

# The `longfbox` package

## Breakable boxes with standard CSS attributes.

Daan Leijen  
2015-12-05

### Contents

<b>1. Introduction</b>	<b>1</b>
<b>2. Overview</b>	<b>2</b>
<b>3. Styling with CSS</b>	<b>4</b>
3.1. Borders	5
3.1.1. Border style	5
3.1.2. Border width	7
3.1.3. Border color	7
3.1.4. Border corner radius	9
3.2. Alignment	10
3.3. Background	14
3.4. Padding and margins	15
3.5. Breaking content	16
3.6. Height and width	17
3.7. Predefined styles	17
<b>4. Advanced topics</b>	<b>18</b>
4.1. Hooks and rendering	18
4.2. Internal attributes	20
4.3. Customizing border dashes and dots	20

## 1. Introduction

The `longfbox` package provides framed boxes that can be customized using standard CSS attributes. It was written to support precise rendering of Madoko documents in  $\LaTeX$ . Notable features are:

- Specify the boxes using standard CSS attributes like `border-style=dashed` or `border-top-left-radius=10pt`. Almost all of the CSS 2.0 attributes with regard to borders, background, padding, and margins are supported.
- Fast and portable: only uses the standard  $\LaTeX$  `picture` environment for drawing and does not depend on loading big packages like `tikz` or

`pstricks`, which makes running  $\text{\LaTeX}$  much faster.

- Supports breakable boxes that span multiple columns or pages. This package builds on the `longbox` package to break boxes over multiple pages.
- Much care has been put in precise rendering with proper baseline alignment and no spurious extra whitespace.
- In contrast to larger packages like `mdframed` or `tcolorbox`, the `longbox` package does not have more extensive features like frame titles, middle lines, skins, etc. The focus of this package is on rendering boxes well with full CSS support where every corner and side can be styled separately with good looking transitions, where we only depend on the standard `picture` environment.

## 2. Overview

There are two ways to create a framed box, the environment `longfbox` and the command `\lfbbox`.

`\lfbbox[\langle options \rangle]{\langle content \rangle}`

The `\langle options \rangle` are optional and specify CSS attributes. Just like the regular `\fbox` command, the `\lfbbox` command sets the content in a horizontal box and cannot break the content over multiple lines (but it can contain other boxes like a `\parbox` or `minipage` environment.). The default width of an `\lfbbox` is the natural width of its content. This corresponds to a CSS inline element.

Here is an `\lfbbox{inline}` box, just like an `\fbox{fbox}`.

Here is an `inline` box, just like an `fbox`.

`\begin{longfbox}[\langle options \rangle]{\langle content \rangle}\end{longfbox}`

The `longfbox` environment sets the content in a long vertical box and can break content over multiple columns or pages. The default width is the current `\linewidth`. This corresponds to a CSS block element.

```
\begin{longfbox}
The \textsf{longfbox} can contain much
longer content and will by default be
as wide as the current line width.
\end{longfbox}
```

```
The longfbox can contain much longer content and will by default be as
wide as the current line width.
```

```
\newfboxstyle{<name>}{<options>}
```

Defines a new style that can be used to specify commonly used options. For example, the package defines:

```
\newfboxstyle{tight}{padding=0pt,margin=0pt,baseline-skip=false}
```

The new style `tight` can now be used to render a tight box:

```
Here is a \lfbox[tight]{tight} box.
```

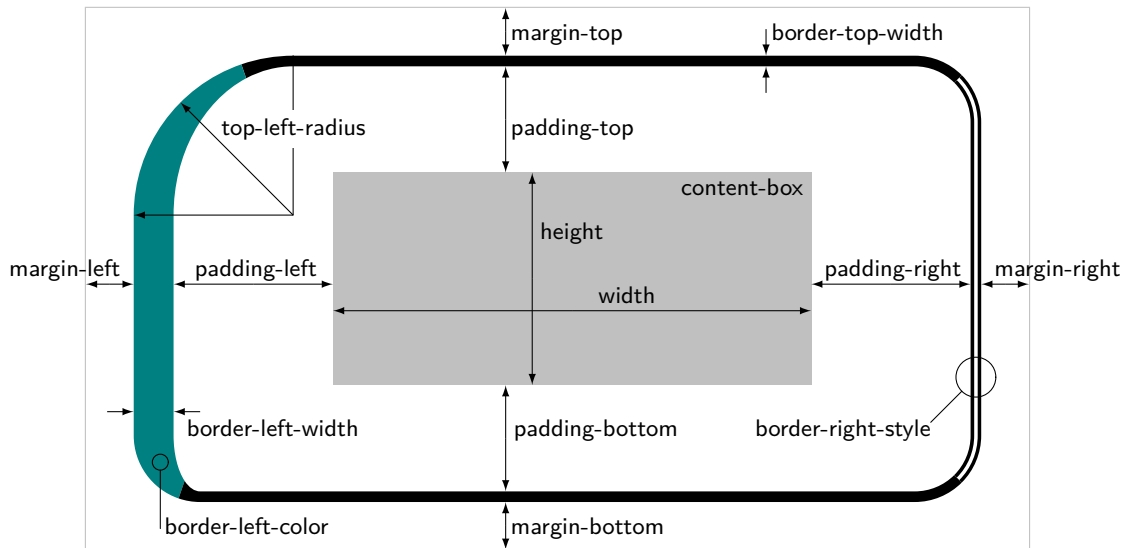
```
Here is a tight box.
```

The `\lfbox` and `longfbox` environment can replace many commands in L<sup>A</sup>T<sub>E</sub>X through the rich CSS interface. In particular:

- `framed`, `\framebox`, `\fbox`: through the `text-align` and `width` attributes.
- `minipage`, `\parbox`: through the `width`, `text-align`, `vertical-align` and `baseline` attributes.
- `\makebox`: through `border-style=none` and the `text-align` and `width` attributes.
- `\raisebox`: through the `raise` attribute.
- `\colorbox`, `\fcolorbox`: through the `background-color` attribute.
- `\doublebox`, `\ovalbox`: through the `double` border style, and the `border-radius` attribute.
- `\shadowbox`: soon :-)

```
\fboxset{<options>}
```

Options can be set for the current scope through this declaration. For example, to make all borders rounded and red by default, use:



**Figure 1.** Attributes of the `longfbox` are modeled after the corresponding CSS attributes. Borders are drawn using the standard `picture` environment.

```
\fboxset{rounded,border-color=red}%
Here is a \lfbox{rounded} box.
Here is a rounded box.
```

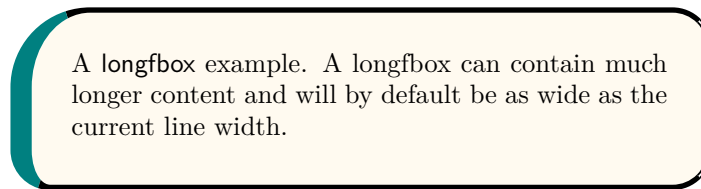
### 3. Styling with CSS

In the options you can specify standard CSS attributes to style your boxes. Figure 1 shows the basic CSS attributes that determine how a box is rendered. If you have used CSS boxes before specifying a box in  $\text{\LaTeX}$  is straightforward. Here is an example with an `\lfbox`:

```
Here is a
\lfbox[
  border-width=0.8pt,
  border-left-color=red,
  border-style=dotted,
  padding={0.2ex,0.4ex}, %top&bottom, right&left
]{fancy} box.
Here is a fancy box.
```

Here is another example using the `longfbox` environment:

```
\begin{longfbox}[
  margin-right=6em,
  padding=1.5em,
  background-color=floralwhite,
  background-clip=padding-box,
  border-width=2pt,
  border-radius=15pt,
  border-top-left-radius=30pt,
  border-left-width=8pt,
  border-left-color=teal,
  border-right-style=double,
]
A \textsf{longfbox} example. A longfbox can contain much
longer content and will by default be as
wide as the current line width.
\end{longfbox}
```



### 3.1. Borders

The border options determine how the frame of the box is rendered.

#### 3.1.1. Border style

Each border can have an individual style:

```
border-style=<style sides>
border-top-style=<style> (=solid)
border-right-style=<style> (=solid)
border-bottom-style=<style> (=solid)
border-left-style=<style> (=solid)
```

where all CSS styles are supported:

*<style>*= none | hidden | solid | dotted | dashed | double

| inset | outset | groove | ridge

The *<style sides>* value can take 1 to 4 style arguments just like in CSS:

```
<attr sides>= <attr>  
| {<top-bottom attr> , <left-right attr>}  
| {<top-attr>, <left-right attr>, <bottom-attr>}  
| {<top-attr>, <right-attr>, <bottom-attr>, <left-attr>}
```

Here are some examples of each CSS style:

```
\lfbx[border-style=solid]{solid},  
\lfbx[border-style=dashed]{dashed},  
\lfbx[border-style=dotted]{dotted},  
\lfbx[border-style=double,border-width=2pt]{double},  
\lfbx[border-style={solid,none,dashed,none}]{various}.
```

solid, dashed, dotted, double, various.

Note, for a dotted border, it is often nicer to use the `dotted` style (Section 3.7) since it makes the dots a bit larger, e.g.

```
Compare \lfbx[dotted]{dotted} versus \lfbx[border-style=dotted]{dotted}.
```

```
Compare dotted versus dotted.
```

The final CSS styles darken sides of the border to give a 3D effect:

```
\lfbx[border-style=inset,border-color=red,border-width=3pt]{inset},  
\lfbx[border-style=outset,border-color=red,border-width=3pt]{outset},  
\lfbx[border-style=groove,border-color=teal,border-width=4pt]{\strut groove},  
\lfbx[border-style=ridge,border-color=teal,border-width=4pt]{\strut ridge}.
```

inset, outset, groove, ridge.

The darkness can be controlled using the `border-dark-mix` attribute (Section 3.1.3).

Beyond CSS, there are also styles for the border top and bottom when a box is broken over multiple pages (see Section 3.5):

```
border-break-style=<style break-sides>  
border-break-top-style=<style> (=none)
```

`border-break-bottom-style=<style>` (=none)

where `<style break-sides>` takes one or two arguments:



`<attr break-sides>=<attr>`  
| {`<break-top-attr>`, `<break-bottom-attr>`}

See Figure 2 in Section 3.5 for more information.

### 3.1.2. Border width

`border-width=<width sides>`  
`border-top-width=<dimen>` (=fboxrule)  
`border-right-width=<dimen>` (=fboxrule)  
`border-bottom-width=<dimen>` (=fboxrule)  
`border-left-width=<dimen>` (=fboxrule)

This sets the width of each border. By default the `\fboxrule` width is used which is normally **0.4pt**.

A `\lfbbox[border-width=3pt, border-left-color=red]{thick}` border,  
and  
`\lfbbox[border-top-width=0pt, border-bottom-width=1pt]{varied}`.  
A  border, and .

`border-break-width=<width break-sides>`  
`border-break-top-width=<dimen>` (=0pt)  
`border-break-bottom-width=<dimen>` (=0pt)

These specify border width around page breaks. See Figure 2 in Section 3.5 for more information.

### 3.1.3. Border color

`border-color=<color sides>`  
`border-top-color=<color>` (=black)

```
border-right-color=<color> (=black)
border-bottom-color=<color> (=black)
border-left-color=<color> (=black)
```

Sets the color of each border.

```
<color>= <color name>
| <xcolor spec>
| \#RRGGBB
| {}
```

Colors can be specified by a name (e.g. `red`), and `xcolor` package color specification (e.g. `red!60`), a direct HTML color (e.g. `\#800080`) or as an empty value. The empty value is used for transparency on backgrounds.

```
Strange
\lfbx[border-width=3pt,
border-top-color=red!50,
border-bottom-color=\#800080,
]{colors}.

\noindent\begin{longfbx}[border-style=none,
border-left-style=solid,
border-left-width=5pt,
border-color=blue,
padding-left=1ex,
]This is a definition
\end{longfbx}

Strange .
 This is a definition
```

Besides the regular CSS attributes, colors for borders around a page break are set as:

```
border-break-color=<color break-sides>
border-break-top-color=<color> (=black)
border-break-bottom-color=<color> (=black)
```

See Section 3.5 for further information.



`border-dark-mix=<color mix> (!70!black)`

The `border-dark-mix` is used to make a color darker and is used for the `inset`, `outset`, `groove` and `ridge` styles. This value is basically appended to the main color of the border. The default takes 70% of the main color and mixes in 30% black.

#### 3.1.4. Border corner radius

```
border-radius=<radius corners>
border-top-left-radius=<radius> (=0pt)
border-top-right-radius=<radius> (=0pt)
border-bottom-left-radius=<radius> (=0pt)
border-bottom-right-radius=<radius> (=0pt)
```

The above attributes specify the corner radius to enable rounded corners.

`<radius>=<dimen> | {<x-radius>,<y-radius>}`

A radius value is either a dimension, or a separate x- and y-radius for elliptical borders. Here are some examples:

```
\lfbbox[border-radius=0.5ex]{rounded},
\lfbbox[border-radius=100ex, background-color=floralwhite]{\strut elliptical},
\lfbbox[border-radius=0.5em,height=1em,width=1em,tight,
text-align=center,height-align=middle]{1}.
```

`rounded`, `elliptical`, ①.

Here is an example of a `\longfbox` with an elliptical corner:

```

\begin{longfbox}[
  border-radius=15pt,
  border-width=2pt,
  border-top-left-radius={50pt,25pt},
  border-right-style=dashed,
  border-left-style=double,
  border-left-color=teal,
  padding={1em,15pt},
  width=0.6\linewidth,
]
\hobbit[7]
\end{longfbox}

```

There is little or no magic about them, except the ordinary everyday sort which helps them to disappear quietly and quickly when large stupid folk like you and me come blundering along, making a noise like elephants which they can hear a mile off.

### 3.2. Alignment

Much care has been taken to ensure proper alignment and preservation of the baseline. In our examples, we use the following definition where we enable `show-markers` so we can see the baseline and dimensions of the longbox:

```

\newcommand\alignbox[1]{%
  \begin{longfbox}[width=2em,height=4.25em,show-markers,
    baseline-skip=false,#1]
    foo,\ \ bar,\ \ gnu.
  \end{longfbox}%
}

```

`baseline=` bottom | middle | top

The `baseline` attribute is not a CSS attribute, but we can use it to control the baseline of the content of a box. For example:

```
Aligning \alignbox{baseline=bottom},
\alignbox{baseline=middle},
\alignbox{baseline=top} done.
```

The diagram shows the word "Aligning" followed by three boxes, each containing the text "foo, bar, gnu.". The first box is aligned to the bottom of the text "done.". The second box is aligned to the middle of the text "done.". The third box is aligned to the top of the text "done.". The word "done." is positioned to the right of the third box.

The vertical-align property aligns a box:

```
vertical-align= baseline | bottom | middle | top
| text-bottom | text-top | super | sub
```

The default `baseline` attribute aligns the baseline of the box with the baseline of the text. The `bottom` attribute aligns the bottom of the box with the bottom of the text, `middle` aligns the middle of the box with the middle of the text, and `top` aligns the top of the box with the top of the text.

Here are the various attributes in action (compare this to the examples for different baselines):

```
Aligning \alignbox{vertical-align=bottom},
\alignbox{vertical-align=middle},
\alignbox{vertical-align=top} done.
```

The diagram shows the word "Aligning" followed by three boxes, each containing the text "foo, bar, gnu.". The first box is vertically aligned to the bottom of the text "done.". The second box is vertically aligned to the middle of the text "done.". The third box is vertically aligned to the top of the text "done.". The word "done." is positioned to the right of the third box.

Finally, the `super` and `sub` attributes align the baseline of the box with the baseline of super- and sub-scripts:

```
$x^x_x$ \alignbox{vertical-align=super}, \alignbox{vertical-align=sub} $x^x_x$.
```

Unfortunately, in  $\text{\LaTeX}$  we cannot determine the height or depth of the current text line easily so these values are fixed at  $0.7\text{\baselineskip}$  and  $0.3\text{\baselineskip}$ . The `text-bottom` and `text-top` are equal to `bottom` and `top` for that reason.

`raise=<dimen>`

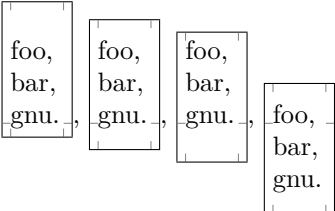
In CSS it is also possible to assign a length to the `vertical-align` attribute. In  $\text{\LaTeX}$  we use the `raise` attribute instead which raises an aligned box by the assigned value.

```
Aligning \alignbox{raise=1em}, \alignbox{raise=-1em}, done.
```

`height-align= top | middle | bottom`

This is another non-CSS attribute (but probably one of the most desired :-)): it specifies the vertical alignment of the content inside the box. This attribute only has an effect for boxes where the `height` is specified. This attribute keeps the baseline unchanged and does not influence the vertical alignment directly.

Aligning `\alignbox{height-align=bottom},`  
`\alignbox{height-align=middle},`  
`\alignbox{height-align=top},`  
`\alignbox{height-align=middle,baseline=top},`  
done.

Aligning  done.

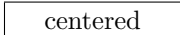
`text-align= default | left | center | right | justify`

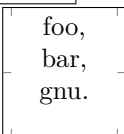
Specifies the alignment of the content horizontally. The default is left for a horizontal `\lfbbox` and justify for a `longfbox` environment.

Here is `\lfbbox[width=6em,text-align=center]{centered}` text.

And vertical


`\alignbox{text-align=center,baseline=middle,width=4em}` too.

Here is  text.

And vertical  too.

Just like with `\makebox` we can use a narrow width to let the text overlay to the left or right:

`\lfbbox[width=1.7em,text-align=right]{bummer, too long}.`




bummer, too .

### 3.3. Background

`background-color=<color> (={})`  
`background-clip=border-box | padding-box | content-box`

Specifies the background color. The `background-clip` attribute determines if the background color spans all the way out to the border (default), just the padding box, or only the content box itself.


```
\newcommand\bgbox[2]{%
  \lfbbox[rounded,border-style=double,border-width=3pt,
    border-color=teal,
    background-color=#1,background-clip=#2]%
}
a\rlap{b}\bgbox{teal!30}{border-box}{foo},
a\rlap{b}\bgbox{teal!30}{padding-box}{foo},
a\rlap{b}\bgbox{teal!30}{content-box}{foo}.
```

a , a , a .

`background-padding-color=<color> (= \option{background-color})`  
`background-border-color=<color> (= \option{background-color})`  
`background-content-color=<color> (= \option{background-color})`

Going beyond CSS, we can also specify the colors for the background at the border and padding separately.

```
A \lfbbox[rounded,
  background-color=teal!30,
  background-padding-color=floralwhite,
  background-border-color=gray!50,
  border-style=double,border-width=3pt
]{too colorful} box.
```

A  box.

### 3.4. Padding and margins

```
padding=<padding sides>
padding-top=<dimen> (=\\fboxsep)
padding-right=<dimen> (=\\fboxsep)
padding-bottom=<dimen> (=\\fboxsep)
padding-left=<dimen> (=\\fboxsep)
```

The padding is the space between the content and the border of a box. By default the padding is equal to the `\\fboxsep` value.

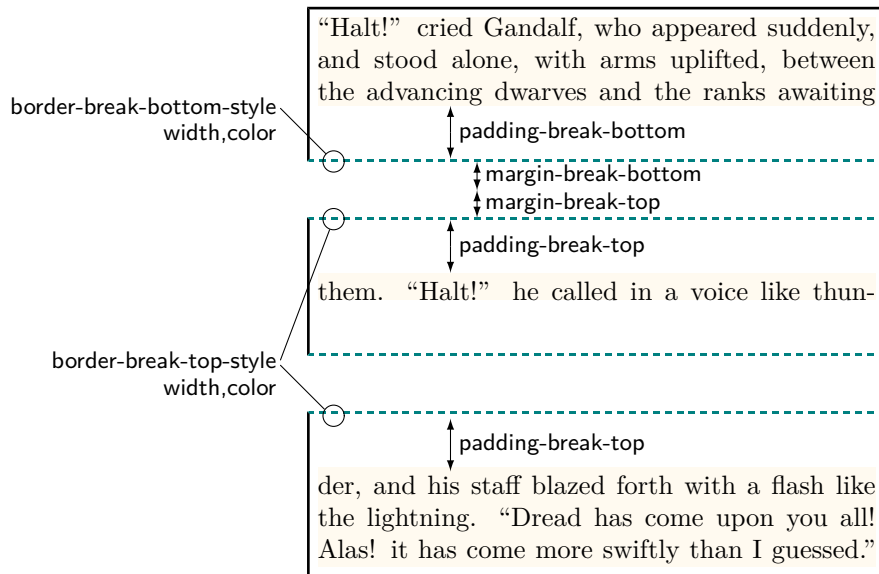
```
margin=<margin sides>
margin-top=<dimen> (=0pt)
margin-right=<dimen> (=0pt)
margin-bottom=<dimen> (=0pt)
margin-left=<dimen> (=0pt)
```

The margin is the transparent area outside the borders. In contrast to the CSS attribute, the margins do *not* overlap. Please use the standard `\\addvspace` command for adding overlapping top- and bottom-margins.

```
\\begin{longfbox}[width=6.8em]
Here is a longer paragraph with a
\\lfbbox[margin={1em,0.5em,1em},padding=1ex,show-markers]{box}
inside with margins.
\\end{longfbox}
```

Here is a longer  
paragraph with  
a box inside  
with margins.

```
padding-break-top=<dimen> (=0.5em)
padding-break-bottom=<dimen> (=0.5em)
margin-break-top=<dimen> (=0pt)
margin-break-bottom=<dimen> (=0pt)
```



**Figure 2.** A box that has been broken in three parts. The border style and the padding and margin of each break can be set separately. Use the `breakable` option to make boxes breakable over page boundaries. In the above example we also used the `breakat={75pt}` option to break at specific heights, and the `background-clip=content-box` to limit the background color to the content only.

These attributes determine the margin and padding around a page break. By default, there is some padding set around a break such that the vertical borders of a box ‘stick out’ a bit giving the reader a hint that the box continues at the next page (see Figure 2)

### 3.5. Breaking content

`breakable=⟨bool⟩` (=false)

When the option `breakable` is present, a `longfbox` environment can be broken at page boundaries. All the borders, margins, padding, etc. are preserved over the page break. Moreover, the border, padding, and margin for each break at the top and bottom can be set separately, as shown in in Figure 2.

`extra-split=⟨dimen⟩` (=1.5\baselineskip)

(Advanced) This option determines the minimal height of a box that is broken.



If the height would be less the box is not broken and moved to the next page. This prevents for example to have just the top border and padding on one page followed by content on the next page.

`breakat={⟨dimen1⟩, . . . , ⟨dimenn⟩}`

(Advanced) This option lets you break a box manually at specific heights. The `breakable` option must be set for this to work. Usually a box is broken to fit the space available on the page, but when `breakat` is given, the box breaks at `⟨dimen1⟩`. After the break, the first dimension is removed from the list and the box is broken again at `⟨dimen2⟩` until only one element (`⟨dimenn⟩`) is left. After that the box is repeatedly broken at `⟨dimenn⟩` until all content is processed. For example, in Figure 2 we used `breakat={75pt}` to break the box at three `75pt` heights. Note that when a box breaks, it chooses the largest height that is less than `75pt` where the content can be broken nicely, i.e. a box never breaks in the middle of a line for example.

The `breakat` can also be used to break a box in a `multicols` environment.

### 3.6. Height and width

`height=⟨dimen⟩`  
`width=⟨dimen⟩`

Specifies the `height` or `width` of the inner content box, see Figure 1. By default, the width of horizontal `\lfbox` is the natural width of its content. In contrast, the default width of a vertical `longfbox` environment is the current `\linewidth`. The default `height` is always the natural height of the content. When the `height` is fixed, you can use `height-align` to align the content in the available area.

`outer-height=⟨dimen⟩`  
`outer-width=⟨dimen⟩`

For convenience, there are also these non-CSS attributes that let you specify the width and height of the entire box including the padding, border, and margins. Internally translated to the `height` or `width` before processing the box.

### 3.7. Predefined styles

`tight={padding=0pt,margin=0pt,padding-break=0pt}`  
`rounded={border-radius=1ex}`

```
dotted={border-width=0.8pt,border-style=dotted}
```

Some predefined convenient styles. `tight` puts the box tightly around the content. The `rounded` style uses a border radius that looks nice relative to the font size, and `dotted` uses a slightly larger border width which looks better with dots.

```
A \lfbbox[tight]{tight}, \lfbbox[dotted]{dotted},  
and \lfbbox[rounded]{rounded} box.  
A tight, dotted, and rounded box.
```

## 4. Advanced topics

### 4.1. Hooks and rendering

```
render=default | plain | picture
```

There are two ways to render a box border: using just plain  $\LaTeX$  and the  $\LaTeX$  `picture` environment. The plain renderer just uses `\hrule` etc. and can only render simple boxes with `solid` or `dashed` styles without rounded corners. The `default` will pick the `plain` renderer if a box is simple, and use the `picture` renderer otherwise. This is mostly to make the rendering more efficient but there should be no other difference.

```
insert-before=<value>  
insert-after=<value>
```

These are hooks to insert content just before or after the content box.

```
A \lfbbox[insert-before={"\bgroup\itshape},  
insert-after={\egroup}]{quoted}  
box.  
A "quoted" box.
```

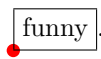
```
render-insert-before=<value>  
render-insert-after=<value>
```

These are hooks to insert content just before or after the border-box is rendered.

```

\newcommand\makecircle{%
  \setlength\unitlength{1pt}%
  \begin{picture}(0,0)% make it occupy no space
    \color{red}%
    \put(0,0){\circle*{5}}%
  \end{picture}%
}
\lfbbox[render-insert-before=\makecircle]{funny}.

```



```

picture-insert-before=<value>
picture-insert-after=<value>

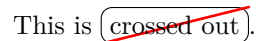
```

These insert content just before or after rendering the frame in a `picture` environment. In both case, the origin is at the bottom-right corner. Here is an example where we cross-out a box:

```

\newcommand\crossout{%
  \color{red}\linethickness{1pt}\roundcap%
  \moveto(0,0)
  \lineto(\optionunit{/fbox/@border-box-width},
    \optionunit{/fbox/@border-box-height})
  \strokepath
}
This is
\lfbbox[rounded,picture-insert-after=\crossout]{crossed out}.

```



```

plain-side-insert-before=<value>
plain-side-insert-after=<value>
picture-side-insert-before=<value>
picture-side-insert-after=<value>

```

These attributes insert content just before rendering a border side.

## 4.2. Internal attributes

`baseline-skip`= $\langle bool \rangle$  (`=true`)

Usually, the library adds a small vertical skip in front of a vertical `longfbox` such that its content aligns to a `\baselineskip`. Similarly, it adds such skip just after the box. This generally makes text align better, especially on a double column layout. Setting it to `false` disables such vertical spacing.

`show-markers`= $\langle bool \rangle$   
`marker-color`= $\langle color \rangle$  (`=gray`)  
`marker-width`= $\langle dimen \rangle$  (`=0.1pt`)

If `show-markers` is `true` it will show the padding, baseline, margin and border markers using `marker-color` and `marker-width`.

`eject`= $\langle value \rangle$  (`={\protect\vfill\protect\eject}`)

After breaking a box, this command is issued which normally ends the current page. The `breakat` command sets this to empty so no page break occurs in that case.

`debug`= $\langle bool \rangle$  (`=false`)  
`verbose`= $\langle bool \rangle$  (`=false`)

Enable debug and verbose log messages.

## 4.3. Customizing border dashes and dots

The following attributes are used to render dashes and dots for the dashed and dotted border styles.

`border-dash`= $\langle dash sides \rangle$   
`border-top-dash`= $\langle dimen \rangle$  (`=0.7ex`)  
`border-right-dash`= $\langle dimen \rangle$  (`=0.7ex`)  
`border-bottom-dash`= $\langle dimen \rangle$  (`=0.7ex`)  
`border-left-dash`= $\langle dimen \rangle$  (`=0.7ex`)  
`border-break-top-dash`= $\langle dimen \rangle$  (`=\option{border-top-dash}`)  
`border-break-bottom-dash`= $\langle dimen \rangle$  (`=\option{border-bottom-dash}`)

```

border-dashskip=(dashskip sides)
border-top-dashskip=(dimen) (=0.7\option{border-top-dash})
border-right-dashskip=(dimen) (=0.7\option{border-right-dash})
border-bottom-dashskip=(dimen) (=0.7\option{border-bottom-dash})
border-left-dashskip=(dimen) (=0.7\option{border-left-dash})
border-break-top-dashskip=(dimen) (= \option{border-top-dashskip})
border-break-bottom-dashskip=(dimen) (= \option{border-bottom-dashskip})

```

```

border-dotskip=(dotskip sides)
border-top-dotskip=(dimen) (=0.7\option{border-top-width})
border-right-dotskip=(dimen) (=0.7\option{border-right-width})
border-bottom-dotskip=(dimen) (=0.7\option{border-bottom-width})
border-left-dotskip=(dimen) (=0.7\option{border-left-width})
border-break-top-dotskip=(dimen) (= \option{border-top-dotskip})
border-break-bottom-dotskip=(dimen) (= \option{border-bottom-dotskip})

```

Created with [Madoko.net](https://www.madoko.net/).