Typesetting captions with the caption package*

Axel Sommerfeldt
caption@sommerfeldt.net

2005/06/13

Abstract

The caption package provides many ways to customise the captions in floating environments such `figure` and `table` and cooperates with many other packages.[1]

1 Introduction

Within the standard LaTeX classes captions haven't received the attention they deserve. Simply typeset as an ordinary paragraph there is no remarkable visual difference from the rest of the text, like here:

Figure 1: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

There should be possibilities to change this; e.g., it would be nice if you can make the text of the caption a little bit smaller as the normal text, add an extra margin, typeset the caption label with the same font family and shape as your headings etc. Just like this one:

Figure 2: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

With this package you can do this easily as there are many ready-to-use caption formatting options, but you are free to define your very own stuff, too.

*This package has version number v3.0g, last revised 2005/06/28.

[1]A complete re-work of the user interface done together with Steven D. Cochran and Frank Mittelbach has lead to this new enhanced version 3.0.

2 Using the package

`\usepackage` Insert

```
\usepackage[⟨options⟩]{caption}[2005/06/28]
```

into the preamble of your document, i.e. the part of your document between `\documentclass` and `\begin{document}`. The options control how your captions will look like; e.g.,

```
\usepackage[margin=10pt,font=small,labelfont=bf]{caption}
```

would result in captions looking like the second one in the introduction.

`\captionsetup` For a later change of options the caption package provides the command

```
\captionsetup[⟨float type⟩]{⟨options⟩}
```

So

```
\usepackage[margin=10pt,font=small,labelfont=bf]{caption}
```

and

```
\usepackage{caption}
\captionsetup{margin=10pt,font=small,labelfont=bf}
```

are equal in their results.

It's good to know that `\captionsetup` has an effect on the current environment only. So if you want to change some settings for the current `figure` or `table` only, just place the `\captionsetup` command inside the `figure` or `table` right before the `\caption` command. For example

```
\begin{figure}
  ...
  \captionsetup{singlelinecheck=off}
  \caption{...}
\end{figure}
```

switches the single-line-check off, but only for this `figure` so all the other captions remain untouched.

(For a description of the optional parameter ⟨*float type*⟩ see section 4: *"Useful stuff"*.)

3 Options

3.1 Formatting

`format=` A figure or table caption mainly consits of three parts: the caption label, which says if this object is a 'Figure' or 'Table' and what number is associated with it, the caption text itself, which is normally a short description of contents, and the caption separator which separates the text from the label.

The *caption format* determines how this information will be presented; it is specified with the option `format=`⟨*format name*⟩ ,

having the name of the caption format as its argument.
There are two standard caption formats:[2]

`default`	Typesets the captions as a normal paragraph. (This is the default behaviour, it is adapted from the standard LaTeX document classes.)
`hang`	Indents the caption text, so it will 'hang' under the first line of the text.

An example: Specifying the option

```
format=hang
```

yields captions like this:

Figure 3: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

`indention=` For both formats (`default` and `hang`) you can setup an extra indention starting at the second line of the caption. You do this with the option

`indention=`⟨*amount*⟩.

Three examples:

```
format=default,indention=.5cm
```

Figure 4: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

```
format=hang,indention=-0.5cm
```

Figure 5: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

`labelformat=` With the option

`labelformat=`⟨*label format name*⟩

New description v3.0e

you specify how the caption label will be typeset. There are three standard caption label formats:

`default`	The caption label will be typeset as specified by the document class, usually this means the name and the number (like `simple`). (This is the default behaviour.)

[2]You have the option to define your own ones, too. See section 5: *"Do it yourself!"* for details.

`empty`	The caption label will be empty. This option only makes sense when used together with other options like `labelsep=none`.
`simple`	The caption label will be typeset as a name and a number.
`parens`	The number of the caption label will be typeset in parentheses.

An example: Using the options

```
labelformat=parens,labelsep=quad
```

yields captions like this one:

Figure (6)   White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

`labelsep=` With the options

`labelsep=`⟨*label separator name*⟩

you specify what caption separator will be used. You can choose one of the following:

`none`	There is no caption separator. This option only makes sense when used together with other options like `labelformat=empty`.
`colon`	The caption label and text will be separated by a colon and a space. (This is the default one.)
`period`	The caption label and text will be separated by a period and a space.
`space`	The caption label and text will be separated by a single space.
`quad`	The caption label and text will be separated by a `\quad`.
`newline`	The caption label and text will be separated by a line break (`\\`).

Two examples:

```
labelsep=period
```

Figure 7. White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

```
labelsep=newline,singlelinecheck=false
```

Figure 8
White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

3.2 Justification

`justification=` As addition to the caption format you could also specify a *caption justification*; it is specified with the option

`justification=`⟨*justification name*⟩ .

You can choose one of the following:

`justified`	Typesets the caption as a normal paragraph. (This is the default.)
`centering`	Each line of the caption will be centered.
`centerlast`	The last line of each paragraph of the caption text will be centered.
`centerfirst`	Only the first line of the caption will be centered.
`raggedright`	Each line of the caption will be moved to the left margin.
`RaggedRight`	Each line of the caption will be moved to the left margin, too. But this time the command `\RaggedRight` of the ragged2e package will be used to achieve this. This difference is that this time the word breaking algorithm of TeX will work inside the caption.
`raggedleft`	Each line of the caption will be moved to the right margin.

Two examples:

```
justification=centerlast
```

Figure 9: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

```
format=hang,justification=raggedright
```

Figure 10: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

```
labelsep=newline,justification=centering
```

Figure 11
White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

`singlelinecheck=` If the caption fit in a single line it will always be centered, ignoring the justification you set:

Figure 12: A short caption.

This behaviour is adapted from the standard LaTeX document classes `article`, `report`, and `book`), but using the caption package you can switch this special treatment of such short captions off with the option

```
singlelinecheck=⟨bool⟩ .
```

Using `false`, `no`, `off` or `0` for ⟨*bool*⟩ you switch off the extra centering:

```
singlelinecheck=false
```

Doing so the above short caption would look like

Figure 12: A short caption.

Using `true`, `yes`, `on` or `1` for ⟨*bool*⟩ you switch on the extra centering again. (The default is on.)

3.3 Fonts

`font=`
`labelfont=`
`textfont=`

There are three font options which affects different parts of the caption: One affecting the whole caption (`font`), one which only affects the caption label and separator (`labelfont`) and at last one which only affects the caption text (`testfont`). You set them up using the options

```
     font={⟨font options⟩}  ,
labelfont={⟨font options⟩}  and
 textfont={⟨font options⟩}  .
```

And these are the available font options:

`scriptsize`	Very small size
`footnotesize`	The size usually used for footnotes
`small`	Small size
`normalsize`	Normal size
`large`	Large size
`Large`	Even larger size
`up`	Upright shape
`it`	*Italic shape*
`sl`	*Slanted shape*
`sc`	SMALL CAPS SHAPE
`md`	Medium series
`bf`	**Bold series**

`rm`	Roman family
`sf`	Sans Serif family
`tt`	`Typewriter family`

If you use only one of these options you can omit the braces; e.g., the options `font={small}` and `font=small` yield the same result.

Two examples:

```
font={small,it},labelfont=bf
```

Figure 13: *White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.*

```
font=small,labelfont=bf,textfont=it
```

Figure 14: *White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.*

3.4 Margins and further paragraph options

`margin=` `width=` For all captions you can specify *either* an extra margin *or* a fixed width. You do this using the options

`margin=`⟨*amount*⟩ *or*
`width=`⟨*amount*⟩

Nevertheless what option you use, the left and right margin will be the same.

Two examples illustrating this:

```
margin=10pt
```

Figure 15: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

```
width=.75\textwidth
```

Figure 16: White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

`parskip=` This option is useful for captions containing more than one paragraph. If specifies the extra vertical space inserted between them:

`parskip=`⟨*amount*⟩

One example:

```
margin=10pt,parskip=5pt
```

Figure 17: First paragraph of the caption. This one contains some test, just to show how these options affect the layout of the caption.

Second paragraph of the caption. This one contains some text, too, to show how these options affect the layout of the caption.

`hangindent=` The option

hangindent=⟨*amount*⟩

is for setting up a hanging indention starting from the second line of each paragraph. If the caption contains just a single paragraph, using this option leads to the same result as the option `indention=` you already know about. But if the caption contains multiple paragraphs you will notice the difference:

```
format=hang,indention=-.5cm
```

Figure 18: First paragraph of the caption. This one contains some test, just to show how these options affect the layout of the caption.
Second paragraph of the caption. This one contains some text, too, to show how these options affect the layout of the caption.

```
format=hang,hangindent=-.5cm
```

Figure 19: First paragraph of the caption. This one contains some test, just to show how these options affect the layout of the caption.
Second paragraph of the caption. This one contains some text, too, to show how these options affect the layout of the caption.

3.5 Styles

`style=` A suitable combination of caption options is called *caption style*. You can compare them more or less to page styles which you set up with `\pagestyle`: The caption style provides all settings for a whole caption layout.

You switch to an already defined caption style with the option

style=⟨*style name*⟩ .

The caption package usually defines only the style `default` which puts all options you already know about to the default ones. This means that specifying the option

```
style=default
```

has the same effect as specifying all these options:

```
format=default,labelformat=simple,labelsep=colon,
justification=justified,font=default,labelfont=default,
textfont=default,margin=0pt,indention=0pt,parindent=0pt
hangindent=0pt,singlelinecheck=true
```

3.6 Skips

`aboveskip=` `belowskip=` The spaces above and below the caption are controlled by the skips `\abovecaptionskip` and `\belowcaptionskip`. The standard LaTeX document classes `article`, `report` and `book` set `\abovecaptionskip` to `10pt` and `\belowcaptionskip` to `0pt`.

Both skips can be changed with the command `\setlength`, but you can use these options, too:

> `aboveskip=`⟨*amount*⟩ and
> `belowskip=`⟨*amount*⟩ .

`position=` Using `\abovecaptionskip` and `\belowcaptionskip` has a major design flaw: If the caption is typeset *above* (and not *below*) the figure or table they are not set up very useful at default, because there will be some extra space above the caption but no space between the caption and the figure or table itself. (Remember: `\belowcaptionskip` is usually set to `0pt`.)

Please compare the spacing in these small tables:

Table 1: A table

A	B
C	D

A	B
C	D

Table 2: A table

But you can fix this by using the option `position=`: It specifies how the spacing above and below the caption will be used:

> `position=top` (or `position=above`)

tells the caption package to use the spacing useful for caption *above* the figure or table and

> `position=bottom` (or `position=below`)

tells the caption package to use the spacing useful for captions *below* the figure or table. (The last one is the default setting except for `longtable`s.)

So adding an extra `\captionsetup{position=top}` to the left example table gives you proper spacing around both captions:

Table 3: A table

A	B
C	D

A	B
C	D

Table 4: A table

(Technically speaking `\abovecaptionskip` and `\belowcaptionskip` will be swapped if you specify the option `position=top`, so in both cases `\abovecaptionskip` will be used between the caption and the figure or table itself.)

`tableposition=` This option is especially useful when used together with the optional argument of the `\captionsetup` command. (See section 4: *"Useful stuff"* for details.) E.g.,

```
\captionsetup[table]{position=top}
```

New feature v3.0a causes all captions within tables to be treated as captions *above* the table (regarding spacing around it). Because this is a very common setting the caption package offers an abbreviating option for the use with `\usepackage`:

```
\usepackage[...,tableposition=top]{caption}
```

is equivalent to

```
\usepackage[...]{caption}
\captionsetup[table]{position=top}
```

4 Useful stuff

\caption The command

```
\caption[⟨lst_entry⟩]{⟨heading⟩}
```

typesets the caption inside a floating environment like `figure` or `table`. Well, you already know this, but what is new is the fact then when you leave the argument ⟨*lst_entry*⟩ empty, no entry in the list of figures or tables will be made; e.g.,

```
\caption[]{A figure without entry in the list of figures.}
```

\caption* The longtable package defines the command `\caption*` which typesets the caption without label and without entry in the list of tables. An example:

```
\begin{longtable}{cc}
  \caption*{A table}\\
  A & B \\
  C & D \\
\end{longtable}
```

looks like

A table

A	B
C	D

This package does it, too, so you can use this command now within every floating environment like `figure` or `table`, like here:

```
\begin{table}
  \caption*{A table}
  \begin{tabular}{cc}
    A & B \\
    C & D \\
  \end{longtable}
\end{table}
```

\captionof
\captionof* Sometimes you want to typeset a caption *outside* a floating environment, putting a figure within a `minipage` for instance. For this purpose the caption package offers the command

```
\captionof{⟨float type⟩}[⟨lst_entry⟩]{⟨heading⟩}   .
```

Note that the first argument, the ⟨*float type*⟩, is mandatory here, because the `\captionof` command needs to know which name to put into the caption label (e.g. “Figure” or “Table”) and in which list to put the contents entry. An example:

```
\captionof{figure}{A figure}
\captionof{table}{A table}
```

typesets captions like this:

Figure 20: A figure

Table 6: A table

The star variant `\captionof*` has the same behaviour as the `\caption*` command: it typesets the caption without label and without entry to the list of figures or tables.

Please use both `\captionof` and `\captionof*` only *inside* environments (like `minipage` or `\parbox`), otherwise a page break can appear between content and caption. Furthermore some strange effects could occur (e.g., wrong spacing around captions).

`\ContinuedFloat` Sometimes you want to split figures or tables without giving them their own reference number. This is what the command

```
\ContinuedFloat
```

is for; it should be used as first command inside the floating environment. It prevents the increment of the relevant counter so a figure or table with a `\ContinuedFloat` in it gets the same reference number as the figure or table before.

An example:

```
\begin{table}
\caption{A table}
...
\end{table}
...
\begin{table}\ContinuedFloat
\caption{A table (cont.)}
...
\end{table}
```

gives the following result:

Table 7: A table

...

Table 7: A table (cont.)

`\captionsetup` We already know the `\captionsetup` command (see section 2: *"Using the package"*), but this time we get enlighten about the optional argument ⟨*float type*⟩.

Remember, the syntax of this command is

`\captionsetup[`⟨*float type*⟩`]{`⟨*options*⟩`}` .

If a ⟨*float type*⟩ gets specified, all the ⟨*options*⟩ don't change anything at this time. Instead they only get marked for a later use, when a caption inside of a floating environment of the particular type ⟨*float type*⟩ gets typeset. For example

`\captionsetup[figure]{`⟨*options*⟩`}`

forces captions within a `figure` environment to use the given ⟨*options*⟩.

Here comes an example to illustrate this:

```
\captionsetup{font=small}
\captionsetup[figure]{labelfont=bf}
```

gives captions like this:

Figure 21: A figure

Table 8: A table

As you see the command `\captionsetup[figure]{labelfont=bf}` only changed the font of the figure caption labels, not touching all other ones.

`\clearcaptionsetup` If you want to get rid of these parameters marked for an automatic use within a particular environment you can use the command

```
\clearcaptionsetup{⟨Typ⟩}   .
```

For example `\clearcaptionsetup{figure}` would clear the extra handling in the example above:

Figure 22: A figure

Table 9: A table

As ⟨*float type*⟩ you can usually give one of these only two: `figure` and `table`. But as we will see later some LaTeX packages exist (like the float package for example) who can define additional floating enviroments and these two commands also works with them.

5 Do it yourself!

A family of commands is provided to allow users to define their own formats. This enables information on separators, justification, fonts, and styles to be associated with a name and kept in one place (these commands need to appear in the document preamble, this is the part between `\documentclass` and `\begin{document}`).

`\DeclareCaptionFormat` You can define your own caption formats using the command

```
\DeclareCaptionFormat{⟨name⟩}{⟨code using #1, #2 and #3⟩}   .
```

At usage the system replaces #1 with the caption label, #2 with the separator and #3 with the text. So the standard format `default` is defined inside `caption.sty` as

```
\DeclareCaptionFormat{default}{#1#2#3\par}
```

`DeclareCaptionLabelFormat` Likewise you can define your own caption label formats:

```
\DeclareCaptionLabelFormat{⟨name⟩}{⟨code using #1 and #2⟩}
```

At usage #1 gets replaced with the name (e.g. “figure”) and #2 gets replaced with the reference number (e.g. “12”).

`\bothIfFirst`
`\bothIfSecond` When you define your own caption label formats and use the subfig package[10], too, you must take care of empty caption label names. For this purpose the commands

`\bothIfFirst{`⟨*first arg*⟩`}{`⟨*second arg*⟩`}` and
`\bothIfSecond{`⟨*first arg*⟩`}{`⟨*second arg*⟩`}`

are offered. `\bothIfFirst` tests if the first argument exists (means: is not empty), `\bothIfSecond` tests if the second argument exists. If it is so both arguments get typeset, otherwise none of them.

For example the standard label format `simple` isn't defined as

```
\DeclareCaptionLabelFormat{simple}{#1 #2}  ,
```

because this could cause an extra space if #1 is empty. Instead `simple` is defined as

```
\DeclareCaptionLabelFormat{simple}{\bothIfFirst{#1}{ }#2}
,
```

causing the space to appear only if the label name is present.

`eclareCaptionLabelSeparato` You can define your own caption label separators with

`\DeclareCaptionLabelSeparator{`⟨*name*⟩`}{`⟨*code*⟩`}` .

Again an easy example taken from `caption.sty` itself:

```
\DeclareCaptionLabelSeparator{colon}{: }
```

`eclareCaptionJustification` You can define your own caption justifications with

`\DeclareCaptionJustification{`⟨*name*⟩`}{`⟨*code*⟩`}` .

The ⟨*code*⟩ simply gets typeset just before the caption. E.g. using the justification `raggedright`, which is defined as

```
\DeclareCaptionJustification{raggedright}{\raggedright}
,
```

yields captions with all lines moved to the left margin.

`\DeclareCaptionFont` You can define your own caption fonts with

`\DeclareCaptionFont{`⟨*name*⟩`}{`⟨*code*⟩`}` .

For example this package defines the options `small` and `bf` as

`\DeclareCaptionFont{small}{\small}` and
`\DeclareCaptionFont{bf}{\bfseries}` .

`\DeclareCaptionStyle` The best one comes at last: You can define your own caption styles with

`\DeclareCaptionStyle{`⟨*name*⟩`}[`⟨*additional options*⟩`]{`⟨*options*⟩`}`

Remember, caption styles are just a collection of suitable options, saved under a given name. You can wake up these options at any time with the option `style=`⟨*style name*⟩.

All caption styles are based on the default set of options. (See section 3.5: *"Styles"* for a complete list.) So you only need to specify options which are different to them.

If you specify ⟨*additional options*⟩ they get used in addition when the caption fits into a single line and this check was not disabled with the option `singlelinecheck=off`.

Again a very easy example taken from `caption.sty`:

```
\DeclareCaptionStyle{default}[justification=centering]{}
```

5.1 Examples

If you would like to have a colon *and* a line break as caption separator you could define it this way:

```
\DeclareCaptionLabelSeparator{period-newline}{. \\}
```

Selecting this separator with `\captionsetup{labelsep=period-newline}` you get captions like this:

Figure 23.
White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

For short captions—which fit into one single line—this separator may not be satisfying, even when the automatically centering process is switched off (with `singlelinecheck=off`):

Figure 24.
A figure.

An own caption style which selects another caption separator automatically puts this right:

```
\DeclareCaptionStyle{period-newline}%
  [labelsep=period]{labelsep=period-newline}
```

Figure 24. A figure.

If you would like to keep the centering of these captions an appropriate definition is

```
\DeclareCaptionStyle{period-newline}%
  [labelsep=period,justification=centering]%
  {labelsep=period-newline}
```

Using this definition short captions look like

Figure 24. A figure.

while long ones still have a line break after the caption label.

Slightly changed, you also get centered captions if they are longer than one line:

```
\DeclareCaptionStyle{period-newline}%
  [labelsep=period]%
  {labelsep=period-newline,justification=centering}
```

Figure 25.
White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

Another example: You want captions to look like this:

White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

(Figure 26)

You could do it this way:

```
\DeclareCaptionFormat{reverse}{#3#2#1}
\DeclareCaptionLabelFormat{fullparens}{(\bothIfFirst{#1}{ }#2)}
\DeclareCaptionLabelSeparator{fill}{\hfill}
\captionsetup{format=reverse,labelformat=fullparens,
              labelsep=fill,font=small,labelfont=it}
```

Another example: The caption text should go into the left margin; a possible solution would be:

```
\DeclareCaptionFormat{llap}{\llap{#1#2}#3\par}
\captionsetup{format=llap,labelsep=quad,singlelinecheck=no}
```

As a result you would get captions like this:

Figure 27 White sand beaches. The pink smoothness of the conch shell. A sea abundant with possibilities. Duty-free shops filled with Europe's finest gifts and perfumes. Play your favorite game of golf amidst the tropical greens on one of the many championship courses.

6 Using non-standard document classes

New description v3.0d

The caption package was developed using the standard document classes `article`, `report` and `book`.

If you would like to use the caption package with the KOMA-Script classes or with the memoir class, you have to take into consideration that all the possibilities for customization of the captions the KOMA-Script classes or memoir class have to offer will get lost. (And they have a lot of possibilites to offer!) So class options like `tablecaptionabove` and commands like `\captionabove`, `\captionbelow`, `\captionformat`, `\figureformat`, `\tableformat`, `\setcapindent`, `\setcaphanging`, `\captionstyle` etc. will not work anymore. So make a wise decision!

Using the caption package together with document classes not mentioned so far is not recommended at the moment – unwanted layout changes, side effects or failures could occur. (But future versions of the caption package will contain adaptions for more document classes!

7 Using other packages

The caption package contains special adaptions to other packages who handle with captions, too, so the captions always should look like you have specified them to look like.

These are the packages the caption package is adapted to:

`float`	Gives you the possibility to define new floating environments
`hypcap`	Adjusting hyperref anchors of captions
`listings`	Typesets source code listings
`longtable`	Typesets tables spanned over multiple pages
`rotating`	Supports rotated figures and tables
`sidecap`	Offers captions *beside* figures or tables
`supertabular`	Typesets tables spanned over multiple pages

New feature v3.0b If you use one of the above packages together with the caption package you get the additional possibility to set up captions with

`\captionsetup[`⟨*environment*⟩`]{`⟨*options*⟩`}` .

These options will apply for captions inside these environments automatically. For example

```
\captionsetup[lstlisting]{labelfont=bf}
```

forces captions inside the `lstlisting` environment to have bold labels. (Please note that this do not work with the `sideways` environments offered by the rotating package.)

If a certain support is not desired you can switch it off using the caption package option

`\usepackage[...,`⟨*package*⟩`=no]{caption}` .

For example specifying the option `float=no` means you don't like the caption package to support the float package. (Note: You can specify these options only within the `\usepackage` command, especially *not* at a later time with `\captionsetup`.)

For further information about the supported packages please take a look at the documentation belonging to it or buy yourself The LaTeX Companion[1].

7.1 The float package

A very useful feature is provided by the float package[2]: It offers the float placement specifier `H` which is much more restrictive than the specifier `h` offered by LaTeX. While the latter one is only a recommendation to LaTeX to set the float "here", the `H` forces the float to appear exactly at the spot where it occurs in your input file and nowhere else.

Furthermore it offers different styles for floating environments, these styles are `plain`, `plaintop`, `ruled`, and `boxed`. You can link one of these styles to either new floating environments or to one of the existing environments `figure` and `table`.

If you are using the caption package together with the float package this caption style called `ruled` gets defined automatically:

```
\DeclareCaptionStyle{ruled}{labelfont=bf,labelsep=space}
```

This style represents the caption layout in `ruled` styled floats. For you as an end user this means that captions within `ruled` floats will always look like this, nevertheless what generic caption options do you specify:

Program 7.1 The first program. This hasn't got anything to do with the package but is included as an example. Note the `ruled` float style.

```
#include <stdio.h>

int main(int argc, char **argv)
{
        for (int i = 0; i < argc; ++i)
                printf("argv[%d] = %s\n", i, argv[i]);
        return 0;
}
```

If you want a different layout for `ruled` captions you have to define your own one using the command

`\DeclareCaptionStyle{ruled}{`⟨*options*⟩`}` .

This mechanism also works with all other float styles. If you want a special caption layout for `plain` or `boxed` floats for example you can simply define a suitable caption style with the same name as the float style.

Note: For successful cooperation you need the float package version 1.3 or newer.

7.2 The listings package

New description v3.0b

The listings package[6] is a source code printer for LaTeX. You can typeset stand alone files as well as listings with an environment similar to `verbatim` as well as you can print code snippets using a command similar to `\verb`. Many parameters control the output and if your preferred programming language isn't already supported, you can make your own definition.

Note: For successful cooperation you need the listings package version 1.2 or higher. You'll get an error message when using an older version!

7.3 The longtable package

The longtable package[7] offers the environment `longtable` which behaves similar to the `tabular` environment, but the table itself can span multiple pages.

Note: For successful cooperation you need the longtable package version 3.15 or newer.

7.4 The rotating package

The rotating package[8] offers the floating environments `sidewaysfigure` and `sideways-table` which are just like normal figures and tables but rotated by 90 degree. Furthermore they always use a full page on their own.

7.5 The sidecap package

New description v3.0b

The sidecap package[9] offers the floating environments `SCfigure` and `SCtable` which are like normal figures and tables but the caption will be put *beside* the contents.

The sidecap package offers it's own options for justification. If set, they will override the one specified with the caption option `justification=` for captions beside their contents.

`listof=` Using the sidecap package you will probably notice that suppressing the entry in the list of figures or tables with `\caption[]{...}` won't work inside these environments. This is caused by the implementation design of the sidecap package, but you can use `\captionsetup{listof=false}` inside the figure or table as an alternative here.

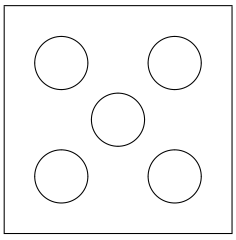

Figure 28: A small example with the caption beside the figure.

7.6 The supertabular package

The supertabular package[11] offers the environment `supertabular` which is quite similar to the `longtable` environment provided by the longtable package. Both offers the typesetting of tables which can span multiple pages. For a detailed discussion about the differences between these powerful packages please take a look at The LaTeX Companion[1].

7.7 Known incompatibilities

New description v3.0b

Using the caption package together with one of the following packages is not recommended; usually this would cause unwanted side effects or even errors:

ccaption, hvfloat, nonfloat

8 Compatibility to older versions

8.1 caption version $1.x$

This version of the caption package still supports the old options and commands provided by the version $1.x$ of this package. So there shouldn't occur any problems compiling old documents, but please don't mix old options and commands with the new ones. This isn't supported and can yield to ugly side effects.

Here comes a short oversight of the old options and commands and how they are replaced within this version of the caption package:

caption $1.x$	caption $3.x$
`normal`	`format=default`
`hang`	`format=hang`
`isu`	`format=hang`

caption 1.x	caption 3.x
`center`	`justification=centering`
`centerlast`	`justification=centerlast`
`anne`	`justification=centerlast`
`nooneline`	`singlelinecheck=off`
`scriptsize`	`font=scriptsize`
`footnotesize`	`font=footnotesize`
`small`	`font=small`
`normalsize`	`font=normalsize`
`large`	`font=large`
`Large`	`font=Large`
`up`	`labelfont=up`
`it`	`labelfont=it`
`sl`	`labelfont=sl`
`sc`	`labelfont=sc`
`md`	`labelfont=md`
`bf`	`labelfont=bf`
`rm`	`labelfont=rm`
`sf`	`labelfont=sf`
`tt`	`labelfont=tt`
`\setlength{\captionmargin}`	`margin=`⟨*amount*⟩
`\renewcommand{\captionfont}`	`\DeclareCaptionFont` + `\captionsetup{font=`⟨*name*⟩`}`
`\renewcommand{\captionsize}`	`\DeclareCaptionFont` + `\captionsetup{font=`⟨*name*⟩`}`
`\renewcommand{\captionlabelfont}`	`\DeclareCaptionLabelFont` + `\captionsetup{labelfont=`⟨*name*⟩`}`

8.2 caption2 version 2.x

Although they do very similar stuff the packages caption and caption2 have a very different implementation design. So this version of the caption package isn't compatible to the caption2 package at all. Of course for compiling old documents you can still use the caption2 package, the latest version is provided with this package. But newly created documents shouldn't use the caption2 package, please use the caption package instead as described in this manual.

9 Further reading

I recommend the following documents for further reading:

- The TEX FAQ - Frequently asked questions about TEX and LATEX:

  `http://faq.tug.org/`

- A French FAQ can be found at

  `http://www.grappa.univ-lille3.fr/FAQ-LaTeX/`

- epslatex from Keith Reckdahl contains many tips around graphics in LATEX 2ε. You will find this document in the directory

```
ftp://ftp.ctan.org/pub/tex/info/
```

as `epslatex.ps` and `epslatex.pdf`.

There is also a french translation available:

```
ftp://ftp.ctan.org/pub/tex/info/fepslatex.ps
```

10 Thanks

I would like to thank Katja Melzner, Steven D. Cochran, Frank Mittelbach, David Carlisle, Carsten Hinz, and Olga Lapko. Thanks a lot for all your help, ideas, patience, spirit, and support!

Also I would like to thank Harald Harders, Peter Löffler, Peng Yu, Alexander Zimmermann, Matthias Pospiech, Jürgen Wieferink, Christoph Bartoschek, Uwe Stöhr, and Ralf Stubner who all helped to make this package a better one.

11 The Implementation

The caption package consists of two parts – the kernel and the main package.
The kernel provides all the user commands and internal macros which are necessary for typesetting captions and setting parameters regarding these. While the standard LaTeX document classes provides an internal command called `\@makecaption` and no options to control its behavior (except the vertical skips above and below them), we provide similar commands called `\caption@make` and `\caption@@make`, but with a lot of options. Loading the kernel part do not change the output of a LaTeX document – it just provides functionality which can be used by LaTeX 2ε packages which typesets captions, like the caption package or the subfig package.
The caption package itself redefines the LaTeX commands `\caption`, `\@caption`, and `\@makecaption` and maps the latter one to `\caption@@make`, giving the user the possibility to control the captions of the floating environments `figure` and `table`. Furthermore it does similar to the caption stuff coming from other packages like the longtable package: Mapping the appropriate internal commands (like `\LT@makecaption`) to the ones offered by the caption kernel. So you can think of the caption package as a layer package, it simply provides adaption layers between the caption stuff coming from LaTeX itself or any LaTeX 2ε package and the caption stuff offered by the caption kernel.

11.1 Kernel

Identification

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{caption3}[2005/06/11 v3.0f caption3 kernel (AS)]
```

Generic helpers

`\@nameundef` This is the opposite part to `\@namedef` which is offered by the LaTeX kernel. We use it to remove the definition of some commands and keyval options after `\begin{document}` (to save TeX memory) or to remove caption options defined with `\captionsetup[`⟨*type*⟩`]`.

```
\providecommand*\@nameundef[1]{%
  \expandafter\let\csname #1\endcsname\@undefined}
```

`\l@addto@macro` The LaTeX 2ε kernel offers the internal helper macro `\g@addto@macro` which globally adds commands to any existising macro, like in `\AtBeginDocument`. This is the same but it works local, not global.

```
\providecommand\l@addto@macro[2]{%
  \begingroup
    \toks@\expandafter{#1#2}%
    \edef\@tempa{\endgroup\def\noexpand#1{\the\toks@}}%
  \@tempa}
```

`\bothIfFirst`
`\bothIfSecond` `\bothIfFirst` tests if the first argument is not empty, `\bothIfSecond` tests if the second argument is not empty. If yes both arguments get typeset, otherwise none of them.

```
\def\bothIfFirst#1#2{%
  \protected@edef\caption@tempa{#1}%
  \ifx\caption@tempa\@empty\else
    #1#2%
  \fi}
\def\bothIfSecond#1#2{%
```

```
  \protected@edef\caption@tempa{#2}%
  \ifx\caption@tempa\@empty\else
    #1#2%
  \fi}
```

`\caption@ifinlist` This helper macro checks if the first argument is in the list which is offered as second argument. So for example

```
    \caption@ifinlist{frank}{axel,frank,steven}{yes}{no}
```

would expand to `yes`.

```
\def\caption@ifinlist#1#2{%
  \let\next\@secondoftwo
  \edef\caption@tempa{#1}%
  \@for\caption@tempb:={#2}\do{%
    \ifx\caption@tempa\caption@tempb
      \let\next\@firstoftwo
    \fi}%
  \next}
```

`\caption@setbool` `\caption@ifbool` For setting and testing boolean options we offer these two helper macros:

`\caption@setbool{`⟨*name*⟩`}{`⟨*value*⟩`}`
(with `value = false/true/no/yes/off/on/0/1`)
`\caption@ifbool{`⟨*name*⟩`}{`⟨*if-clause*⟩`}{`⟨*else-clause*⟩`}`

```
\def\caption@setbool#1#2{%
  \caption@ifinlist{#2}{1,true,yes,on}{%
    \expandafter\let\csname caption@if#1\endcsname\@firstoftwo
  }{\caption@ifinlist{#2}{0,false,no,off}{%
    \expandafter\let\csname caption@if#1\endcsname\@secondoftwo
  }{%
    \PackageError{caption}{Undefined boolean value `#2'}{\caption@eh}%
  }}}
\def\caption@ifbool#1{\@nameuse{caption@if#1}}
```

Using the keyval package

We need the keyval package for option handling, so we load it here.

```
\RequirePackage{keyval}[1997/11/10]
```

`\undefine@key` This helper macro is the opposite of `\define@key`, it removes a keyval definition.

```
\providecommand*\undefine@key[2]{%
  \@nameundef{KV@#1@#2}\@nameundef{KV@#1@#2@default}}
```

`\DeclareCaptionOption` `\DeclareCaptionOption{`⟨*option*⟩`}{`⟨*code*⟩`}`
`\DeclareCaptionOption*{`⟨*option*⟩`}{`⟨*code*⟩`}`
The starred form makes the option only available during the lifetime of the current package, so they are only avail at `\usepackage`, but can't be used with `\captionsetup` later on.

```
\newcommand\DeclareCaptionOption{%
  \@ifstar{\caption@declareoption\AtEndOfPackage}%
          {\caption@declareoption\@gobble}}
```

```
\newcommand*\caption@declareoption[2]{%
  #1{\undefine@key{caption}{#2}}\define@key{caption}{#2}}
\@onlypreamble\DeclareCaptionOption
\@onlypreamble\caption@declareoption
```

`\captionsetup` `\captionsetup[`⟨*type*⟩`]{`⟨*keyval-list of options*⟩`}`
If 'type' is set, we simply save or append the option list, otherwise we 'execute' it with `\setkeys`.

```
\def\captionsetup{\@ifnextchar[\caption@setuptype\caption@setup}
\def\caption@setuptype[#1]#2{%
  \@ifundefined{caption@typ@#1}%
    {\@namedef{caption@typ@#1}{#2}}%
    {\expandafter\l@addto@macro\csname caption@typ@#1\endcsname{,#2}}}
\def\caption@setup{\setkeys{caption}}
```

`\caption@settype` `\caption@settype{`⟨*type*⟩`}`
Caption options which have been saved with `\captionsetup[`⟨*type*⟩`]` can be executed using this macro. (It simply executes the saved option list, if there is any.)

```
\def\caption@settype#1{%
  \@ifundefined{caption@typ@#1}{}{%
    \caption@esetup{\csname caption@typ@#1\endcsname}}}
```

`\caption@esetup` To execute a keyval-list of options saved within a macro we need this special version of `\caption@setup` which expands the argument first.

```
\def\caption@esetup#1{%
  \edef\caption@tempa{\noexpand\caption@setup{#1}}%
  \caption@tempa}
```

`\clearcaptionsetup` `\clearcaptionsetup{`⟨*type*⟩`}`
This removes the saved option list associated with ⟨*type*⟩.

```
\newcommand*\clearcaptionsetup[1]{\@nameundef{caption@typ@#1}}
```

`\showcaptionsetup` `\showcaptionsetup[`⟨*package*⟩`]{`⟨*type*⟩`}`
This command is for debugging issues: It shows the saved option list which is associated with ⟨*type*⟩.

```
\newcommand*\showcaptionsetup[2][\@firstofone]{%
  \GenericWarning{}{%
    #1 Caption Info: KV list on `#2'\MessageBreak
    #1 Caption Data: (%
    \@ifundefined{caption@typ@#2}{%
      % Empty -- print nothing.
    }{%
      \@nameuse{caption@typ@#2}%
    }%
    )}}
```

Errors

`\caption@eh` We only provide this simple error message as helper for the user.

```
\newcommand\caption@eh{%
  If you do not understand this error, please take a closer look\MessageBreak
  at the documentation of the `caption' package.\MessageBreak
  \@ehc}
```

Margin resp. width

`\captionmargin` `\captionwidth` `\ifcaption@width`

`\captionmargin` and `\captionwidth` contains the extra margin resp. the total width used for captions. Never set these values in a direct way, they are just accessible in user documents to provide compatibility to `caption.sty` v1.*x*.

```
\newdimen\captionmargin
\newdimen\captionwidth
\newif\ifcaption@width
```

```
\DeclareCaptionOption{margin}{\caption@setmargin{#1}}
\DeclareCaptionOption{width}{\caption@setwidth{#1}}
```

`\caption@setmargin` `\caption@setwidth`

Note that we can only set one at a time, 'margin' *or* 'width'. Which dimension is actually set will be recognized by `\ifcaption@width`.

```
\newcommand\caption@setmargin{%
  \caption@widthfalse
  \setlength\captionmargin}
\newcommand\caption@setwidth{%
  \caption@widthtrue
  \setlength\captionwidth}
```

Indentions

`\captionindent` `\captionparindent` `\captionhangindent`

These are the indentions we support.

```
\newdimen\captionindent
\newdimen\captionparindent
\newdimen\captionhangindent
```

```
\DeclareCaptionOption{indent}[\leftmargini]{\setlength\captionindent{#1}}% obsolet
\DeclareCaptionOption{indention}[\leftmargini]{\setlength\captionindent{#1}}
\DeclareCaptionOption{parindent}[\parindent]{\setlength\captionparindent{#1}}% cha
\DeclareCaptionOption{hangindent}[0pt]{\setlength\captionhangindent{#1}}% changed
```

Styles

`\DeclareCaptionStyle`

`\DeclareCaptionStyle{`⟨*name*⟩`}[`⟨*single-line-list-of-KV*⟩`]{`⟨*list-of-KV*⟩`}`

```
\newcommand*\DeclareCaptionStyle[1]{%
  \@ifnextchar[{\caption@declarestyle{#1}}{\caption@declarestyle{#1}[]}}
\def\caption@declarestyle#1[#2]#3{%
  \global\@namedef{caption@sls@#1}{#2}%
  \global\@namedef{caption@sty@#1}{#3}}
\@onlypreamble\DeclareCaptionStyle
\@onlypreamble\caption@declarestyle
```

```
\DeclareCaptionOption{style}{\caption@setstyle{#1}}
```

`\caption@setstyle`

`\caption@setstyle{`⟨*name*⟩`}`
`\caption@setstyle*{`⟨*name*⟩`}`
Selecting a caption style means saving the additional ⟨*single-line-list-of-KV*⟩ (this will be done by `\caption@sls`), resetting the caption options to the default ones (this will be done using `\caption@setdefault`) and executing the ⟨*list-of-KV*⟩ options (this will be done using `\caption@esetup`).
The starred version will give no error message if the given style is not defined.

```
\newcommand\caption@setstyle{%
  \@ifstar{\caption@@setstyle\@gobble}{\caption@@setstyle\@firstofone}}
\newcommand*\caption@@setstyle[2]{%
  \@ifundefined{caption@sty@#2}%
    {#1{\PackageError{caption}{Undefined caption style `#2'}{\caption@eh}}}%
{\expandafter\let\expandafter\caption@sls\csname caption@sls@#2\endcsname
     \caption@setdefault\caption@esetup{\csname caption@sty@#2\endcsname}}}
```

`\caption@setdefault`

This resets (nearly) all caption options to the default ones. (Note that this does not touch the skips and the positioning!)

```
\newcommand\caption@setdefault{\captionsetup{%
  format=default,labelformat=default,labelsep=default,justification=default,%
  font=default,labelfont=default,textfont=default,%
  margin=0pt,indention=0pt,parindent=0pt,hangindent=0pt,singlelinecheck}}
```

There is only one pre-defined style, called ‘default’. It’s a perfect match to the standard LaTeX document classes: If the caption fits in one single line, it is typeset centered.

```
\DeclareCaptionStyle{default}[indent=0pt,justification=centering]{}
```

Formats

`\DeclareCaptionFormat`

`\DeclareCaptionFormat{`⟨*name*⟩`}{`⟨*code with #1, #2, and #3*⟩`}`
`\DeclareCaptionFormat*{`⟨*name*⟩`}{`⟨*code with #1, #2, and #3*⟩`}`
The starred form causes the code being typeset in vertical (instead of horizontal) mode, but does not support the `indention=` option.

```
\def\DeclareCaptionFormat{%
  \@ifstar{\caption@declareformat\@gobble}{\caption@declareformat\@firstofone}}
\newcommand\caption@declareformat[3]{%
  \global\expandafter\let\csname caption@ifh@#2\endcsname#1%
  \global\long\expandafter\def\csname caption@fmt@#2\endcsname##1##2##3{#3}}
\@onlypreamble\DeclareCaptionFormat
\@onlypreamble\caption@declareformat
```

```
\DeclareCaptionOption{format}{\caption@setformat{#1}}
```

`\caption@setformat`

`\caption@setformat{`⟨*name*⟩`}`
Selecting a caption format simply means saving the code (in `\caption@fmt`) and if the code should be used in horizontal or vertical mode (`\caption@ifh`).

```
\newcommand*\caption@setformat[1]{%
  \@ifundefined{caption@fmt@#1}%
    {\PackageError{caption}{Undefined caption format `#1'}{\caption@eh}}%
    {\expandafter\let\expandafter\caption@ifh\csname caption@ifh@#1\endcsname
     \expandafter\let\expandafter\caption@fmt\csname caption@fmt@#1\endcsname}}
```

There are two pre-defined formats, called ‘@normal’ and ‘hang’.

```
\DeclareCaptionFormat{@normal}{#1#2#3\par}
\DeclareCaptionFormat{hang}{%
  \@hangfrom{#1#2}%
  \advance\captionparindent\hangindent
  \advance\captionhangindent\hangindent
  \caption@@par
  #3\par}
```

‘default’ usually maps to ‘@normal’.

```
\def\caption@fmt@default{\caption@fmt@@normal}
\def\caption@ifh@default{\caption@ifh@@normal}% bugfix v3.0e (05-04-28)
```

Label formats

DeclareCaptionLabelFormat

`\DeclareCaptionLabelFormat{`⟨*name*⟩`}{`⟨*code with #1 and #2*⟩`}`

```
\newcommand*\DeclareCaptionLabelFormat[2]{%
  \global\expandafter\def\csname caption@lfmt@#1\endcsname##1##2{#2}}
\@onlypreamble\DeclareCaptionLabelFormat
```

```
\DeclareCaptionOption{labelformat}{\caption@setlabelformat{#1}}
```

\caption@setlabelformat

`\caption@setlabelformat{`⟨*name*⟩`}`
Selecting a caption label format simply means saving the code (in `\caption@lfmt`).

```
\newcommand*\caption@setlabelformat[1]{%
  \@ifundefined{caption@lfmt@#1}%
    {\PackageError{caption}{Undefined caption label format `#1'}{\caption@eh}}%
    {\expandafter\let\expandafter\caption@lfmt\csname caption@lfmt@#1\endcsname}}
```

There are three pre-defined label formats, called ‘empty’, ‘simple’, and ‘parens’.

```
\DeclareCaptionLabelFormat{empty}{}
\DeclareCaptionLabelFormat{simple}{\bothIfFirst{#1}{\nobreakspace}#2}
\DeclareCaptionLabelFormat{parens}{\bothIfFirst{#1}{\nobreakspace}(#2)}
```

‘default’ usually maps to ‘simple’.

```
\def\caption@lfmt@default{\caption@lfmt@simple}
```

Label separators

lareCaptionLabelSeparator

`\DeclareCaptionLabelSeparator{`⟨*name*⟩`}{`⟨*code*⟩`}`

```
\newcommand\DeclareCaptionLabelSeparator[2]{%
  \global\long\@namedef{caption@lsep@#1}{#2}}
\@onlypreamble\DeclareCaptionLabelSeparator
```

```
\DeclareCaptionOption{labelsep}{\caption@setlabelseparator{#1}}
\DeclareCaptionOption{labelseparator}{\caption@setlabelseparator{#1}}
```

caption@setlabelseparator

`\caption@setlabelseparator{`⟨*name*⟩`}`
Selecting a caption label separator simply means saving the code (in `\caption@lsep`).

```
\newcommand*\caption@setlabelseparator[1]{%
  \@ifundefined{caption@lsep@#1}%
    {\PackageError{caption}{Undefined caption label separator `#1'}{\caption@eh}}%
    {\expandafter\let\expandafter\caption@lsep\csname caption@lsep@#1\endcsname}}
```

There are six pre-defined label separators, called ‘none’, ‘colon’, ‘period’, ‘space’, ‘quad’, and ‘newline’.

```
\DeclareCaptionLabelSeparator{none}{}
\DeclareCaptionLabelSeparator{colon}{: }
\DeclareCaptionLabelSeparator{period}{. }
\DeclareCaptionLabelSeparator{space}{ }
\DeclareCaptionLabelSeparator{quad}{\quad}
\DeclareCaptionLabelSeparator{newline}{\\}% 05-03-23 (v3.0f)
```

‘default’ usually maps to ‘colon’.

```
\def\caption@lsep@default{\caption@lsep@colon}
```

Justifications

clareCaptionJustification

`\DeclareCaptionJustification{`⟨*name*⟩`}{`⟨*code*⟩`}`

```
\newcommand*\DeclareCaptionJustification[2]{%
  \global\@namedef{caption@hj@#1}{#2}}
%\newcommand\DeclareCaptionJustification{\DeclareCaptionFont}
\@onlypreamble\DeclareCaptionJustification
```

```
\DeclareCaptionOption{justification}{\caption@setjustification{#1}}
```

\caption@setjustification

`\caption@setjustification{`⟨*name*⟩`}`
Selecting a caption justification simply means saving the code (in `\caption@hj`).

```
\newcommand*\caption@setjustification[1]{%
  \@ifundefined{caption@hj@#1}%
    {\PackageError{caption}{Undefined caption justification `#1'}{\caption@eh}}%
    {\expandafter\let\expandafter\caption@hj\csname caption@hj@#1\endcsname}}
%\newcommand\caption@setjustification{\caption@setfont{@hj}}
```

These are the pre-defined justification code snippets.

```
\DeclareCaptionJustification{justified}{}
\DeclareCaptionJustification{centering}{\centering}
\DeclareCaptionJustification{centerfirst}{\caption@centerfirst}
\DeclareCaptionJustification{centerlast}{\caption@centerlast}
\DeclareCaptionJustification{raggedleft}{\raggedleft}
\DeclareCaptionJustification{raggedright}{\raggedright}
```

‘default’ usually maps to ‘justified’.

```
\def\caption@hj@default{\caption@hj@justified}
```

\caption@centerfirst
\caption@centerlast

Please blame Frank Mittelbach for `\caption@centerfirst` and Anne Brüggemann-Klein for `\caption@centerlast :-)`

```
\newcommand\caption@centerfirst{%
  \edef\caption@normaladjust{%
    \leftskip\the\leftskip
    \rightskip\the\rightskip
    \parfillskip\the\parfillskip\relax}%
  \leftskip\z@\@plus -1fil%
  \rightskip\z@\@plus 1fil%
  \parfillskip\z@skip
  \noindent\hskip\z@\@plus 2fil%
  \@setpar{\@@par\@restorepar\caption@normaladjust}}
\newcommand\caption@centerlast{%
  \leftskip\z@\@plus 1fil%
  \rightskip\z@\@plus -1fil%
  \parfillskip\z@\@plus 2fil\relax}
```

We also support the upper-case commands offered by the ragged2e package. Note that these just map to their lower-case variants if the ragged2e package is not available.

```
\DeclareCaptionJustification{Centering}{%
  \caption@ragged\Centering\centering}
```

```
\DeclareCaptionJustification{RaggedLeft}{%
  \caption@ragged\RaggedLeft\raggedleft}
\DeclareCaptionJustification{RaggedRight}{%
  \caption@ragged\RaggedRight\raggedright}
```

`\caption@ragged`

`\caption@ragged` will be basically defined as

```
\AtBeginDocument{\IfFileExists{ragged2e.sty}%
  {\RequirePackage{ragged2e}\let\caption@ragged\@firstoftwo}%
  {\let\caption@ragged\@secondoftwo}}
```

but with a warning if the ragged2e package is not avail. (This warning will by typeout only ones per option, that's why we need the `caption\string#1` stuff.)

```
\newcommand*\caption@ragged[2]{%
  \@ifundefined{caption\string#1}{%
    \PackageWarning{caption}{%
      Cannot locate the 'ragged2e' package, therefore\MessageBreak
      substituting \string#2 for \string#1\MessageBreak}%
    \global\@namedef{caption\string#1}}{}%
  #2}
```

```
\AtBeginDocument{\IfFileExists{ragged2e.sty}{%
  \RequirePackage{ragged2e}\let\caption@ragged\@firstoftwo}{}}
```

Fonts

`\DeclareCaptionFont`

`\DeclareCaptionFont{`⟨*name*⟩`}{`⟨*code*⟩`}`

```
\newcommand\DeclareCaptionFont[2]{%
  \define@key{caption@fnt}{#1}[]{\g@addto@macro\caption@tempa{#2}}}
\@onlypreamble\DeclareCaptionFont
```

```
\DeclareCaptionOption{font}{\caption@setfont{font}{#1}}
\DeclareCaptionOption{labelfont}{\caption@setfont{labelfont}{#1}}
\DeclareCaptionOption{textfont}{\caption@setfont{textfont}{#1}}
```

`\caption@setfont`

`\caption@setfont{`⟨*command*⟩`}{`⟨*keyval-list of names*⟩`}`

Selecting a caption font means saving all the code snippets (in `#1`). Because we use `\setkeys` recursive here we need to put this into an extra group and collect all the code snippets in `\caption@tempa` first.

```
\newcommand*\caption@setfont[2]{%
  \let\caption@tempa\@empty
  \begingroup
    \setkeys{caption@fnt}{#2}%
  \endgroup
  \expandafter\let\csname caption#1\endcsname\caption@tempa}
```

These are the pre-defined font code snippets.

```
\DeclareCaptionFont{default}{}
```

```
\DeclareCaptionFont{scriptsize}{\scriptsize}
\DeclareCaptionFont{footnotesize}{\footnotesize}
\DeclareCaptionFont{small}{\small}
\DeclareCaptionFont{normalsize}{\normalsize}
\DeclareCaptionFont{large}{\large}
\DeclareCaptionFont{Large}{\Large}
```

```
\DeclareCaptionFont{up}{\upshape}
\DeclareCaptionFont{it}{\itshape}
\DeclareCaptionFont{sl}{\slshape}
\DeclareCaptionFont{sc}{\scshape}
\DeclareCaptionFont{md}{\mdseries}
\DeclareCaptionFont{bf}{\bfseries}
\DeclareCaptionFont{rm}{\rmfamily}
\DeclareCaptionFont{sf}{\sffamily}
\DeclareCaptionFont{tt}{\ttfamily}
```

`\captionsize` The old versions 1.*x* of the caption package offered this command to setup the font size used for captions. We still do so old documents will work fine.

```
\providecommand\captionsize{}
```

```
\DeclareCaptionOption{size}{\caption@setfont{size}{#1}}% changed v3.0a
```

Vertical spaces before and after captions

`\abovecaptionskip` `\belowcaptionskip` Usually these skips are defined within the document class, but some document classes don't do so.

```
\@ifundefined{abovecaptionskip}{%
  \newlength\abovecaptionskip\setlength\abovecaptionskip{10\p@}}{}
\@ifundefined{belowcaptionskip}{%
  \newlength\belowcaptionskip\setlength\belowcaptionskip{0\p@}}{}
```

```
\DeclareCaptionOption{aboveskip}{\setlength\abovecaptionskip{#1}}
\DeclareCaptionOption{belowskip}{\setlength\belowcaptionskip{#1}}
\DeclareCaptionOption{skip}{\setlength\abovecaptionskip{#1}}% new 3.0d
```

Positioning

These macros handle the right position of the caption. Note that the position is actually *not* controlled by the caption kernel options, but by the user (or the package) instead. The user can put the `\caption` command where ever he likes! So this stuff is only to give us a hint where to put the right skips, the user usually has to take care for himself that this hint actually matches the right position. The user can also try out the experimental setting `position=auto` which means that the caption package should try to guess the actual position of the caption for himself. (But in many cases, for example in `longtables`, this is doomed to fail, so it's not documented in the user part of the documentation.)

```
\DeclareCaptionOption{position}{\caption@setposition{#1}}
```

`\caption@setposition` Selecting the caption position means that we put `\caption@position` to the right value. Please do *not* use the internal macro `\caption@position` in your own package or document, but use the wrapper macro `\caption@iftop` instead.

```
\newcommand*\caption@setposition[1]{%
  \caption@ifinlist{#1}{d,default}{%
    \def\caption@position{\caption@defaultpos}%
  }{\caption@ifinlist{#1}{t,top,above}{%
    \let\caption@position\@firstoftwo
  }{\caption@ifinlist{#1}{b,bottom,below}{%
    \let\caption@position\@secondoftwo
  }{\caption@ifinlist{#1}{a,auto}{%
```

```
    \let\caption@position\@undefined
  }{%
    \PackageError{caption}{Undefined caption position `#1'}{\caption@eh}%
  }}}}}
```

\caption@defaultpos The default 'position' is 'bottom', this means that the (larger) skip will be typeset above the caption. This correspondents to the `\@makecaption` implementation in the standard LaTeX document classes.

```
%\caption@setdefaultpos{b}% default = bottom
\let\caption@defaultpos\@secondoftwo
```

\caption@iftop `\caption@iftop{`⟨*true-code*⟩`}{`⟨*false-code*⟩`}`
(If `\caption@position` is set to 'auto' we assume a 'bottom' position.)

```
\newcommand\caption@iftop{% bugfixed v3.0a, improved v3.0d
  \ifx\caption@position\@undefined
    \expandafter\@secondoftwo
  \else
    \expandafter\caption@position
  \fi}
```

\caption@fixposition This macro checks if the 'position' is set to 'auto'. If yes, `\caption@autoposition` will be called to set `\caption@position` to a proper value we can actually use.

```
\newcommand\caption@fixposition{%
  \ifx\caption@position\@undefined
    \caption@autoposition
  \fi}
```

\caption@autoposition We guess the actual position of the caption by checking `\prevdepth`.

```
\newcommand\caption@autoposition{% bugfixed v3.0a
  \ifvmode
    \ifodd\caption@debug\relax
      \edef\caption@tempa{\the\prevdepth}%
      \PackageInfo{caption}{\protect\prevdepth=\caption@tempa}%
    \fi
%
%   \caption@setposition{\ifdim\prevdepth>-\p@ b\else t\fi}%
    \ifdim\prevdepth>-\p@
      \let\caption@position\@secondoftwo
    \else
      \let\caption@position\@firstoftwo
    \fi
  \else
    \ifodd\caption@debug\relax
      \PackageInfo{caption}{no \protect\prevdepth}%
    \fi
%
%   \caption@setposition{b}%
    \let\caption@position\@secondoftwo
  \fi}
```

Hooks

\AtBeginCaption `\AtBeginDocument {`⟨*code*⟩`}`
\AtEndCaption `\AtEndDocument {`⟨*code*⟩`}`

These hooks can be used analogous to `\AtBeginDocument` and `\AtEndDocument`.

```
\newcommand\caption@beginhook{}
\newcommand\caption@endhook{}
\newcommand\AtBeginCaption{\l@addto@macro\caption@beginhook}
\newcommand\AtEndCaption{\l@addto@macro\caption@endhook}
```

Miscellaneous options

```
\DeclareCaptionOption{parskip}[5pt]{\AtBeginCaption{\setlength\parskip{#1}}}
```

```
\DeclareCaptionOption{listof}{\caption@setbool{lof}{#1}}% new v3.0b
\DeclareCaptionOption{singlelinecheck}[1]{\caption@setbool{slc}{#1}}
\DeclareCaptionOption{strut}{\caption@setbool{strut}{#1}}% new v3.0d
```

```
\DeclareCaptionOption{debug}{\def\caption@debug{#1}}
```

Initialization of parameters

```
\captionsetup{style=default,position=default,listof=1,strut=1,debug=0}
```

`\ifcaption@star` If the starred form of `\caption` is used, this will be set to `true`. (Note: This will be replaced by `\caption@iflabel` in future versions of the caption package, so I can use `\caption@setbool` so set this value.)

```
\newif\ifcaption@star
```

Typesetting the caption

`\caption@make` `\caption@make{`⟨*float name*⟩`}{`⟨*ref. number*⟩`}{`⟨*text*⟩`}`

```
\newcommand\caption@make[2]{%
  \caption@@make{\caption@lfmt{#1}{#2}}}
```

`\caption@@make` `\caption@@make{`⟨*caption label*⟩`}{`⟨*caption text*⟩`}`

```
\newcommand\caption@@make[2]{%
% \begingroup
  \caption@beginhook
  \caption@calcmargin
```

Special single-line treatment (Improvement v3.0d: moved to here)

```
  \caption@ifslc{%
    \ifx\caption@sls\@empty\else
      \caption@startslc
      \setbox\@tempboxa\hbox{\caption@@@make{#1}{#2}}%
      \ifdim\wd\@tempboxa >\captionwidth
        \caption@endslc
      \else
        \caption@endslc
        \caption@esetup\caption@sls
        \caption@calcmargin
      \fi
    \fi}{}%
```

Bugfix v3.0d: Use `\@tempdima` instead of `\captionmargin`, `\ifdim added` (04-10-26)

```
  \@tempdima\captionmargin
  \caption@ifh{\advance\@tempdima by \captionindent}%
  \ifdim\@tempdima=\z@\else
    \hskip\@tempdima
  \fi
```

Bugfix v3.0d: Use `\@tempdima` instead of `\captionwidth` (04-10-26)

```
  \@tempdima\captionwidth
  \caption@ifh{\advance\@tempdima by -\captionindent}%
  \caption@startbox\@tempdima
```

Bugfix v3.0b: `\ifdim` added (04-05-05)
Bugfix v3.0d: `\leavevmode` added (05/02/09)
Improvement v3.0d: `\caption@ifh` (05/02/09)

```
    \caption@ifh{%
      \ifdim\captionindent=\z@
        \leavevmode
      \else
        \hskip-\captionindent
      \fi}%
```

Bugfix v3.0d: `\strut` moved from here to `\caption@@@make`

```
    \caption@@@make{#1}{#2}%
```

```
  \caption@endbox
```

Bugfix v3.0d: This `\hskip` added

```
  \ifdim\captionmargin=\z@\else
    \hskip\captionmargin
  \fi
```

```
  \caption@endhook
% \endgroup
  \global\caption@starfalse}
```

`\caption@calcmargin` Calculate `\captionmargin` & `\captionwidth`, so both contain valid values.

```
\newcommand\caption@calcmargin{%
  \ifcaption@width
    \captionmargin\hsize
    \advance\captionmargin by -\captionwidth
    \divide\captionmargin by 2
  \else
    \captionwidth\hsize
    \advance\captionwidth by -2\captionmargin
  \fi
%
  \ifodd\caption@debug\relax
    \PackageInfo{caption}{\protect\hsize=\the\hsize,
      \protect\margin=\the\captionmargin,
      \protect\width=\the\captionwidth}%
  \fi}
```

`\caption@startslc` Re-define anything which would disturb the single line check
Bugfix v3.0b: re-definition of `\label` & `\@footnotetext` was missing here
Improvement v3.0b: re-define `\stepcounter` instead of `\footnote(mark)`
Improvement v3.0d: `\let\caption@hj\relax` added

```
\newcommand\caption@startslc{%
```

```
  \begingroup
  \let\label\@gobble\let\@footnotetext\@gobble
  \def\stepcounter##1{\advance\csname c@##1\endcsname\@ne\relax}%
  \let\caption@hj\relax}
\newcommand\caption@endslc{%
  \endgroup}
```

\caption@startbox
\caption@endbox

These macros start and end the box which surrounds the caption paragraph.

```
\newcommand*\caption@startbox[1]{\vbox\bgroup\hsize#1}%
%\newcommand*\caption@startbox[1]{\vbox\bgroup\setlength\hsize{#1}\@parboxrestore}
\newcommand*\caption@endbox{\egroup}
%\newcommand*\caption@endbox{\@finalstrut\strutbox\@@par\egroup}
```

\caption@@@make

`\caption@@@make{`⟨*caption label*⟩`}{`⟨*caption text*⟩`}`
This one finally typesets the caption paragraph, without margin and indention.

```
\newcommand\caption@@@make[2]{%
```

Empty text? Then use no caption label separator.

```
  \caption@ifempty{#2}{% changed v3.0e
    \let\caption@lsep\relax
    \let\caption@ifstrut\@secondoftwo % added v3.0e
  }%
```

Take care that `\captionparindent` and `\captionhangindent` will be used to typeset the paragraph.

```
  \def\caption@@par{%
    \parindent\captionparindent\hangindent\captionhangindent}%
  \@setpar{\@@par\caption@@par}\caption@@par
```

Finally the caption will be typeset.

```
  \caption@hj\captionsize\captionfont
```

Bugfix v3.0e: Handling of `\ifcaption@star` changed

```
  \caption@fmt{\ifcaption@star\else{\captionlabelfont#1}\fi}%
              {\ifcaption@star\else{\captionlabelfont\caption@lsep}\fi}%
              {{\captiontextfont
```

Bugfix v3.0d: Use some kind of `\@startstrut\strutbox` instead of `\strut` (04-12-16)

```
                   \caption@ifstrut{\vrule\@height\ht\strutbox\@width\z@}{}%
```

Bugfix v3.0b: `\allowhyphens` added (04-05-06)

```
                   \nobreak\hskip\z@skip
                   #2%
```

Bugfix v3.0d: `\@finalstrut\strutbox` added (05-01-23)

```
%                  \caption@ifstrut{\vrule\@height\z@\@depth\dp\strutbox\@width\z@}{}
                   \caption@ifstrut{\@finalstrut\strutbox}{}%
                   \par}}}
```

\caption@ifempty

`\caption@ifempty{`⟨*text*⟩`}{`⟨*if-clause*⟩`}`
(new v3.0e, 05/05/05)

```
\newcommand\caption@ifempty[1]{%
  \def\caption@tempa{#1}%
  \def\caption@tempb{\ignorespaces}%
  \ifx\caption@tempa\caption@tempb
    \let\caption@tempa\@empty
  \fi
  \ifx\caption@tempa\@empty
    \expandafter\@firstofone
  \else
    \expandafter\@gobble
  \fi}
```

11.2 Main package

Identification

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{caption}[2005/06/28 v3.0g Customising captions (AS)]
```

Loading the caption kernel

```
\RequirePackage{caption3}
```

```
\DeclareCaptionOption{type}{\def\@captype{#1}}% new v3.0d
```

Float names

\caption@floatname

\caption@floatname{⟨*type*⟩}
Usually all float names (which partly build the caption label) follow the same naming convention. But some packages (for example the float package) do not, so we use this wrapper macro which can be extended later on.

```
\newcommand*\caption@floatname[1]{\@nameuse{#1name}}
```

Support for `figure` and `table`

```
\DeclareCaptionOption*{figureposition}{\captionsetup[figure]{position=#1}}%  new v
\DeclareCaptionOption*{tableposition}{\captionsetup[table]{position=#1}}%    new v
```

Configuration files

```
\DeclareCaptionOption{config}[caption]{%
   \InputIfFileExists{#1.cfg}{\typeout{*** Local configuration file
                                       #1.cfg used ***}}%
                             {\PackageWarning{caption}{Configuration
                               file #1.cfg not found}}}
```

Compatibility options (caption v1.*x*)

```
\DeclareCaptionOption*{normal}[]{\caption@setformat{normal}}
\DeclareCaptionOption*{isu}[]{\caption@setformat{hang}}
\DeclareCaptionOption*{hang}[]{\caption@setformat{hang}}
\DeclareCaptionOption*{center}[]{\caption@setjustification{centering}}
\DeclareCaptionOption*{anne}[]{\caption@setjustification{centerlast}}
\DeclareCaptionOption*{centerlast}[]{\caption@setjustification{centerlast}}
```

```
\DeclareCaptionOption*{scriptsize}[]{\def\captionfont{\scriptsize}}
\DeclareCaptionOption*{footnotesize}[]{\def\captionfont{\footnotesize}}
\DeclareCaptionOption*{small}[]{\def\captionfont{\small}}
\DeclareCaptionOption*{normalsize}[]{\def\captionfont{\normalsize}}
\DeclareCaptionOption*{large}[]{\def\captionfont{\large}}
\DeclareCaptionOption*{Large}[]{\def\captionfont{\Large}}
```

```
\DeclareCaptionOption*{up}[]{\l@addto@macro\captionlabelfont\upshape}
\DeclareCaptionOption*{it}[]{\l@addto@macro\captionlabelfont\itshape}
\DeclareCaptionOption*{sl}[]{\l@addto@macro\captionlabelfont\slshape}
\DeclareCaptionOption*{sc}[]{\l@addto@macro\captionlabelfont\scshape}
\DeclareCaptionOption*{md}[]{\l@addto@macro\captionlabelfont\mdseries}
\DeclareCaptionOption*{bf}[]{\l@addto@macro\captionlabelfont\bfseries}
\DeclareCaptionOption*{rm}[]{\l@addto@macro\captionlabelfont\rmfamily}
\DeclareCaptionOption*{sf}[]{\l@addto@macro\captionlabelfont\sffamily}
\DeclareCaptionOption*{tt}[]{\l@addto@macro\captionlabelfont\ttfamily}
```

```
\DeclareCaptionOption*{nooneline}[]{\caption@setbool{slc}{0}}
```

```
\caption@setbool{ruled}{0}
\DeclareCaptionOption*{ruled}[]{\caption@setbool{ruled}{1}}
```

Generic package support

`\DeclareCaptionPackage` Each single package support can be switched on or off by using the appropriate option. By default all of them are enabled.

```
\newcommand*\DeclareCaptionPackage[1]{%
  \caption@setbool{pkt@#1}{1}%
  \DeclareCaptionOption*{#1}{\caption@setbool{pkt@#1}{##1}}}
\AtEndOfPackage{\let\DeclareCaptionPackage\@undefined}
```

`\caption@ifpackage` `\caption@ifpackage{`⟨*package name*⟩`}{`⟨*package macro*⟩`}{`⟨*package code*⟩`}`

```
\newcommand\caption@ifpackage[3]{%
  \caption@ifbool{pkt@#1}{%
    \@ifundefined{#2}%
      {\let\next\AtBeginDocument}%
      {\let\next\@firstofone}%
  }{%
    \let\next\@gobble
  }%
%
  \ifodd\caption@debug\relax
    \edef\caption@tempa{%
      \caption@ifbool{pkt@#1}{%
        \@ifundefined{#2}{AtBeginDocument}{firstofone}%
      }{gobble}}%
    \PackageInfo{caption}{#1 = \caption@ifbool{pkt@#1}{1}{0} %
         (\@ifundefined{#2}{not }{}loaded -> \caption@tempa)}%
  \fi
%
  \@nameundef{caption@ifpkt@#1}%  bugfixed v3.0a
%
  \next{%
    \expandafter\ifx\csname #2\endcsname\relax
    \else
      #3
    \fi}}
\AtEndOfPackage{\let\caption@ifpackage\@undefined}
```

These are the packages we support:

```
\DeclareCaptionPackage{caption}
```

```
\DeclareCaptionPackage{float}
\DeclareCaptionPackage{floatrow}
\DeclareCaptionPackage{hyperref}
\DeclareCaptionPackage{hypcap}
\DeclareCaptionPackage{listings}
\DeclareCaptionPackage{longtable}
\DeclareCaptionPackage{rotating}
\DeclareCaptionPackage{sidecap}
\DeclareCaptionPackage{supertabular}
```

`\ProcessOptionsWithKV` We process our options using the keyval package.

```
\def\ProcessOptionsWithKV#1{% bugfixed v3.0a
  \let\@tempc\relax
  \let\caption@tempa\@empty
  \@for\CurrentOption:=\@classoptionslist\do{%
    \@ifundefined{KV@#1@\CurrentOption}%
    {}%
    {%
      \edef\caption@tempa{\caption@tempa,\CurrentOption,}%
      \@expandtwoargs\@removeelement\CurrentOption
        \@unusedoptionlist\@unusedoptionlist
    }%
  }%
  \edef\caption@tempa{%
    \noexpand\setkeys{#1}{%
      \caption@tempa\@ptionlist{\@currname.\@currext}%
    }%
  }%
  \caption@tempa
```

Bugfix, see `<400D360C.9678329F@gmx.net>` for details

```
  \let\CurrentOption\@empty
  \AtEndOfPackage{\let\@unprocessedoptions\relax}}
```

```
\ProcessOptionsWithKV{caption}
\let\ProcessOptionsWithKV\@undefined
```

```
\caption@ifbool{pkt@caption}{}{\endinput}
\@nameundef{caption@ifpkt@caption}
```

Usefull stuff

`\captionof` `\captionof`(*)`{`⟨*type*⟩`}[`⟨*lst_entry*⟩`]{`⟨*heading*⟩`}`

```
\def\captionof{\@ifstar{\caption@of{\caption*}}{\caption@of\caption}}
\newcommand*\caption@of[2]{\def\@captype{#2}#1}
```

`\ContinuedFloat` `\ContinuedFloat`

```
\providecommand\ContinuedFloat{%
  \ifx\@captype\@undefined
    \@latex@error{\noexpand\ContinuedFloat outside float}\@ehd
  \else
    \addtocounter\@captype\m@ne
    \caption@ContinuedFloat\@captype
  \fi}%
```

\caption@ContinuedFloat
ption@resetContinuedFloat

```
\let\caption@ContinuedFloat\@gobble
\let\caption@resetContinuedFloat\@gobble
```

Internal helpers

\caption@begin

`\caption@begin{`⟨*type*⟩`}` (changed in v3.0b+v3.0e)

```
\newcommand*\caption@begin[1]{%
  \caption@resetContinuedFloat{#1}%
  \begingroup
  \caption@setfloattype{#1}%
% \caption@setfnum{#1}%
  \ifx\caption@lfmt\caption@lfmt@default\else
    \@namedef{fnum@#1}{%
      \caption@lfmt{\caption@floatname{#1}}{\@nameuse{the#1}}}%
  \fi
  \caption@fixposition
  \global\let\caption@fixedposition\caption@position
  \caption@@begin{#1}}
```

\caption@beginex

`\caption@beginex{`⟨*type*⟩`}{`⟨*list entry*⟩`}`

```
\newcommand*\caption@beginex[1]{%
  \caption@begin{#1}%
  \caption@preparelof}
```

\caption@end

`\caption@end`

```
\newcommand*\caption@end{%
  \caption@@end
  \endgroup
  \let\caption@position\caption@fixedposition}
```

\caption@setfloattype

A macro for setting up the right float type within `\@caption`, `\LT@makecaption` etc. Usually this is equivalent to `\caption@settype` but I made it an own macro so I can extend it later on, for example if the float package is loaded.

```
\let\caption@setfloattype\caption@settype%  new v3.0a
```

\caption@letfloattype

`\caption@letfloattype{`⟨*type*⟩`}{`⟨*extra code*⟩`}`
(new in v3.0b, additional argument in v3.0e)

```
\newcommand*\caption@letfloattype[2]{%
  \def\caption@setfloattype##1{%
    \caption@settype{##1}#2\caption@settype{#1}}}
```

\caption@preparelof

`\caption@preparelof{`⟨*list entry*⟩`}`

```
\newcommand*\caption@preparelof[1]{%  changed v3.0b
  \caption@iflof%
    {\def\caption@tempa{#1}}%
    {\let\caption@tempa\@empty}%
  \ifx\caption@tempa\@empty
    \def\addcontentsline##1##2##3{}%
  \fi}
```

`\caption@@begin` `\caption@@end`

`\caption@@begin{`⟨*type*⟩`}`
`\caption@@end`

```
\let\caption@@begin\@gobble % new v3.0a
\let\caption@@end\@empty %    new v3.0a
```

Caption support

Some packages (like the hyperref package for example) redefines `\caption` and `\@caption`, too, but without chaining to the previous definition. So we have to use `\AtBeginDocument` here, so we can make sure our definition don't get lost.

```
\AtBeginDocument{%
  \let\caption@old\caption
  \let\caption@@old\@caption
```

```
  \@ifundefined{cc@caption}{%
```

`\caption` Define `\caption*` ...
(07/18/03: `\global` added, so this works with sidecap) (05/22/05: `\ContinuedFloat` added)

```
    \def\caption{\caption@caption\caption@old}%
    \def\caption@caption#1{%
      \@ifstar{\ContinuedFloat\global\caption@startrue #1[]}{#1}}%
```

`\@caption` Define `\caption[]{...}` ...

```
    \long\def\@caption#1[#2]#3{%
      \caption@beginex{#1}{#2}%
        \caption@@old{#1}[{#2}]{#3}%
      \caption@end}%
```

```
  }{%
```

Minimum captcont package support (bugfixed v3.0c, 04-07-15)

```
    \PackageInfo{caption}{captcont package v2.0 detected}%
    \def\caption@caption#1{#1}%  added v3.0c
  }%
}
```

`\@makecaption` `\@makecaption{`⟨*label*⟩`}{`⟨*text*⟩`}`

Original code (from `latex/base/classes.dtx`):

```
\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip
  \sbox\@tempboxa{#1: #2}%
  \ifdim \wd\@tempboxa >\hsize
    #1: #2\par
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
```

```
\renewcommand\@makecaption[2]{%
  \caption@iftop{\vskip\belowcaptionskip}{\vskip\abovecaptionskip}%
  \ifnum\caption@debug>1 %
    \llap{$\caption@iftop\downarrow\uparrow$ }%
  \fi
  \caption@@make{#1}{#2}%
  \caption@iftop{\vskip\abovecaptionskip}{\vskip\belowcaptionskip}}
```

KOMA-script classes support

(new in 3.0a)

```
\AtBeginDocument{\let\scr@caption\caption}
```

float package support

The float package usually do not use the LaTeX kernel command `\@caption` to typeset the caption but `\float@caption` instead. (`\@caption` will only be used if the float is re-styled with `\restylefloat*`.)
The main two things `\float@caption` is doing different are:

- The caption will be typeset inside a savebox called `\@floatcapt` so it can be placed above or below the float contents afterwards.
- `\@makecaption` will not be used to finally typeset the caption. Instead `\@fs@capt` will be used which definition is part of the float style. (Note that `\@fs@capt` will not typeset any vertical space above or below the caption; instead this space will be typeset by the float style code itself.)

So our main goal is to re-define `\float@caption` so our macro `\caption@@make` will be used instead of `\@fs@capt`.
To allow different caption styles for different float styles we will also determine the current float style (e.g. 'ruled') at run time and setup a caption style (and additional settings) with the same name, if defined.

`\caption@setfloatposition` First of all we provide a macro which converts `\@fs@iftopcapt` (which is part of a float style and controls where the caption will be typeset, above or below the float contents) to our `position=` setting. Since the spacing above and below the caption will be done by the float style and *not* by us this sounds quite useless. But in fact it isn't, since some packages based on the caption package (like the subfig package) could have an interest for this information and therefore use the `\caption@iftop` macro we provide in our kernel. Furthermore we need this information for ourself in `\captionof` which uses `\@makecaption` to finally typeset the caption.

```
\def\caption@setfloatposition{%
  \caption@setposition{\@fs@iftopcapt t\else b\fi}}
```

```
\caption@ifpackage{float}{@float@setevery}{%
  \PackageInfo{caption}{float package v1.3 (or newer) detected}%
```

Since `\float@caption` puts the float contents into a savebox we need a special version of `\captionof` which 'unfolds' this box afterwards, so the caption actually gets typeset. Furthermore we have to typeset the spacing above and below the caption for ourself, since this space is not part of the box.

Please note that this version of `\captionof` only works *outside* floating environments defined with the float package, so for example a `\captionof{Program}` used within a 'standard' `figure` or a `minipage` will work fine, but not within a re-styled `figure` or an `Example` environment defined with `\newfloat`. (We don't check for this so you'll get wired errors if you try to do so!)

`\caption@of@float` Usually no special action is necessary, so we define `\caption@of@float` to `\@gobble`. We will redefine it later on to `\@firstofone` to activate the code which 'unfolds' the savebox.

```
\let\caption@of@float\@gobble
```

`\caption@of` If the float is defined by the float package (which means `\fst@`⟨*type*⟩ is defined) we activate the special treatment for such captions typeset with `\captionof`. Furthermore we 'execute' this float style, so `\@fs@iftopcapt` is set to its proper value.

```
\renewcommand*\caption@of[2]{%
  \@ifundefined{fst@#2}{}{%
    \let\caption@of@float\@firstofone
    \@nameuse{fst@#2}\@float@setevery{#2}}%
  \def\@captype{#2}#1}%
```

`\float@caption` Our version of `\float@caption` nearly looks like our version of `\@caption`. The main differences are that `\@fs@capt` will be replaced by our `\caption@@make` and that the savebox called `\@floatcapt` will be unfolded if required. (See above)

```
\let\caption@@float\float@caption
\long\def\float@caption#1[#2]#3{%
  \caption@beginex{#1}{#2}%
    \let\@fs@capt\caption@@make
    \caption@@float{#1}[{#2}]{#3}%
    \caption@of@float{%
      \def\caption@@make##1##2{\unvbox\@floatcapt}%
      \@makecaption{}{}}%
  \caption@end}%
```

`\@float@setevery` `\@float@setevery{`⟨*float type*⟩`}` is provided by the float package; it's called every time a floating environment defined with `\newfloat` or `\restylefloat` begins. We use this hook to do some adaptions and to setup the proper caption style (if defined) and additional settings declared with `\captionsetup[`⟨*float style*⟩`]`.

```
\let\caption@float@setevery\@float@setevery
\def\@float@setevery#1{%
  \caption@float@setevery{#1}%
```

LaTeX and most packages use `\`⟨*type*⟩`name` to provide a macro for the float name – for example the command `\figurename` will usually contain the name of the floating environment `figure`:

```
\newcommand\figurename{Figure}
```

But the float package don't follow this naming convention, it uses `\fname@`⟨*type*⟩ instead. So we have to adapt `\caption@floatname` here, so our captions will be still ok.

```
  \def\caption@floatname##1{\@nameuse{fname@#1}}%
```

`\newfloat` and `\restylefloat` saves the *actual* definition of `\@caption` or `\float@caption` in `\@float@c@`⟨*captype*⟩ with `\let` (instead of using `\def`), so redefinitions of `\@caption` (and of course our redefinition of `\float@caption`) will never been used if the `\newfloat` or `\restylefloat` command takes place in front of the redefinitions provided by the caption or other packages like the hyperref package.
So here we determine if the user has used `\restylefloat` or `\restylefloat*` and bring `\@float@c@`⟨*captype*⟩ up-to-date. This is quite easy: If `\@float@c@`⟨*captype*⟩ is the same as the original or our own definition of `\float@caption`, the user has used `\restylefloat` (and `\float@caption` should be used), otherwise we assume he has used `\restylefloat*` (and `\@caption` should be used). (This test will fail if some other package re-defines `\float@caption`, too, so we have to assume that we are the only one.)

```
  \expandafter\let\expandafter\caption@tempa\csname @float@c@#1\endcsname
  \ifx\caption@tempa\float@caption
  \else\ifx\caption@tempa\@caption
  \else\ifx\caption@tempa\caption@@float
%   \ifodd\caption@debug\relax
%     \PackageInfo{caption}{\protect\@float@c@#1\space := \protect\float@caption
%   \fi
    \expandafter\let\csname @float@c@#1\endcsname\float@caption
  \else
%   \ifodd\caption@debug\relax
%     \PackageInfo{caption}{\protect\@float@c@#1\space := \protect\@caption}%
%   \fi
    \expandafter\let\csname @float@c@#1\endcsname\@caption
  \fi\fi\fi
```

If the floating environment is defined with `\newfloat` or `\restylefloat` (and *not* with `\restylefloat*`), `\@float@c@`⟨*type*⟩ will now be identical to `\float@caption`.

```
  \expandafter\ifx\csname @float@c@#1\endcsname\float@caption
```

First of all we set the caption position to it's proper value. (See above definition of `\caption@setfloatposition`)

```
    \caption@setfloatposition%  changed v3.0b+f
```

Now we'll have to determine the current float style. This is not so easy because the only hint provided by the float package is the macro `\fst@`⟨*float type*⟩ which points to the macro which represents the float style. So for example after

```
\floatstyle{ruled}
\newfloat{Program}{tbp}{lop}
```

`\fst@Program` will be defined as

```
\def\fst@Program{\fs@ruled}  .
```

So here is what we do: We copy `\fst@`⟨*float type*⟩ to `\caption@fst` and make it a string so we can gobble the first four tokens (= `\fs@`), so only the the name of the float style is left.

```
    \expandafter\let\expandafter\caption@fst\csname fst@#1\endcsname
    \edef\caption@fst{\noexpand\string\expandafter\noexpand\caption@fst}%
    \edef\caption@fst{\noexpand\@gobblefour\caption@fst}%
%   \edef\caption@fst{\caption@fst}%
```

`\caption@fst` now contains the float style (e.g. 'ruled') so we can use it to set the corresponding style (if defined) and additional options.

```
      \caption@setstyle*\caption@fst
      \caption@settype\caption@fst% new v3.0f
```

```
    \fi}%
```

`\fs@plaintop` `\fs@boxed` The float styles `plaintop` and `boxed` don't use our skip which can be set with `skip=` : `plaintop` uses `\belowcaptionskip` instead of `\abovecaptionskip`, and `boxed` uses a fixed space of `2pt`. So we patch the according float style macros here.

```
  \g@addto@macro\fs@plaintop{\def\@fs@mid{\vspace\abovecaptionskip\relax}}%
  \g@addto@macro\fs@boxed{\def\@fs@mid{\kern\abovecaptionskip\relax}}%
```

```
}
```

```
\captionsetup[boxed]{skip=2pt}%
```

To emulate the 'ruled' definition of `\@fs@capt` we provide a caption style 'ruled' with appropriate options. But if the package option `ruled` was specified, we setup additional caption settings to emulate the behaviour of the caption package v1.*x* option `ruled` instead: The current caption settings will be used, but without margin and without 'single-line-check'.

```
\caption@ifbool{ruled}{%
  \captionsetup[ruled]{margin=0pt,singlelinecheck=0}% new v3.0f
}{% v3.0f: "strut=0" added
  \DeclareCaptionStyle{ruled}{labelfont=bf,labelsep=space,strut=0}}
\let\caption@ifruled\@undefined
```

floatrow package support

The floatrow package is adapted for usage with the caption package. So the main work has already been done, there are only two little things we have to take care about.

```
\caption@ifpackage{floatrow}{flrow@setlist}{%
  \PackageInfo{caption}{floatrow package v0.1f (or newer) detected}%
```

`\caption@of` Captions typeset with `\captionof` should have the correct layout, so we have to 'activate' this layout here with `\flrow@setlist`.
(Please note that this version of `\captionof` has the same restrictions than the `\captionof` offered for floating environments defined with the float package, see above.)

```
  \renewcommand*\caption@of[2]{%
    \def\@captype{#2}\flrow@setlist{{#2}}#1}%
```

`\caption@floatname` The floatrow package followes the same naming convention as the float package; so we have to adapt `\caption@floatname` here, too.

```
  \renewcommand*\caption@floatname[1]{%
    \@nameuse{\@ifundefined{fname@#1}{#1name}{fname@#1}}}%
```

```
}
```

hyperref package support

When the hyperref package is used we have the problem that the usage of `\ContinuedFloat` will create duplicate hyperlinks – both `\@currentHlabel` and `\@currentHref` will be the same for the main float and the continued ones. So we have to make sure unique labels and references will be created each time. We do this by extending `\theHfigure` and `\theHtable`, so for continued floats the scheme

⟨*type*⟩.⟨*type #*⟩.⟨*continue #*⟩

will be used instead of

⟨*type*⟩.⟨*type #*⟩ .

(This implementation follows an idea from Steven Douglas Cochran.)
Note: This does not help if `\Hy@naturalnamestrue` is set.

```
\caption@ifpackage{hyperref}{theHfigure}{%
  \PackageInfo{caption}{hyperref package v6.74m (or newer) detected}%
```

`\caption@ContinuedFloat` If `\theH`⟨*type*⟩ is defined, we extend it with .⟨*continue #*⟩. Furthermore we set `\caption@resetContinuedFloat` to `\@gobble` so the continuation counter will not be reset to zero inside `\caption`.

```
  \def\caption@ContinuedFloat#1{%
    \@ifundefined{theH#1}{}{%
      \@ifundefined{CF@#1}{%
        \expandafter\newcount\csname CF@#1\endcsname
        \caption@resetContinuedFloat{#1}}{}%
      \global\advance\csname CF@#1\endcsname\@ne\relax
      \expandafter\l@addto@macro\csname theH#1\endcsname{.\expandafter\@arabic\csn
      \let\caption@resetContinuedFloat\@gobble
    }}%
```

`ption@resetContinuedFloat` If a continuation counter is defined, we reset it.

```
  \def\caption@resetContinuedFloat#1{%
    \@ifundefined{CF@#1}{}{\global\csname CF@#1\endcsname\z@\relax}}%
}
```

hypcap package support

When the hypcap package is used the following problems occur:

1. The hypcap package uses `\capstart`, `\hc@caption`, and `\hc@@caption` instead of `\caption` and `\@caption`. So we have to patch these macros, too.
2. `\caption` will be saved to `\hc@org@caption` when the hypcap package is loaded. We have to change this so our definition of `\caption` will always be used.
3. Both, `\capstart` and `\hc@@caption`, call `\hyper@makecurrent`. But since we offer `\ContinuedFloat` the float counters could have changed between these both calls! So we fix this by saving the hyperref reference (= `\@currentHref`) in `\capstart` and restoring it later on in `\hc@@caption`.

   (This also fixes the problem that hypcap does not work if `\Hy@hypertexnamesfalse` is set. This come in handy; we set it locally to avoid duplicated hyperref labels which could occur if `\ContinuedFloat` will be used.)

4. `\capstart` will call `\H@refstepcounter` to increase the float number. This collides with a following`\ContinuedFloat`, too, so we have to move this call from here to `\caption`. (Since we set `\Hy@hypertexnamesfalse` we can do this without problems.)

```
\caption@ifpackage{hypcap}{hc@caption}{%
  \PackageInfo{caption}{hypcap package v1.0 (or newer) detected}%
```

`\capstart` Here comes our version of `\capstart`:

```
  \let\caption@capstart\capstart
  \def\capstart{%
```

First of all we update `\hc@org@caption` to correct the problem that the hypcap package has saved an older definition of `\caption`.

```
    \let\hc@org@caption\caption
```

Since we don't know the float counter yet (it could be changed with `\ContinuedFloat` afterwards!) we make sure `\H@refstepcounter` will not be used and `\Hy@hypertexnamesfalse` is set, so unique hyperref labels will be generated by the original definition of `\capstart`. Afterwards we save the reference which was generated by `\hyper@makecurrent`.

```
    \begingroup
      \let\H@refstepcounter\@gobble
      \Hy@hypertexnamesfalse
      \caption@capstart
      \global\let\caption@currentHref\@currentHref
    \endgroup
```

The hypcap package restores the previous definition of `\caption` inside `\hc@@caption`. But since we will call this inside a group later on (making this restauration non-working), we have to make this for ourself inside `\caption`. (This would not be necessary if hypcap would do this inside `\hc@caption` instead of `\hc@@caption`.)
Additionally we increase the float counter here (since we have suppressed this in `\capstart`) and use `\caption@caption` here, so `\caption*` will work as expected.

```
    \def\caption{%
      \let\caption\hc@org@caption
      \H@refstepcounter\@captype
      \caption@caption\hc@caption}}%
```

`\hc@@caption` Here comes our version of `\hc@@caption`:

```
  \let\caption@hc@@caption\hc@@caption
  \long\def\hc@@caption#1[#2]#3{%
    \caption@beginex{#1}{#2}%
```

Beside the usual `\caption@begin` and `\caption@end` stuff (to support local options etc.) we make sure our saved hyperref reference will be used.

```
      \let\caption@hyper@makecurrent\hyper@makecurrent
      \def\hyper@makecurrent\@captype{%
        \let\hyper@makecurrent\caption@hyper@makecurrent
        \global\let\@currentHref\caption@currentHref}%

      \caption@hc@@caption{#1}[{#2}]{#3}%
    \caption@end}%

}
```

listings package support

```
\caption@ifpackage{listings}{lst@MakeCaption}{%
  \PackageInfo{caption}{listings package v1.2 (or newer) detected}%
%
  \let\caption@lst@MakeCaption\lst@MakeCaption
  \def\lst@MakeCaption#1{%
    \let\caption@setfloattype\caption@settype
    \def\caption@autoposition{\caption@setposition{#1}}%
    \caption@begin{lstlisting}%
      \caption@lst@MakeCaption{#1}%
    \caption@end}%
%
}
```

longtable package support

```
\caption@ifpackage{longtable}{LT@makecaption}{%
  \PackageInfo{caption}{longtable package v3.15 (or newer) detected}%
%
% Original code:
% \def\LT@makecaption#1#2#3{%
%   \LT@mcol\LT@cols c{\hbox to\z@{\hss\parbox[t]\LTcapwidth{%
%     % Based on article class "\@makecaption", "#1" is "\@gobble" in star
%     % form, and "\@firstofone" otherwise.
%     \sbox\@tempboxa{#1{#2: }#3}%
%     \ifdim\wd\@tempboxa>\hsize
%       #1{#2: }#3%
%     \else
%       \hbox to\hsize{\hfil\box\@tempboxa\hfil}%
%     \fi
%     \endgraf\vskip\baselineskip}%
%   \hss}}}
%
  \def\LT@makecaption#1#2#3{%
%   \noalign{\vskip...}%
%
    \LT@mcol\LT@cols c{\hbox to\z@{\hss\parbox[t]\hsize{%
%
      \caption@letfloattype{longtable}{% bugfixed v3.0e
        \ifdim\LTcapwidth=4in \else
          \caption@setwidth\LTcapwidth
        \fi}%
%     \caption@setdefaultpos{t}% default = top
      \let\caption@defaultpos\@firstoftwo% default = top
      \def\caption@autoposition{% does not work within \end(last)foot!
        \caption@setposition{\ifcase\LT@rows t\else b\fi}}%
%
      \caption@begin{table}%
%
%       This skip has 2 purposes:
%       1. Correct the heigth of the \parbox[t]. Usual it's the height of
%          the very first line, but because of our extra skip it's always 0pt.
%       2. Correct \arraystretch, which usually also affect the longtable
%          caption. (If this is not requested, take \strutbox instead.)
%       NOTE: This is only a quick workaround, it has to be revised later on.
%
        \vskip-\ht\@arstrutbox
%
        \caption@iftop{\vskip\belowcaptionskip}{\vskip\abovecaptionskip}%
%       \let\caption@beginbox\caption@beginLTbox
        \caption@startrue#1\caption@starfalse
        \caption@@make{#2}{#3}\endgraf
        \caption@iftop{\vskip\abovecaptionskip}{\vskip\belowcaptionskip}%
      \caption@end}%
%
    \hss}}}%
%
}
```

rotating package support

```
\caption@ifpackage{rotating}{@rotcaption}{%
  \PackageInfo{caption}{rotating package v2.0 (or newer) detected}%
%
  \let\caption@rot\rotcaption
  \def\rotcaption{\caption@caption\caption@rot}%
%
  \let\caption@@rot\@rotcaption
  \long\def\@rotcaption#1[#2]#3{%
    \caption@beginex{#1}{#2}%
      \caption@@rot{#1}[{#2}]{#3}%
    \caption@end}%
%
% Original code:
% \long\def\@makerotcaption#1#2{%
%   \setbox\@tempboxa\hbox{#1: #2}%
%   \ifdim \wd\@tempboxa > .8\vsize
%     \rotatebox{90}{%
%     \begin{minipage}{.8\textheight}#1: #2\end{minipage}%
%     }\par
%   \else%
%     \rotatebox{90}{\box\@tempboxa}%
%   \fi
%   \hspace{12pt}%
% }
%
  \long\def\@makerotcaption#1#2{%
    \rotatebox{90}{%
      \begin{minipage}{.8\textheight}%
        \caption@@make{#1}{#2}%
      \end{minipage}%
    }\par
    \hspace{12pt}}%
%
}
```

sidecap package support

```
\caption@ifpackage{sidecap}{endSC@FLOAT}{%
  \PackageInfo{caption}{sidecap package v1.4d (or newer) detected}%
```

```
%
% First of all, we let sidecap use an actual definition of \caption:
% (This is only required for version 1.5d of the sidecap package.)
%
\let\SC@caption=\caption
%
% Make \caption* and local settings (\captionsetup) work
%
\let\caption@SC@zfloat\SC@zfloat
\def\SC@zfloat#1#2#3[#4]{%
% #2 = ‘figure’ or ‘table’ => \SC@captype
  \caption@SC@zfloat{#1}{#2}{#3}[#4]%
%
  \global\let\SC@CAPsetup\@empty
  \def\captionsetup##1{\g@addto@macro\SC@CAPsetup{,##1}}%
%
  \let\caption@old\caption
% \def\caption{\renewcommand\captionsetup[1]{}\caption@caption\caption@old}%
  \def\caption{\caption@caption\caption@old}%
}%
%
% Before typesetting the caption, we set the captionmargin to zero
% because the extra margin is only disturbing here.
% (We don’t need to take care about the caption position because
%  the sidecap package set both \abovecaptionskip and \belowcaptionskip
%  to a skip of zero anyway.)
% Furthermore \SC@justify will override the caption justification, if set.
%
% Very old version (1.4): \SC@justify is not defined
% Older versions (1.5): \SC@justify is \relax when not set
% Newer versions (1.6): \SC@justify is \@empty when not set
%
\let\caption@endSC@FLOAT\endSC@FLOAT
\def\endSC@FLOAT{%
%  (Note that \@captype isn’t defined so far, this will be done inside
%   the original definition of \endSC@FLOAT.)
%  We set \@captype already here, so \captionsetup will
%  work with \@captype-based options, too. (new v3.0d)
  \let\@captype\SC@captype
  \caption@esetup\SC@CAPsetup
%
  \caption@letfloattype{SC\@captype}{% bugfixed v3.0e
    \caption@setmargin\z@
    \@ifundefined{SC@justify}{}{%
      \ifx\SC@justify\@empty\else
        \let\caption@hj\SC@justify
        \let\SC@justify\@empty
      \fi}}%
%
  \long\def\caption@ifempty##1{% bugfix v3.0e
    \ifx\SC@CAPtext\@empty
      \expandafter\@firstofone
    \else
      \expandafter\@gobble
      \fi}%
%
    \caption@endSC@FLOAT}%
%
}
```

supertabular package support

```
\def\caption@setSTposition{%
  \caption@setposition{\if@topcaption t\else b\fi}}
%
\caption@ifpackage{supertabular}{ST@caption}{%
  \PackageInfo{caption}{supertabular package detected}%
%
% Improvement v3.0e: \topcaption* and \bottomcaption*
  \let\caption@tablecaption\tablecaption
  \def\tablecaption{\caption@caption\caption@tablecaption}%
%
% Original code:
% \long\def\ST@caption#1[#2]#3{\par%
%   \addcontentsline{\csname ext@#1\endcsname}{#1}%
%                   {\protect\numberline{%
%                       \csname the#1\endcsname}{\ignorespaces #2}}
%   \begingroup
%     \@parboxrestore
%     \normalsize
%     \if@topcaption \vskip -10\p@ \fi
%     \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
%     \if@topcaption \vskip 10\p@ \fi
%   \endgroup}
%
  \let\caption@ST\ST@caption
  \long\def\ST@caption#1[#2]#3{\par%  bugfixed v3.0a
    \caption@letfloattype{supertabular}{}%
    \let\caption@fixposition\caption@setSTposition
    \caption@beginex{#1}{#2}%
      \addcontentsline{\csname ext@#1\endcsname}{#1}%
                      {\protect\numberline{%
                          \csname the#1\endcsname}{\ignorespaces #2}}%
      \@parboxrestore
      \normalsize
      \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
    \caption@end}%
%
}
```

References

[1] Frank Mittelbach and Michel Goossens: *The LaTeX Companion (2nd. Ed.)*, Addison-Wesley, 2004.

[2] Anselm Lingnau: *An Improved Environment for Floats*, 2001/11/08

[3] Olga Lapko: *The floatrow package documentation*, 2005/05/22

[4] Sebastian Rahtz: *Hypertext marks in LaTeX*, 2003/11/30

[5] Heiko Oberdiek: *The hypcap package – Adjusting anchors of captions* 2001/08/27

[6] Carsten Heinz: *The Listings Package*, 2004/02/13

[7] David Carlisle: *The longtable package*, 2000/10/22

[8] Sebastian Rahtz and Leonor Barroca: *A style option for rotated objects in LaTeX*, 1997/09/26

[9] Rolf Niepraschk und Hubert Gäßlein: *The sidecap package*, 2003/06/06

[10] Steven D. Cochran: *The subfig package*, 2004/01/16

[11] Johannes Braams und Theo Jurriens: *The supertabular environment*, 2002/07/19