟩ is provided, that will be set as the frame's IDL otherwise the IDL will be the same as the IDN. **§2.2**; 40

## O

**\onecolumn** [⟨*page-list*⟩] [⟨*label*⟩] (redefined by flowfram)

Creates a flow frame that spans the typeblock, with the height adjusted if the conditional `\iffvadjust` is true. **§5.1**; 112

**\onecolumnbottom**[⟨*page-list*⟩]{⟨*type*⟩}{⟨*bottom-height*⟩}[⟨*label*⟩]
flowfram
(preamble only)

As `\onecolumnbottominarea` where the area is given by the typeblock. **§5.2**; 122

**\onecolumnbottominarea**[⟨*page-list*⟩]{⟨*type*⟩}{⟨*bottom-height*⟩}{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩] flowfram
(preamble only)

Creates a frame of the given type (flow, static or dynamic) and height ⟨*bottom-height*⟩ and a flow frame that fit in the given area so that the first frame is below the second frame, separated by `\vcolumnsep`. The height of the second frame is adjusted if `\ifffvadjust` is true. **§5.2**; 116

**\onecolumnDbottom**[⟨*page-list*⟩]{⟨*bottom-height*⟩}[⟨*label*⟩] flowfram
(preamble only)

As `\onecolumnbottom` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 123

**\onecolumnDbottominarea**[⟨*page-list*⟩]{⟨*bottom-height*⟩}{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩] flowfram
(preamble only)

As `\onecolumnbottominarea` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 122

**\onecolumnDtop**[⟨*page-list*⟩]{⟨*top-height*⟩}[⟨*label*⟩] flowfram
(preamble only)

As `\onecolumntop` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 118

**\onecolumnDtopinarea**[⟨*page-list*⟩]{⟨*top-height*⟩}{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩] flowfram
(preamble only)

As `\onecolumntopinarea` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 118

**\onecolumninarea**[⟨*page-list*⟩]{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩] flowfram
(preamble only)

Creates a flow frame that spans the given area, with the height adjusted if the conditional `\ifffvadjust` is true. **§5.1**; 114

**\onecolumnSbottom**[⟨*page-list*⟩]{⟨*bottom-height*⟩}[⟨*label*⟩] flowfram
(preamble only)

As `\onecolumnbottom` with the ⟨*type*⟩ set to `static`. **§5.2**; 123

**`\onecolumnSbottominarea`** [⟨*page-list*⟩] {⟨*bottom-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\onecolumnbottominarea` with the ⟨*type*⟩ set to `static`. **§5.2**; 122

**`\onecolumnStop`** [⟨*page-list*⟩] {⟨*top-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\onecolumntop` with the ⟨*type*⟩ set to `static`. **§5.2**; 118

**`\onecolumnStopinarea`** [⟨*page-list*⟩] {⟨*top-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\onecolumntopinarea` with the ⟨*type*⟩ set to `static`. **§5.2**; 117

**`\onecolumntop`** [⟨*page-list*⟩] {⟨*type*⟩} {⟨*top-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\onecolumntopinarea` where the area is given by the typeblock. **§5.2**; 118

**\onecolumntopinarea** [⟨*page-list*⟩] {⟨*type*⟩} {⟨*top-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

Creates a frame of the given type (flow, static or dynamic) and height ⟨*top-height*⟩ and a flow frame that fit in the given area so that the first frame is above the second frame, separated by `\vcolumnsep`. The height of the second frame is adjusted if `\ifffvadjust` is true. **§5.2**; 116

## P

**\part** [⟨*toc*⟩] [⟨*thumbtab-title*⟩] {⟨*title*⟩} *modifier:* `*`

Modified by flowfram, this has an additional optional argument for the thumbtab titles (see `sections-extra-option` if you are using a class that also has a second optional argument). Note that the starred version also has both optional arguments.

## R

**\RefSavedRelativeLocation** {⟨*label*⟩} flowfram v2.0+

Fetches the information saved on the previous run with `\SaveRelativeFrameLocation`. **§4.3**; 108

**\relativeframelocation**{⟨*type1*⟩}{⟨*ID1*⟩}{⟨*type2*⟩}{⟨*ID2*⟩}
*modifier:* * flowfram v1.11+

Produces a textual description of the relative location of the ⟨*type1*⟩ frame identified by ⟨*ID1*⟩ to the ⟨*type2*⟩ frame identified by ⟨*ID2*⟩ (the starred version references the frames by their IDL and the unstarred version references them by their IDN). **§4.3**; 104

**\reldynamicloc**{⟨*ID1*⟩}{⟨*ID2*⟩} *modifier:* * flowfram v1.11+

Shortcut for `\relativeframelocation` where both frame types are dynamic. **§4.3**; 106

**\relflowloc**{⟨*ID1*⟩}{⟨*ID2*⟩} *modifier:* * flowfram v1.11+

Shortcut for `\relativeframelocation` where both frame types are flow. **§4.3**; 107

**\relstaticloc**{⟨*ID1*⟩}{⟨*ID2*⟩} *modifier:* * flowfram v1.11+

Shortcut for `\relativeframelocation` where both frame types are static. **§4.3**; 106

### S

**\SaveRelativeFrameLocation**{⟨*label*⟩}{⟨*type1*⟩}{⟨*ID1*⟩}{⟨*type2*⟩}{⟨*ID2*⟩} *modifier:* `*` flowfram v2.0+

Similar to `\relativeframelocation` but defers the calculations for the next LaTeX run. The result may be referenced with `\RefSavedRelativeLocation`. **§4.3**; 107

**\sdfparindent** *initial:* `0pt` flowfram

The default paragraph indentation within static and dynamic frames (may be overridden for individual frames through the parindent option). **§7.2**; 148

**\section**[⟨*toc*⟩][⟨*thumbtab-title*⟩]{⟨*title*⟩} *modifier:* `*`

Modified by flowfram, this has an additional optional argument for the thumbtab titles (see `sections-extra-option` if you are using a class that also has a second optional argument). Note that the starred version also has both optional arguments.

**\setalldynamicframes**{⟨*key=value list*⟩} flowfram

Sets the attributes in ⟨⟨*key=value list*⟩⟩ for all defined dynamic frames. **§3**; 70

**\setallflowframes**{⟨*key=value list*⟩} flowfram

Sets the attributes in ⟨⟨*key=value list*⟩⟩ for all defined flow frames. **§3**; 69

`\setallstaticframes{⟨key=value list⟩}` flowfram

Sets the attributes in ⟨⟨*key=value list*⟩⟩ for all defined static frames. **§3**; 69

`\setdynamiccontents[options]{⟨ID⟩}{⟨content⟩}` *modifier:* `*` flowfram

Sets the content of the dynamic frames identified by ⟨*ID*⟩, which is the frame's IDN (unstarred) or IDL (starred). If ⟨*options*⟩ is present, they will be applied to the frame at the same time. **§2.4**; 52

`\setdynamicframe{⟨ID-list⟩}{⟨key=value list⟩}` *modifier:* `*` flowfram

Sets the attributes for the given dynamic frames, identified by the list of IDs, which each identify a frame by its IDN (unstarred) or IDL (starred). **§3**; 70

`\setffdraftcolor` flowfram

Sets the colour of the bounding box outlines in draft mode. **§7.1**; 147

`\setffdrafttypeblockcolor` flowfram

Sets the colour of the typeblock outline in draft mode. **§7.1**; 147

**\setflowframe**{⟨*ID-list*⟩}{⟨*key=value list*⟩} *modifier:* * flowfram

Sets the attributes for the given flow frames, identified by the list of IDs, which each identify a frame by its IDN (unstarred) or IDL (starred). **§3**; 69

**\SetOneColumnFrame**{⟨*ID*⟩} *modifier:* * flowfram v2.0+

For use with `column-changes=switch`, this command identifies the flow frame to use if `\onecolumn` is encountered in the document environment. **§1.2.3**; 23

**\setstaticcontents**`[options]`{⟨*ID*⟩}{⟨*content*⟩} *modifier:* * flowfram

Sets the content of the given static frames identified by ⟨*ID*⟩, which is the frame's IDN (unstarred) or IDL (starred). If ⟨*options*⟩ is present, they will be applied to the frame at the same time. **§2.3**; 41

**\setstaticframe**{⟨*ID-list*⟩}{⟨*key=value list*⟩} *modifier:* * flowfram

Sets the attributes for the given static frames, identified by the list of IDs, which each identify a frame by its IDN (unstarred) or IDL (starred). **§3**; 69

**\setthumbtab**{⟨*index*⟩|`all`}{⟨*key=value list*⟩} flowfram

Sets the given attributes for all of the dynamic frames for the thumbtab with the identified index or for all thumbtabs if the first argument is the keyword `all`. **§6.1**; 142

**\setthumbtabindex**{⟨*index*⟩|all}{⟨*key=value list*⟩} flowfram

Sets the given attributes for the thumbtab index frame with the identified index or for all thumbtabs index frames if the first argument is the keyword `all`. **§6.1**; 142

**\SetTwoColumnFrames**[⟨*header-type*⟩][⟨*header-id*⟩]{⟨*col-1*⟩}[⟨*header-col1*⟩]{⟨*col-2*⟩}[⟨*header-col-2*⟩] *modifier:* * flowfram v2.0+

For use with `column-changes=switch`, this command identifies the frames to use if `\twocolumn` is encountered in the document environment. **§1.2.3**; 24

**\sfswapoddeven**{⟨*ID-list*⟩} *modifier:* * flowfram

Swap the odd and even co-ordinates for all listed static frames. **§3.1.3**; 85

**\showframebboxfalse** flowfram

Sets the `\ifshowframebbox` conditional to false. **§1.4**; 30

**\showframebboxtrue** flowfram

Sets the `\ifshowframebbox` conditional to true. **§1.4**; 30

**\showmarginsfalse** flowfram

Sets the `\ifshowmargins` conditional to false. §1.4; 30

**\showmarginstrue** flowfram

Sets the `\ifshowmargins` conditional to true. §1.4; 30

**\showtypeblockfalse** flowfram

Sets the `\ifshowtypeblock` conditional to false. §1.4; 29

**\showtypeblocktrue** flowfram

Sets the `\ifshowtypeblock` conditional to true. §1.4; 29

🗑 **\simpar** flowfram

Old name for `\FLFsimpar` provided for backward-compatibility.

**\staticaddexclusion**{⟨*IDN*⟩}{⟨*list*⟩} flowfram v1.14+

Appends ⟨*list*⟩ to the exclusion list for the static frame identified by its IDN. §3.1.1; 74

**\staticframe** flowfram

Temporary box register used in as an intermediate when setting the contents of a static frame. **§2.3**; 42

**\staticframeevenx**{⟨*IDN*⟩} flowfram

Expands to the even-page $x$-coordinate of the static frame identified by its IDN. **§3.1.3**; 82

**\staticframeeveny**{⟨*IDN*⟩} flowfram

Expands to the even-page $y$-coordinate of the static frame identified by its IDN. **§3.1.3**; 83

**\staticframeheight**{⟨*IDN*⟩} flowfram

Expands to the height of the static frame identified by its IDN. **§3.1.3**; 80

**\staticframewidth**{⟨*IDN*⟩} flowfram

Expands to the width of the static frame identified by its IDN. **§3.1.3**; 80

**\staticframex**{⟨*IDN*⟩} flowfram

Expands to the $x$-coordinate of the static frame identified by its IDN. **§3.1.3**; 83

```
\staticframey{⟨IDN⟩}                                          flowfram
```

Expands to the $y$-coordinate of the static frame identified by its IDN. §3.1.3; 84

```
\staticsetexclusion{⟨IDN⟩}{⟨list⟩}                            flowfram v1.14+
```

Sets the exclusion list for the static frame identified by its IDN. §3.1.1; 74

```
\staticsetpagelist{⟨IDN⟩}{⟨page-list⟩ | ⟨keyword⟩}            flowfram v1.14+
```

Sets the page list for the static frame identified by its IDN. §3.1.1; 72

```
\staticswitchoffnext{⟨ID-list⟩}               modifier: *      flowfram v1.14+
```

Switch off the listed static frames from the next page onwards. §3.3; 98

```
\staticswitchoffnextodd{⟨ID-list⟩}            modifier: *      flowfram v1.14+
```

Switch off the listed static frames from the next odd page onwards (after that they will be off for both even and odd pages). §3.3; 98

```
\staticswitchoffnextoddonly{⟨ID-list⟩}                        modifier: *
flowfram v1.14+
```

Switch off the listed static frames for the next odd page only. §3.3; 98

**`\staticswitchoffnextonly`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch off the listed static frames for the next page only. **§3.3**; 98

**`\staticswitchonnext`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed static frames from the next page onwards. **§3.3**; 97

**`\staticswitchonnextodd`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed static frames from the next odd page onwards (after that they will show on both even and odd pages). **§3.3**; 98

**`\staticswitchonnextoddonly`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed static frames for the next odd page only. **§3.3**; 98

**`\staticswitchonnextonly`**{⟨*ID-list*⟩} *modifier:* * flowfram v1.14+

Switch on the listed static frames for the next page only. **§3.3**; 98

## T

**\thumbtabformat**{⟨*text*⟩}{⟨*height*⟩} flowfram

Used to format the thumbtab text. §6.1; 141

**\thumbtabhyperlinkformat**{⟨*anchor*⟩}{⟨*text*⟩}{⟨*height*⟩}
flowfram v2.0

If hyperref is loaded, this command will be redefined to create a hyperlink for the thumbtab frame content. §6.1; 141

**\thumbtabindex**[⟨*page-list*⟩] flowfram

Used internally by flowfram's modified `\tableofcontents` to show the thumbtab index frames. However, if `adjust-toc=off` has been used to prevent this, the thumbtab index frames can be switched on with `\thumbtabindex`. §6.1; 140

**\thumbtabindexformat**{⟨*anchor*⟩}{⟨*text*⟩}{⟨*height*⟩} flowfram

Used to format the content of the thumbtab index frames. §6.1; 141

**\thumbtabregularformat**{⟨*anchor*⟩}{⟨*text*⟩}{⟨*height*⟩} flowfram v2.0+

Used to format the content of the thumbtab (non-index) frames. §6.1; 141

**\thumbtabwidth** *initial:* `1cm` flowfram

The default width of thumbtabs. **§6.1**; 138

**\twocolumn**[⟨*page-list*⟩][⟨*label*⟩] (redefined by flowfram)

Creates two flow frames that fit in the typeblock separated by the distance given by `\columnsep`, with the height adjusted if the conditional `\ifffvadjust` is true. **§5.1**; 112

**\twocolumnbottom**[⟨*page-list*⟩]{⟨*type*⟩}{⟨*bottom-height*⟩}[⟨*label*⟩]
flowfram
(preamble only)

As `\twocolumnbottominarea` where the area is given by the typeblock. **§5.2**; 124

**\twocolumnbottominarea**[⟨*page-list*⟩]{⟨*type*⟩}{⟨*bottom-height*⟩}{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩] flowfram
(preamble only)

Creates a frame of the given type (flow, static or dynamic) and height ⟨*top-height*⟩ and two side-by-side flow frames that all fit in the given area such that the first frame is below the other two frames, separated by `\vcolumnsep`. The height of the two side-by-side flow frames is adjusted if `\ifffvadjust` is true. The horizontal gap between them is given by `\columnsep`. **§5.2**; 117

**`\twocolumnDbottom`** [⟨*page-list*⟩] {⟨*bottom-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\twocolumnbottom` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 124

**`\twocolumnDbottominarea`** [⟨*page-list*⟩] {⟨*bottom-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\twocolumnbottominarea` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 124

**`\twocolumnDtop`** [⟨*page-list*⟩] {⟨*top-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\twocolumntop` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 120

**`\twocolumnDtopinarea`** [⟨*page-list*⟩] {⟨*top-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\twocolumntopinarea` with the ⟨*type*⟩ set to `dynamic`. **§5.2**; 119

**\twocolumninarea** [⟨*page-list*⟩] {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*labels*⟩] flowfram
(preamble only)

Creates two flow frames that span the given area separated by the distance given by `\columnsep`, with the height adjusted if the conditional `\ifffvadjust` is true. **§5.1**; 114

**\twocolumnSbottom** [⟨*page-list*⟩] {⟨*bottom-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\twocolumnbottom` with the ⟨*type*⟩ set to `static`. **§5.2**; 124

**\twocolumnSbottominarea** [⟨*page-list*⟩] {⟨*bottom-height*⟩} {⟨*width*⟩} {⟨*height*⟩} {⟨*x*⟩} {⟨*y*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\twocolumnbottominarea` with the ⟨*type*⟩ set to `static`. **§5.2**; 123

**\twocolumnStop** [⟨*page-list*⟩] {⟨*top-height*⟩} [⟨*label*⟩] flowfram
(preamble only)

As `\twocolumntop` with the ⟨*type*⟩ set to `static`. **§5.2**; 120

**\twocolumnStopinarea**[⟨*page-list*⟩]{⟨*top-height*⟩}{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩] flowfram
(preamble only)

As `\twocolumntopinarea` with the ⟨*type*⟩ set to `static`. **§5.2**; 119

**\twocolumntop**[⟨*page-list*⟩]{⟨*type*⟩}{⟨*top-height*⟩}[⟨*label*⟩] flowfram
(preamble only)

As `\twocolumntopinarea` where the area is given by the typeblock. **§5.2**; 119

**\twocolumntopinarea**[⟨*page-list*⟩]{⟨*type*⟩}{⟨*top-height*⟩}{⟨*width*⟩}{⟨*height*⟩}{⟨*x*⟩}{⟨*y*⟩}[⟨*label*⟩] flowfram
(preamble only)

Creates a frame of the given type (flow, static or dynamic) and height ⟨*top-height*⟩ and two side-by-side flow frames that all fit in the given area such that the first frame is above the other two frames, separated by `\vcolumnsep`. The height of the two side-by-side flow frames is adjusted if `\ifffvadjust` is true. The horizontal gap between them is given by `\columnsep`. **§5.2**; 116

**\typeblockheight** flowfram v2.0+

A length (dimension) that stores the height of the typeblock (initialised to `\textheight`). **§1**; 2

**`\typeblockoffsety`** flowfram v2.0+

A length (dimension) that stores the vertical offset of the typeblock (initialised to the sum of `\topmargin`, `\headheight`, `\headsep` and `\voffset`). **§1**; 2

**`\typeblockwidth`** flowfram v2.0+

A length (dimension) that stores the width of the typeblock (initialised to `\textwidth`). **§1**; 2

## V

**`\vcolumnsep`** *initial:* `\columnsep` flowfram

The vertical distance between the top/bottom frame and the column flow frames when using commands like `\Ncolumntop`. **§7.2**; 149

**`\vNtone`** [⟨*page-list*⟩] [⟨*x-offset*⟩] {⟨*n*⟩} {⟨*W1*⟩} {⟨*C1*⟩} {⟨*L1*⟩} ... {⟨*Wn*⟩} {⟨*Cn*⟩} {⟨*Ln*⟩} flowfram

As `\vtwotone` but creates ⟨*n*⟩ static frames. **§5.4.1**; 129

**\vNtonebottom**[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*n*⟩}{⟨*H*⟩}{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}...{⟨*Wn*⟩}{⟨*Cn*⟩}{⟨*Ln*⟩} flowfram

As `\vtwotonebottom` but creates ⟨*n*⟩ static frames. **§5.4.1**; 130

**\vNtonetop**[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*n*⟩}{⟨*H*⟩}{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}...{⟨*Wn*⟩}{⟨*Cn*⟩}{⟨*Ln*⟩} flowfram

As `\vtwotonetop` but creates ⟨*n*⟩ static frames. **§5.4.1**; 130

**\vtwotone**[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*W2*⟩}{⟨*C2*⟩}{⟨*L2*⟩} flowfram

Defines two side-by-side static frames where the first frame has width ⟨*W1*⟩, background colour ⟨*C1*⟩ and label ⟨*L1*⟩, and the second frame has width ⟨*W2*⟩, background colour ⟨*C1*⟩ and label ⟨*L2*⟩. Both frames have the height set to `\paperheight` and are positioned so that their base is at the bottom of the page, with the first frame offset from the left side of the page by ⟨*x-offset*⟩. Note that the colour specifications ⟨*C1*⟩ and ⟨*C2*⟩ should be enclosed in braces, for example `{[gray]{0.5}}` not `[gray]{0.5}`. **§5.4.1**; 129

**\vtwotonebottom**[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*H*⟩}{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*W2*⟩}{⟨*C2*⟩}{⟨*L2*⟩} flowfram

As `\vtwotone` but instead of being the full height of the page, the height for both frames is set to ⟨*H*⟩. **§5.4.1**; 129

**\vtwotonetop**[⟨*page-list*⟩][⟨*x-offset*⟩]{⟨*H*⟩}{⟨*W1*⟩}{⟨*C1*⟩}{⟨*L1*⟩}{⟨*W2*⟩}{⟨*C2*⟩}{⟨*L2*⟩} flowfram

As `\vtwotonebottom` but vertically offset so that the top of the frames is at the top of the page. **§5.4.1**; 129

## B.2. Environment Summary

\begin{**dynamiccontents**}[⟨*options*⟩]{⟨*IDN*⟩} flowfram v1.09+

Sets the contents of the dynamic frame identified by its IDN. Note that verbatim text can't occur within the environment body. **§2.4**; 50, 52, 53, 65, 67, 184, 192, *249*

\begin{**dynamiccontents***}[⟨*options*⟩]{⟨*IDL*⟩} flowfram

Sets the contents of the dynamic frame identified by its IDL. Note that verbatim text can't occur within the environment body. **§2.4**; 53, 65, 67, 184, *249*

\begin{**staticcontents**}[⟨*options*⟩]{⟨*IDN*⟩} flowfram

Sets the contents of the static frame identified by its IDN. **§2.3**; 41, 42, 43, 47, 50, 192, *249*

```
\begin{staticcontents*}[⟨options⟩]{⟨IDL⟩}                flowfram
```

Sets the contents of the static frame identified by its IDL. **§2.3**; 41, 42, 43, *250*

```
\begin{staticfigure}                                      flowfram
```

Simulate a figure in a static or dynamic frame. **§1.3**; 28, 183, *250*

```
\begin{statictable}                                       flowfram
```

Simulate a table in a static or dynamic frame. **§1.3**; 28, 183, *250*

# C. Package Option Summary

```
\usepackage[⟨options⟩]{flowfram}
```

Package for constructing custom document columns and fixed frames. §1.2; 9, *251*

**`adjust-pre-chapter`**=⟨*boolean*⟩ *default:* `true`; *initial:* `true` flowfram v2.0+ (package option only)

If true and `\chapter` is defined, this setting will define `\ffprechapterhook` and `\chapterfirstpagestyle`. §1.2.4; 25, 32, *251*

**`adjust-toc`**=⟨*setting*⟩ flowfram v2.0+

Specifies whether or not the table of contents should be adjusted to support thumbtab indexes and mini-tocs. §1.2.1; 11, *251*

> `adjust-toc=`**`header`**
> Adjust the table of contents, including the header. 11, *251*
>
> `adjust-toc=`**`notheader`**
> Adjust the table of contents, but not the header. 11, *251*
>
> `adjust-toc=`**`off`**
> Don't adjust the table of contents, which means there won't be any support for thumbtab indexes or mini-tocs. 11, 12, 140, 242, *251*

**`backmatter-sections`**=⟨*setting*⟩ *initial:* `no-number` flowfram v2.0+

Classes that support `\backmatter` tend to suppress the chapter number but not section numbers. As this can cause interference, this option may be used to suppress section numbers as well. §1.2.1; 11, *251*

> `backmatter-sections=`**`no-change`**
> Don't alter secnumdepth in the back matter. 12, *251*

`backmatter-sections`=**`no-number`**
Suppress section numbers in the back matter (`\backmatter` will set secnumdepth to -1). 12, *252*

**`color`**=⟨*boolean*⟩ *default:* `true`; *initial:* `true` flowfram v1.0 (preamble only)
If true, enable colour support. **§1.2.4**; 27, *252*, 255

**`column-changes`**=⟨*setting*⟩ *initial:* `ignore` flowfram v2.0+
Indicates what to do if `\onecolumn` or `\twocolumn` are encountered in the document environment. **§1.2.3**; 22, 179, *252*

`column-changes`=**`clearpage`**
Clear the page but otherwise ignore any instance of `\onecolumn` or `\twocolumn` found in the document environment. 23, *252*

`column-changes`=**`ignore`**
Ignore any instance of `\onecolumn` or `\twocolumn` found in the document environment. 23, *252*

`column-changes`=**`switch`**
Switch to the designated frames. 23, 236, 237, *252*

**`draft`** flowfram v1.0+
Switches on draft mode. **§1.2.4**; 2, 5, 25, 28–31, 179, *252*

**`dynamic-empty-page-style`**=`{`⟨*value*⟩`}` *initial:* `hide` flowfram v2.0+
Controls how the ffempty page style behaves with dynamic header and footer frames. **§1.2.2**; 20, 21, 62, *252*

`dynamic-empty-page-style`=**`empty`**
The header and footer frames have their contents set to empty but will still be shown on the page. 20, 62, *252*

`dynamic-empty-page-style=`**`headings`**
Behaves like ffheadings. 20, *253*

`dynamic-empty-page-style=`**`hide`**
The hide attribute will be switched on with `\pagestyle` and the hidethis attribute will be switched on with `\thispagestyle`. 20, 21, *253*

`dynamic-empty-page-style=`**`ignore`**
The style will do nothing, retaining the current page style. 21, *253*

`dynamic-empty-page-style=`**`myheadings`**
Behaves like ffmyheadings. 21, *253*

`dynamic-empty-page-style=`**`plain`**
Behaves like ffplain. 20, *253*

`dynamic-empty-page-style=`**`show`**
The hide or hidethis attribute won't be switched on by the ffempty style. 20, 21, *253*

**`dynamic-header-case`**=⟨*setting*⟩ *initial:* `no-change` flowfram v2.0+
Indicates whether or not `\chaptermark` (if chapters are defined) or `\sectionmark` (otherwise) with the ffheadings and ffmyheadings styles should change the case of the header text. **§1.2.2**; 18, 61, 62, *253*

`dynamic-header-case=`**`no-change`**
Don't change the case of the header. 18, *253*

`dynamic-header-case=`**`uc`**
Change the header to uppercase. 18, *253*

**`dynamic-page-style-header-font`**`={`⟨*code*⟩`}` *initial:* `\emph`
flowfram v2.0+
Use the given ⟨*code*⟩ to set the font used by `\chaptermark` (if chapters are available) or `\sectionmark` (otherwise) with ffheadings. The ⟨*code*⟩ may be declarations or the final command in ⟨*code*⟩ may be a text-block command. **§1.2.2**; 19, 61, 62, *253*, 254

**dynamic-page-style-subheader-font**={⟨*code*⟩} flowfram v2.0+
Use the given ⟨*code*⟩ to set the font used by `\sectionmark` (if chapters are available) or `\subsectionmark` (otherwise) with ffheadings. Initialised to match `dynamic-page-style-header-font`. **§1.2.2**; 19, 62, *254*

**dynamic-page-style**=⟨*setting*⟩ *initial:* `adjust` flowfram v2.0+ (preamble only)
Indicates whether or not to adjust the standard page styles if the header and footer are converted into dynamic frames. **§1.2.2**; 17, 223, *254*

> `dynamic-page-style=`**adjust**
> If used, `\makedfheaderfooter` will set (`\let`) the empty, plain, headings and myheadings page styles to ffempty, ffplain, ffheadings and ffmyheadings. 18, 20, 65, *254*
>
> `dynamic-page-style=`**noadjust**
> Don't adjust the empty, plain, headings and myheadings page styles. 18, *254*

**dynamic-subheader-case**=⟨*setting*⟩ *initial:* `no-change` flowfram v2.0+
Indicates whether or not `\sectionmark` (if chapters are defined) or `\subsectionmark` (otherwise) with the ffheadings and ffmyheadings styles should change the case of the sub-header text. **§1.2.2**; 18, 62, *254*

> `dynamic-subheader-case=`**no-change**
> Don't change the case of the sub-header. 19, *254*
>
> `dynamic-subheader-case=`**uc**
> Change the sub-header to uppercase. 19, *254*

**final** flowfram v1.0
Switches off draft mode. **§1.2.4**; 25, *254*

**LR** flowfram v1.12+

(preamble only)

The column commands, such as `\twocolumn`, will create flow frames from left to right. **§1.2.3**; 22, 112, 127, 218, *254*

**`matter-thumbtabs`**=⟨*setting*⟩ flowfram v2.0+

Indicates whether or not unstarred sectional units outside of the main matter should have thumbtabs. **§1.2.1**; 10, 13, 139, *255*

> `matter-thumbtabs`=**`all`**
> Support thumbtabs for unstarred units everywhere. 14, *255*
>
> `matter-thumbtabs`=**`main-only`**
> Only have thumbtabs for unstarred units in the main matter. 13, *255*
>
> `matter-thumbtabs`=**`not-back`**
> Only have thumbtabs for unstarred units in the main matter and front matter but not in the back matter. 14, *255*
>
> `matter-thumbtabs`=**`not-front`**
> Only have thumbtabs for unstarred units in the main matter and back matter but not in the front matter. 14, *255*

**`nocolor`** flowfram v1.0
(preamble only)

Equivalent to `color=false`. **§1.2.4**; 27, *255*

**`norotate`** flowfram v1.0

Prevents frame rotation and sets `thumbtab-text=stack`. **§1.2.4**; 26, *255*

**`pages`**=⟨*setting*⟩ *initial:* `relative` flowfram v1.14+
(preamble only)

Indicates which value page lists refer to. **§1.2.4**; 24, 36, 73, 187, *255*

`pages=`**`absolute`**
All page lists refer to the absolutepage counter. 2, 4, 25, 73, 94, *256*

`pages=`**`relative`**
All page lists refer to the page counter. 4, 25, 73, *256*

**`prohibit-frame-rotation`**`=`⟨*boolean*⟩ *default:* `true`; *initial:* `false` flowfram v2.0+
If true, disable frame rotation. **§1.2.4**; 26, *256*

**`RL`** flowfram v1.12+ (preamble only)
The column commands, such as `\twocolumn`, will create flow frames from right to left. **§1.2.3**; 22, 112, 127, 218, *256*

**`rotate`**`=`⟨*boolean*⟩ *default:* `true`; *initial:* `false` flowfram v1.0
If true, enables frame rotation and sets `thumbtab-text=rotate-right`. **§1.2.4**; 26, *256*

**`sections-extra-option`**`=`⟨*setting*⟩ flowfram v2.0+
Indicates whether or not the second optional argument of sectioning commands should be passed to the original or just be used for the thumbtab title. **§1.2.1**; 16, 190, 232, 234, *256*

`sections-extra-option=`**`as-original`**
Indicates that the underlying command has a second optional argument and any second optional argument should simply be passed and not used as the thumbtab title. 17, *256*

`sections-extra-option=`**`original-and-thumbtab`**
Indicates that the underlying command has a second optional argument and any second optional argument should be both passed to the underlying command and used as the thumbtab title (if enabled). 17, *256*

`sections-extra-option`=**`thumbtab-only`**
Indicates that the second optional argument (if present) should only be used as the thumbtab title (if enabled) and not passed to the underlying command. 17, *257*

**`thumbtab-links`**=⟨*setting*⟩ flowfram v2.0+
Indicates which thumbtabs should have `\hyperlinks` (if supported). **§1.2.1**; 14, 141, *257*

`thumbtab-links`=**`all`**
Have `\hyperlinks` in all the thumbtabs. 14, *257*

`thumbtab-links`=**`none`**
Don't have `\hyperlinks` in any of the thumbtabs. 15, *257*

`thumbtab-links`=**`toc-only`**
Only have `\hyperlinks` in the thumbtab indexes. 14, *257*

**`thumbtab-text`**=⟨*setting*⟩ *initial:* `rotate` flowfram v2.0+
Indicates how the text should be shown in the thumbtabs. **§1.2.1**; 15, 141, *257*

`thumbtab-text`=**`normal`**
No rotation or stacking. 15, *257*

`thumbtab-text`=**`rotate-left`**
Rotate the text to the left. 15, *257*

`thumbtab-text`=**`rotate-right`**
Rotate the text to the right. 15, 26, 256, *257*

`thumbtab-text`=**`rotate`**
Turn the text sideways. 15, *257*

`thumbtab-text`=**`stack`**
Stack the letters vertically. 15, 26, 255, *257*

**thumbtabs**=⟨*setting*⟩ flowfram v1.14+ (preamble only)

Indicates whether or not the number or title should be shown in the thumbtabs. **§1.2.1**; 16, 26, 27, *258*

`thumbtabs=`**`both`**
Show both the number (if applicable) and the title. 16, *258*

`thumbtabs=`**`none`**
Don't show either the number or title. 16, *258*

`thumbtabs=`**`number`**
Only show the number (if applicable). 16, *258*

`thumbtabs=`**`title`**
Only show the title. 16, *258*

**toc-thumbtabs**=⟨*setting*⟩ *default:* `true`; *initial:* `false` flowfram v2.0+

Sets whether or not to show the thumbtab index frames in the table of contents (if thumbtabs have been created). **§1.2.1**; 12, *258*

`toc-thumbtabs=`**`aligned`**
Separate the table of contents into blocks that are the same height as the corresponding thumbtab (for single-paged table of contents only). 12, *258*

`toc-thumbtabs=`**`false`**
Don't show the thumbtab indexes in the table of contents. 13, *258*

`toc-thumbtabs=`**`true`**
Show the thumbtab indexes on every page of the table of contents. 13, *258*

`toc-thumbtabs=`**`unaligned`**
Don't align sub-blocks of the table of contents with the thumbtabs and only show the thumbtab indexes on the first page of the table of contents. 12, *258*

**ttbnonum** ☰ flowfram v1.0
(preamble only)
Don't include the number in the thumbtabs (if enabled). **§1.2.4**; 27, *259*, *see* `thumbtabs`

**ttbnotitle** ☰ flowfram v1.0
(preamble only)
Don't include the title in the thumbtabs (if enabled). **§1.2.4**; 26, *259*, *see* `thumbtabs`

**ttbnum** ☰ flowfram v1.0
(preamble only)
Include the number in the thumbtabs (if enabled). **§1.2.4**; 27, *259*, *see* `thumbtabs`

**ttbtitle** ☰ flowfram v1.0
(preamble only)
Include the title in the thumbtabs (if enabled). **§1.2.4**; 26, *259*, *see* `thumbtabs`

**unstarred-thumbtabs**=⟨*boolean*⟩ *default:* `true`; *initial:* `false`
flowfram v2.0+
Indicates whether or not unstarred units should have thumbtabs. **§1.2.1**; 10, 13, *259*

**verbose**=⟨*boolean*⟩ *default:* `true`; *initial:* `false` flowfram v1.14+
Switches on/off verbose mode. **§1.2.4**; 25, *259*

\usepackage[⟨*options*⟩]{**flowframtkutils**}

Supplementary package for use with `flowframtk`. **§8.4**; 78, 153, 155, 156–158, 162, 185, 209–212, 220, 221, *259*, 260

**outline**=⟨*boolean*⟩ *default:* `true`; *initial:* `false` flowframtkutils
If true, provide support for outline text according to the TEX engine. **§8.4.1**; 156, 158, 221, *259*

**path**=⟨*boolean*⟩ *default:* `true`; *initial:* `false` flowframtkutils

provide support for text along a path. **§8.4.1**; 156, *260*

# Index