

# Typesetting captions with the caption package<sup>\*</sup>

Axel Sommerfeldt  
caption@sommerfee.de

2007/02/20

## Abstract

The caption package offers customization of captions in floating environments such `figure` and `table` and cooperates with many other packages.<sup>1</sup>

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Using the package</b>	<b>3</b>
<b>3</b>	<b>Options</b>	<b>3</b>
3.1	Formatting . . . . .	3
3.2	Justification . . . . .	6
3.3	Fonts . . . . .	8
3.4	Margins and further paragraph options . . . . .	9
3.5	Styles . . . . .	10
3.6	Skips . . . . .	11
<b>4</b>	<b>Useful stuff</b>	<b>12</b>
<b>5</b>	<b>Do it yourself!</b>	<b>15</b>
5.1	Further Examples . . . . .	17
<b>6</b>	<b>Using non-standard document classes</b>	<b>19</b>

---

<sup>\*</sup>This package has version number v3.0o, last revised 2007/03/09.

<sup>1</sup>A complete re-work of the user interface done with Steven D. Cochran and Frank Mittelbach has lead to this new enhanced version 3.0.

<b>7</b>	<b>Using other packages</b>	<b>19</b>
7.1	The float package . . . . .	20
7.2	The listings package . . . . .	21
7.3	The longtable package . . . . .	21
7.4	The rotating package . . . . .	21
7.5	The sidecap package . . . . .	21
7.6	The supertabular package . . . . .	22
7.7	Known incompatibilities . . . . .	22
<b>8</b>	<b>Compatibility to older versions</b>	<b>22</b>
8.1	The caption package version 1.x . . . . .	22
8.2	The caption2 package version 2.x . . . . .	24
<b>9</b>	<b>Further reading</b>	<b>25</b>
<b>10</b>	<b>Thanks</b>	<b>26</b>
<b>11</b>	<b>The Implementation</b>	<b>27</b>
11.1	Kernel . . . . .	27
11.2	Main package . . . . .	45

# 1 Introduction

Within the standard L<sup>A</sup>T<sub>E</sub>X 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; for example, it would be nice if you could 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.

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

## 2 Using the package

`\usepackage` Insert

```
\usepackage[<options>]{caption}[2007/03/09]
```

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 consists 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:

New description v3.0h	<code>plain</code>	Typesets the captions as a normal paragraph. (This is the default behaviour, it is adapted from the standard $\LaTeX$ document classes.)
	<code>hang</code>	Indents the caption text, so it will ‘hang’ under the first line of the text.
	...	Own formats can be defined using <code>\DeclareCaptionFormat</code> . (See section 5: “ <i>Do it yourself</i> ”)

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.

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

`indentation=<amount>`.

Two examples:

`format=plain,indentation=.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,indentation=-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 four 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 <code>simple</code> ). (This is the default behaviour.)
empty	The caption label will be empty. (This option makes sense when used together with other options like <code>labelsep=none</code> .)
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.
...	Own label formats can be defined using <code>\DeclareCaptionLabelFormat</code> . (See section 5: “ <i>Do it yourself</i> ”)

An example: Using the options

`labelformat=parens, labelsep=quad`

gives 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 option

`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 makes sense when used together with other options like <code>labelformat=empty</code> .)
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 <code>\quad</code> .
newline	The caption label and text will be separated by a line break ( <code>\\</code> ).

New feature v3.0h	endash	The caption label and text will be separated by an en-dash, surrounded by spaces ( -- ).
	...	Own separators can be defined using \DeclareCaptionLabelSeparator. (See section 5: “Do it yourself”)

Three 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.

```
labelsep=endash
```

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.

## 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:

<code>justified</code>	Typesets the caption as a normal paragraph. (This is the default.)
<code>centering</code>	Each line of the caption will be centered.
<code>centerlast</code>	The last line of each paragraph of the caption text will be centered.
<code>centerfirst</code>	Only the first line of the caption will be centered.
<code>raggedright</code>	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 <code>\RaggedRight</code> of the <code>ragged2e</code> package will be used to achieve this. The main difference is that the word breaking algorithm of $\text{\TeX}$ will work inside captions.
raggedleft	Each line of the caption will be moved to the right margin.
...	Own justifications can be defined using <code>\DeclareCaptionJustification</code> . (See section 5: “ <i>Do it yourself</i> ”)

Three examples:

```
justification=centerlast
```

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.

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

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.

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

Figure 12

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=	The standard $\text{\LaTeX}$ document classes ( <code>article</code> , <code>report</code> , and <code>book</code> ) automatically center a caption if it fits in one single line:
------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Figure 13: A short caption.



The caption package adapts this behaviour and therefore usually ignores the justification you have set with `justification=` in such case. But you can switch this special treatment of such short captions off with the option

```
singlelinecheck=<bool> .
```

Using `false`, `no`, `off` or `0` for *<bool>* switches the extra centering off:

```
singlelinecheck=false
```

Doing so the above short caption would look like

Figure 13: A short caption.

You switch the extra centering on again by using `true`, `yes`, `on` or `1` for  $\langle bool \rangle$ . (The default is `on`.)

### 3.3 Fonts

`font=` 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 least one which only affects the caption text (`textfont`). You set them up using the options

```
font={\font options} ,
labelfont={\font options} , and
textfont={\font options} .
```

And these are the available font options:

<code>scriptsize</code>	Very small size
<code>footnotesize</code>	The size usually used for footnotes
<code>small</code>	Small size
<code>normalsize</code>	Normal size
<code>large</code>	Large size
<code>Large</code>	Even larger size
<code>up</code>	Upright shape
<code>it</code>	<i>Italic shape</i>
<code>sl</code>	<i>Slanted shape</i>
<code>sc</code>	SMALL CAPS SHAPE
<code>md</code>	Medium series
<code>bf</code>	<b>Bold series</b>
<code>rm</code>	Roman family
<code>sf</code>	Sans Serif family
<code>tt</code>	Typewriter family



... Own font options can be defined using `\DeclareCaptionFont`.  
(See section 5: “Do it yourself”)

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

Two examples:

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

**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.*

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

**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.*

### 3.4 Margins and further paragraph options

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

```
margin=<amount> or
width=<amount>
```

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

Two examples will illustrating this:

```
margin=10pt
```

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.

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

Figure 17: 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. It specifies the extra vertical space inserted between them:

```
parskip=<amount>
```

One example:

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

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.

hangindent= The option

```
hangindent=<amount>
```

is for setting up a hanging indentation 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 `indentation=` you already know about. But if the caption contains multiple paragraphs you will notice the difference:

```
format=hang,indentation=-.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.

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

Figure 20: 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=default, labelsep=default,
justification=default, font=default, labelfont=default,
textfont=default, margin=0pt, indentation=0pt, parindent=0pt
hangindent=0pt, singlelinecheck=true
```

Own caption styles can be defined using `\DeclareCaptionStyle`. (See section 5: “*Do it yourself*”)

### 3.6 Skips

`aboveskip=` The spaces above and below the caption are controlled by the skips `\abovcaptionskip`  
`belowskip=` and `\belowcaptionskip`. The standard L<sup>A</sup>T<sub>E</sub>X document classes `article`, `report`  
and `book` set `\abovcaptionskip` 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 `\abovcaptionskip` 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	
A	B	C	D
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 `longtables`.)

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.)

This option is especially useful when used together with the optional argument of the `\captionsetup` command. (See section 4: “*Useful stuff*” for details)

For example

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

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}2
```

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 the `caption` package offers an extension: If you leave the argument `⟨lst_entry⟩` empty, no entry in the list of figures or tables will be made. For example:

```
\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

---

<sup>2</sup>Please note that this is *not* sufficient when using a KOMA-Script document class, you need to use the *global* option `tablecaptionabove`, too.

A table

A	B
C	D

This package offers this feature, too, so you can use this command now within every floating environment like figure or table, like:

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

`\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*`

```
\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 21: 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 (usually done by `\caption`) 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 its 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 22:** 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{<float type>} .
```

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

Figure 23: A figure

Table 9: A table

As *<float type>* you can usually give one of these two only: `figure` or `table`. But as we will see later some L<sup>A</sup>T<sub>E</sub>X packages (like the `float`, `longtable`, and `sidecap` package for example) offer additional floating environments and these two commands can also be used 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 `plain` is pre-defined by the caption package as

```
\DeclareCaptionFormat{plain}{#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` If you define your own caption label formats and use the `subfig` package[10], you should  
`\bothIfSecond` 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 yes, both arguments get typeset, otherwise none of them.

For example the standard label format `simple` is *not* 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.

`\DeclareCaptionLabelSeparator` You can define your own caption label separators with

```
\DeclareCaptionLabelSeparator{<name>}{<code>} .
```

Again an easy example taken from the caption package itself:

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

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}
,
```

lets 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} .
```

New description  
v3.0h

The line spacing could be customized using the `setspace` package, for example:

```
\usepackage{setspace}
\DeclareCaptionFont{singlespacing}{\setstretch{1}}3
\DeclareCaptionFont{onehalfspacing}{\onehalfspacing}
\DeclareCaptionFont{doublespacing}{\doublespacing}
\captionsetup{font={onehalfspacing, small}, labelfont=bf}
```

---

<sup>3</sup> Note: Using `\singlespacing` does not work here since it contains a `\vskip` command.



**Figure 24:** 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.

An example which brings color into life:

```
\usepackage{color}
\DeclareCaptionFont{red}{\color{red}}
\DeclareCaptionFont{green}{\color{green}}
\DeclareCaptionFont{blue}{\color{blue}}
\captionsetup{labelfont=blue,textfont=green}
```

**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.

`\DeclareCaptionStyle` 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 the core of this package: The caption style default is pre-defined as

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

## 5.1 Further 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 26.

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 27.**

A figure.

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

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

**Figure 27.** A figure.

If you would like to keep the centering of these captions instead, an appropriate definition would be something like

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

Using this definition short captions look like

**Figure 27.** 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 28.**

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 29)*

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 30 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 possibilities to offer!) So class commands like `\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 adaptations for more document classes!)

## 7 Using other packages

The caption package contains special adaptations to other packages, so the captions should always look like you have specified them to look like.

These are the packages the caption package is adapted to:

<code>float</code>	Gives you the possibility to define new floating environments
<code>listings</code>	Typesets source code listings
<code>longtable</code>	Typesets tables spanned over multiple pages
<code>rotating</code>	Supports rotated figures and tables
<code>sidecap</code>	Offers captions <i>beside</i> figures or tables
<code>supertabular</code>	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} ,
```

where *environment* stands for any environment the above packages offer. (Please note that this do not work with the `sideways` environments offered by the rotating package.) For example

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

forces captions inside the `lstlisting` environment to have bold labels.

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 packages mentioned above please take a look at the documentation belonging to them or buy yourself The  $\text{\LaTeX}$  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  $\text{\LaTeX}$ . While the latter one is only a recommendation to  $\text{\LaTeX}$  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 a 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}{\langle options \rangle} .
```

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  $\text{\LaTeX}$ . 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

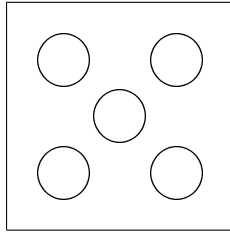
New description  
v3.0b

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[]{\dots}` 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 31:** 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  $\text{\LaTeX}$  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`, `ftcap`, `hvfloat`, and `nonfloat`

# 8 Compatibility to older versions

## 8.1 The caption package 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 cause ugly side effects.

Here comes a short oversight of the obsolete options and commands and how they have been replaced within this version of the caption package:

caption v1.x	caption v3.x
<code>normal</code>	<code>format=plain</code>
<code>hang</code>	<code>format=hang</code>
<code>isu</code>	<code>format=hang</code>
<code>center</code>	<code>justification=centering</code>
<code>centerlast</code>	<code>justification=centerlast</code>
<code>nooneline</code>	<code>singlelinecheck=off</code>
<code>scriptsize</code>	<code>font=scriptsize</code>
<code>footnotesize</code>	<code>font=footnotesize</code>

caption v1.x	caption v3.x
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

Beside the options for setting up the desired font there were also the commands `\captionsize` resp. `\captionfont` and `\captionlabelfont` who could be redefined with `\renewcommand` and allowed an alternate and more flexible way to change the font used for captions. This mechanism was replaced by the commands

```
\DeclareCaptionFont{...}{...}    and
\captionsetup{font=...,labelfont=...} .
```

(See section 5: “*Do it yourself*”)

Setting the margin for captions was done in v1.x with

```
\setlength{\captionmargin}{...} .
```

This was replaced by

```
\captionsetup{margin=...} .
```

(See section 3.4: “*Margins and further paragraph options*”)

For example the old-style code

```
\usepackage[hang,bf]{caption}
\renewcommand\captionfont{\small\sffamily}
\setlength\captionmargin{10pt}
```

will still work fine, but should be written today as

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

or

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

The quite exotic option `ruled` which allowed a partial usage of the caption settings for ruled floats defined with the float package will be emulated by this version of the caption package, too. But using this option is not recommended anymore since this version of the caption package offers a more flexible way for changing the captions of these floating environments:

```
\DeclareCaptionStyle{ruled}{...}
```

resp.

```
\captionsetup[ruled]{...} .
```

(See section 5: “*Do it yourself*”, 4: “*Useful stuff*”, and 7.1: “*The float package*”)

## 8.2 The caption2 package version 2.x

Although they do very similar stuff, the packages `caption` and its experimental and now obsolete variant `caption2` have a very different implementation design. Therefore a full compatibility could not be offered. For that reason you will still find a file called `caption2.sty` in this package distribution, so old documents using the `caption2` package will still compile fine.

Newly created documents should use the actual version of the `caption` package instead. In most cases it’s sufficient to replace the command

```
\usepackage{...}{caption2}
```

by

```
\usepackage{...}{caption} .
```

But some options and commands will not be emulated, so you can get error messages afterwards. This section will hopefully help you removing these errors. If you have problems migrating from `caption2` to `caption` please don’t hesitate to send me an e-mail asking for help.

In addition to the obsolete options shown in the last section these ones will be emulated, too:

caption2 v2.x	caption v3.x
<code>flushleft</code>	<code>justification=raggedright</code>
<code>flushright</code>	<code>justification=raggedleft</code>
<code>oneline</code>	<code>singlelinecheck=on</code>

Setting the margin for captions was done in v2.x with

```
\setcaptionmargin{...} resp. \setcaptionwidth{...} .
```

This was replaced by



```
\captionsetup{margin=...} resp. \captionsetup{width=...} .
```

(See section 3.4: “*Margins and further paragraph options*”)

Setting an indentation was done in *v2.x* with

```
\captionstyle{indent}  
\setlength\captionindent{...} .
```

This is now done with

```
\captionsetup{format=plain,indentation=...} .
```

The so-called single-line-check was controlled by the commands `\onelinecaptionsfalse` (for switching the check off) and `\onelinecaptionstrue` (for switching the check on). This was replaced by `\captionsetup{singlelinecheck=off}` resp. `\captionsetup{singlelinecheck=on}`. (See section 3.2: “*Justification*”)

The commands

```
\captionlabeldelim, \captionlabelsep, \captionstyle,  
\defcaptionstyle, \newcaptionstyle, and \renewcaptionstyle
```

do not have a simple replacement and therefore will not be emulated by this version of the caption package. (So using them will cause error messages.) Rewriting such code is not always easy and straight-ahead, but by conscientious reading of this manual you should find appropriate options and commands instead.

The *v2.x* option `ignoreLTcapwidth` do not have a replacement, too. But in most cases you can simply drop using that option because in this version of the caption package the value of `\LTcapwidth` will be ignored anyway (unless you set it to a different value than the default one). (See section 7.3: “*The longtable package*”)

## 9 Further reading

I recommend the following documents for further reading:

- The  $\text{\TeX}$  FAQ - Frequently asked questions about  $\text{\TeX}$  and  $\text{\LaTeX}$ :

<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 including graphics in  $\text{\LaTeX}$  2 $\epsilon$  documents. You will find this document in the directory

<ftp://ftp.ctan.org/pub/tex/info/epslatex/>

## 10 Thanks

I would like to thank Katja Melzner, Steven D. Cochran, Frank Mittelbach, David Carlisle, Carsten Heinz, Olga Lapko, and Keith Reckdahl. 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, Ralf Stubner, Geoff Vallis, Florian Keiler, Jürgen Göbel, Uwe Siart, Sang-Heon Shim, Henrik Lundell, David Byers, William Asquith, and Prof. Dr. Dirk Hoffmann, who all helped to make this package a better one.

## 11 The Implementation

The caption package consists of two parts – the kernel (`caption3.sty`) and the main package (`caption.sty`).

The kernel provides all the user commands and internal macros which are necessary for typesetting captions and setting parameters regarding these. While the standard  $\text{\LaTeX}$  document classes provides an internal command called `\@makecaption` and no options to control its behavior (except the vertical skips above and below the caption itself), we provide similar commands called `\caption@make` and `\caption@@make`, but with a lot of options which can be selected with `\captionsetup`. Loading the kernel part do not change the output of a  $\text{\LaTeX}$  document – it just provides functionality which can be used by  $\text{\LaTeX 2}_\epsilon$  packages which typesets captions, like the caption package or the subfig package.

The caption package itself redefines the  $\text{\LaTeX}$  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` or `supertabular` package): Mapping the appropriate internal commands (like `\LT@makecaption` or `\ST@caption`) to the ones offered by the caption kernel. So you can think of the caption package as a layer package, it simply provides adaptation layers between the caption stuff coming from  $\text{\LaTeX 2}_\epsilon$  itself or a  $\text{\LaTeX 2}_\epsilon$  package and the caption stuff offered by the caption kernel.

### 11.1 Kernel

#### Identification

```
1 \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2 \ProvidesPackage{caption3}[2007/03/30 v3.0o caption3 kernel (AR)]
```

#### Generic helpers

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

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

`\l@addto@macro` The  $\text{\LaTeX 2}_\epsilon$  kernel offers the internal helper macro `\g@addto@macro` which globally adds commands to any existing macro, like in `\AtBeginDocument`. This is the same but it works local, not global (using `\edef` instead of `\xdef`).

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

`\bothIfFirst` `\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.

```

10 \def\bothIfFirst#1#2{%
11   \protected@edef\caption@tempa{#1}%
12   \ifx\caption@tempa\@empty\else
13     #1#2%
14   \fi}

15 \def\bothIfSecond#1#2{%
16   \protected@edef\caption@tempa{#2}%
17   \ifx\caption@tempa\@empty\else
18     #1#2%
19   \fi}

```

`\AtBeginEnvironment` `\AtBeginEnvironment{<environment>}{<code>}`  
Allows code to be saved and executed at begin of given environments.

```

20 \providecommand*\AtBeginEnvironment[1]{%
21   \@ifundefined{#1}%
22     {\@latex@error{Environment #1 undefined}\@ehc
23     \@gobble}%
24     {\@ifundefined{caption@env@#1}%
25       {\expandafter\let\csname caption@env@#1\expandafter\endcsname
26         \csname #1\endcsname
27         \expandafter\let\csname caption@hook@#1\endcsname\@empty
28         \@namedef{#1}{\@nameuse{caption@hook@#1}\@nameuse{caption@env@#1}}}%
29       {}%
30       \expandafter\g@addto@macro\csname caption@hook@#1\endcsname}}
31 \@onlypreamble\AtBeginEnvironment

```

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

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

would expand to yes.

```

32 \def\caption@ifinlist#1#2{%
33   \let\next\@secondoftwo
34   \edef\caption@tempa{#1}%
35   \@for\caption@tempb:=#2\do{%
36     \ifx\caption@tempa\caption@tempb
37       \let\next\@firstoftwo
38     \fi}%
39   \next}

```

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

```

\caption@ifbool
\caption@undefbool

\caption@setbool{<name>}{<value>}
                    (with value = false/true/no/yes/off/on/0/1)
\caption@ifbool{<name>}{<if-clause>}{<else-clause>}
\caption@undefbool{<name>}

```

```

40 \def\caption@setbool#1#2{%
41   \caption@ifinlist{#2}{1,true,yes,on}{%
42     \expandafter\let\csname caption@if#1\endcsname \@firstoftwo
43   }{\caption@ifinlist{#2}{0,false,no,off}{%
44     \expandafter\let\csname caption@if#1\endcsname \@secondoftwo
45   }}%
46   \caption@error{Undefined boolean value `#2'}%
47   }}}
48 \def\caption@ifbool#1{\@nameuse{caption@if#1}}
49 \def\caption@undefbool#1{\@nameundef{caption@if#1}}

```

## Errors

`\caption@error` This is mainly identical to `\PackageError{caption}{#1}{\caption@eh}`.

```

50 \newcommand\caption@package{caption}
51 \newcommand*\caption@error[1]{%
52   \PackageError{caption@package}{#1}\caption@eh}

```

`\caption@eh` At the moment we only offer this simple error message as generic helper for the user.

```

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

```

## Using the keyval package

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

```

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

```

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

```

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

```

`\DeclareCaptionOption` `\DeclareCaptionOption{<option>}{<code>}`  
`\DeclareCaptionOption*{<option>}{<code>}`

We declare our options using these commands (instead of using `\DeclareOption` offered by  $\text{\LaTeX 2\epsilon}$ ), so the keyval package is used. The starred form makes the option available during the lifetime of the current package only, so they can be used with `\usepackage`, but *not* with `\captionsetup` later on.

```

60 \newcommand\DeclareCaptionOption{%
61   \@ifstar{\caption@declareoption\AtEndOfPackage}%
62   {\caption@declareoption\@gobble}}
63 \newcommand*\caption@declareoption[2]{%
64   #1{\undefine@key{caption}{#2}}\define@key{caption}{#2}}
65 \@onlypreamble\DeclareCaptionOption
66 \@onlypreamble\caption@declareoption

```

```
\captionsetup \captionsetup[<type>]{<keyval-list of options>}
```

If the optional argument ‘type’ is specified, we simply save or append the option list, otherwise we ‘execute’ it with \setkeys.

```
67 \newcommand\captionsetup{\@ifnextchar[\caption@setuptype\caption@setup}
68 \newcommand\caption@typ@{\caption@typ@} % This saves 74 words of TeX memory
69 \def\caption@setuptype[#1]#2{%
70   \@ifundefined{\caption@typ@#1}%
71     {\@namedef{\caption@typ@#1}{#2}}%
72     {\@expandafter\l@addto@macro\csname\caption@typ@#1\endcsname{, #2}}}%
73 \newcommand\caption@setup{\caption@setkeys{caption}}
```

\caption@setkeys This one simply calls \setkeys{*<package>*}{*<args>*} but lets the error messages refer to the *<package>* package instead of the keyval package.

```
74 \newcount\caption@keydepth \caption@keydepth=0
75 \newcommand*\caption@setkeys{%
76   \@dblarg\caption@@setkeys}
77 \long\def\caption@@setkeys[#1]#2#3{%
78   \ifnum\caption@keydepth =0\relax
79     \let\caption@KV@errx\KV@errx
80     \let\caption@KV@err\KV@err
81     \let\KV@errx\caption@error
82     \let\KV@err\caption@error
83   \fi
84   \advance\caption@keydepth\@ne
85 %
86   \def\caption@package{#1}%
87   \caption@Debug{\protect\setkeys{#2}{#3}}%
88   \setkeys{#2}{#3}%
89   \def\caption@package{caption}%
90 %
91   \advance\caption@keydepth\m@ne
92   \ifnum\caption@keydepth =0\relax
93     \let\KV@errx\caption@KV@errx
94     \let\KV@err\caption@KV@err
95   \fi}
```

\caption@settype \caption@settype{*<type>*}

Caption options which have been saved with \captionsetup[*<type>*] can be executed using this command. (It simply executes the saved option list, if there is any.)

```
96 \newcommand*\caption@settype[1]{%
97   \@ifundefined{\caption@typ@#1}{}{%
98     \caption@esetup{\csname\caption@typ@#1\endcsname}}}
```

\caption@esetup \caption@esetup{*<keyval-list of options>*}

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

```
99 \newcommand*\caption@esetup[1]{%
100   \edef\caption@tempa{\noexpand\caption@setup{#1}}%
101   \caption@tempa}
```

```

\clearcaptionsetup \clearcaptionsetup{<type>}
This removes the saved option list associated with <type>.
102 \newcommand*\clearcaptionsetup[1]{\@nameundef{\caption@typ@#1}}

\showcaptionsetup \showcaptionsetup[<package>]{<type>}
This comes for debugging issues: It shows the saved option list which is associated with
<type>.
103 \newcommand*\showcaptionsetup[2][\@firstofone]{%
104   \GenericWarning{}%
105   #1 Caption Info: KV list on `#2'\MessageBreak
106   #1 Caption Data: (%
107   \@ifundefined{\caption@typ@#2}{%
108     % empty -- print nothing
109   }{%
110     \@nameuse{\caption@typ@#2}%
111   }%
112   )}}

\caption@ProcessOptions We process our options using the keyval package, so we use this one instead of
\ProcessOptions offered by LATEX 2ε. (This code was taken from the hyperref pack-
age and improved.)
113 \newcommand*\caption@ProcessOptions[1]{%
114   \let\@tempc\relax
115   \let\caption@tempa\@empty
116   \@for\CurrentOption:=\@classoptionslist\do{%
117     \@ifundefined{KV@#1@CurrentOption}{%
118       }{%
119         \@ifundefined{KV@#1@CurrentOption @default}{%
120           \PackageInfo{#1}{Global option '\CurrentOption' ignored}%
121         }{%
122           \PackageInfo{#1}{Global option '\CurrentOption' processed}%
123           \edef\caption@tempa{\caption@tempa,\CurrentOption,}%
124           \@expandtwoargs\@removeelement\CurrentOption
125           \@unusedoptionlist\@unusedoptionlist
126         }%
127       }%
128     }%
129   \edef\caption@tempa{%
130     \noexpand\caption@setkeys{#1}{%
131       \caption@tempa\@ptionlist{\@currname.\@current}%
132     }%
133   }%
134   \caption@tempa
135   \let\CurrentOption\@empty
136   \AtEndOfPackage{\let\@unprocessedoptions\relax}}
137 \@onlypreamble\caption@ProcessOptions

```

### Margin resp. width

`\captionmargin` `\captionmargin` and `\captionwidth` contain the extra margin resp. the total width used for captions. Please never set these values in a direct way, they are just accessible in user documents to provide compatibility to `caption.sty v1.x`.  
Note that we can only set one value at a time, ‘margin’ *or* ‘width’. If `\captionwidth` is not zero we will take this value afterwards, otherwise `\captionmargin` and `\captionmarginx`.

```
138 \newdimen\captionmargin
139 \newdimen\captionmarginx
140 \newdimen\captionwidth

141 \DeclareCaptionOption{margin}{\setcaptionmargin{#1}}
142 \DeclareCaptionOption{width}{\setcaptionwidth{#1}}
```

`\setcaptionmargin` `\setcaptionmargin{<amount>}` `\setcaptionmargin{<amount>}`  
Please never use this in user documents, it’s just there to provide compatibility to `caption2.sty v2.x`.

```
143 \newcommand*\setcaptionmargin[1]{%
144   \captionwidth\z@
145   \caption@@setmargin#1,#1,\@nil\@@}
146 \def\caption@@setmargin#1,#2,#3\@@{%
147   \setlength\captionmargin{#1}%
148   \setlength\captionmarginx{#2}%
149   \advance\captionmarginx by -\captionmargin}
```

`\setcaptionwidth` `\setcaptionwidth{<amount>}` `\setcaptionwidth{<amount>}`  
Please never use this in user documents, it’s just there to provide compatibility to `caption2.sty v2.x`.

```
150 \newcommand\setcaptionwidth{%
151   \setlength\captionwidth{<amount>}}
```

### Indentions

`\captionindent` `\captionindent` These are the indentions we support.

`\captionparindent` `\captionparindent`

`\captionhangindent` `\captionhangindent`

```
152 \newdimen\captionindent
153 \newdimen\captionparindent
154 \newdimen\captionhangindent

155 \DeclareCaptionOption{indent}{\leftmargin[i]{\setlength\captionindent{#1}}}% obsolete
156 \DeclareCaptionOption{indention}{\leftmargin[i]{\setlength\captionindent{#1}}}
157 \DeclareCaptionOption{hangindent}{\setlength\captionhangindent{#1}}
158 \DeclareCaptionOption{parindent}{\setlength\captionparindent{#1}}
159 \DeclareCaptionOption{parskip}{\l@addto@macro\caption@par{\setlength\parskip{#1}}}
160 \@ifundefined{scr@caption}{}{%
```

There is an option clash between the KOMA-Script document classes and the caption kernel, both define the options `parindent` and `parskip` but with different meaning. Furthermore the ones defined by the caption kernel take a value as parameter but the



KOMA-Script ones do not. So we need special versions of the options `parindent` and `parskip` here, ones who determine if a value is given (and therefore should be treated as our option) or not (and therefore should be ignored by us).

```

161 \let\caption@KV@parindent\KV@caption@parindent
162 \DeclareCaptionOption{parindent}[]{%
163   \def\caption@tempa{#1}%
164   \ifx\caption@tempa\@empty
165     \PackageInfo{caption3}{Option 'parindent' ignored}%
166   \else
167     \caption@KV@parindent{#1}%
168   \fi}%
169 \let\caption@KV@parskip\KV@caption@parskip
170 \DeclareCaptionOption{parskip}[]{%
171   \def\caption@tempa{#1}%
172   \ifx\caption@tempa\@empty
173     \PackageInfo{caption3}{Option 'parskip' ignored}%
174   \else
175     \caption@KV@parskip{#1}%
176   \fi}%
177 }

```

## Styles

```

\DeclareCaptionStyle \DeclareCaptionStyle{<name>}[<single-line-list-of-KV>]{<list-of-KV>}
178 \newcommand*\DeclareCaptionStyle[1]{%
179   \@ifnextchar[{\caption@declarestyle{#1}}{\caption@declarestyle{#1}[]}}
180 \def\caption@declarestyle#1[#2]#3{%
181   \global\@namedef{caption@sls@#1}{#2}%
182   \global\@namedef{caption@sty@#1}{#3}}
183 \@onlypreamble\DeclareCaptionStyle
184 \@onlypreamble\caption@declarestyle
185 \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.

```

186 \newcommand\caption@setstyle{%
187   \@ifstar{\caption@@setstyle\@gobble}{\caption@@setstyle\@firstofone}}
188 \newcommand*\caption@@setstyle[2]{%
189   \@ifundefined{caption@sty@#2}%
190     {#1{\caption@error{Undefined caption style '#2'}}}%
191     {\expandafter\let\expandafter\caption@sls\csname caption@sls@#2\endcsname
192      \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!*

```
193 \newcommand\caption@setdefault{\captionsetup{%
194   format=default,labelformat=default,labelsep=default,textformat=default,%
195   justification=default,font=default,labelfont=default,textfont=default,%
196   margin=0pt,indent=0pt,parindent=0pt,hangindent=0pt,%
197   singlelinecheck=1,strut=1}}
```

Currently there is only one pre-defined style, called ‘default’. It’s a perfect match to the behaviour of `\@makecaption` offered by the standard L<sup>A</sup>T<sub>E</sub>X document classes: If the caption fits in one single line, it is typeset centered.

```
198 \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 `indent=` option.

```
199 \newcommand\DeclareCaptionFormat{%
200   \@ifstar{\caption@declareformat\@gobble}%
201   {\caption@declareformat\@firstofone}%
202 \newcommand\caption@declareformat[3]{%
203   \global\expandafter\let\csname caption@ifh@#2\endcsname#1%
204   \global\long\@namedef{caption@fmt@#2}##1##2##3{#3}}
205 \@onlypreamble\DeclareCaptionFormat
206 \@onlypreamble\caption@declareformat
```

```
207 \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`).

```
208 \newcommand*\caption@setformat[1]{%
209   \@ifundefined{caption@fmt@#1}%
210   {\caption@error{Undefined caption format `#1'}}%
211   {\expandafter\let\expandafter\caption@ifh\csname caption@ifh@#1\endcsname
212   \expandafter\let\expandafter\caption@fmt\csname caption@fmt@#1\endcsname}}
```

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

```
213 \DeclareCaptionFormat{plain}{#1#2#3\par}
214 \DeclareCaptionFormat{hang}{%
215   \@hangfrom{#1#2}%
216   \advance\captionparindent\hangindent
217   \advance\captionhangindent\hangindent
218   \caption@@par
219   #3\par}
```

‘default’ usually maps to ‘plain’.

```
220 \def\caption@fmt@default{\caption@fmt@plain}
221 \def\caption@ifh@default{\caption@ifh@plain}
```

### Label formats

```
\DeclareCaptionLabelFormat \DeclareCaptionLabelFormat{<name>}{<code with #1 and #2>}
222 \newcommand*\DeclareCaptionLabelFormat[2]{%
223   \global\@namedef{caption@lfmt@#1}##1##2{#2}}
224 \@onlypreamble\DeclareCaptionLabelFormat

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

\caption@setlabelformat \caption@setlabelformat{<name>}
Selecting a caption label format simply means saving the code (in \caption@lfmt).
226 \newcommand*\caption@setlabelformat[1]{%
227   \@ifundefined{caption@lfmt@#1}%
228     {\caption@error{Undefined caption label format `#1'}}%
229     {\expandafter\let\expandafter\caption@lfmt\csname caption@lfmt@#1\endcsname}}
```

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

```
230 \DeclareCaptionLabelFormat{empty}{}
231 \DeclareCaptionLabelFormat{simple}{\bothIfFirst{#1}{\nobreakspace}#2}
232 \DeclareCaptionLabelFormat{parens}{\bothIfFirst{#1}{\nobreakspace}{#2}}

‘default’ usually maps to ‘simple’.
233 \def\caption@lfmt@default{\caption@lfmt@simple}
```

### Label separators

```
\DeclareCaptionLabelSeparator \DeclareCaptionLabelSeparator{<name>}{<code>}
\DeclareCaptionLabelSeparator*{<name>}{<code>}
The starred form causes the label separator to be typeset without using \captionlabelfont.
234 \newcommand\DeclareCaptionLabelSeparator{%
235   \@ifstar{\caption@declarelabelseparator\@gobble}%
236           {\caption@declarelabelseparator\@firstofone}}
237 \newcommand\caption@declarelabelseparator[3]{%
238   \global\expandafter\let\csname caption@ifl@#2\endcsname#1%
239   \global\long\@namedef{caption@lsep@#2}{#3}}
240 \@onlypreamble\DeclareCaptionLabelSeparator
241 \@onlypreamble\caption@declarelabelseparator

242 \DeclareCaptionOption{labelsep}{\caption@setlabelseparator{#1}}
243 \DeclareCaptionOption{labelseparator}{\caption@setlabelseparator{#1}}

\caption@setlabelseparator \caption@setlabelseparator{<name>}
Selecting a caption label separator simply means saving the code (in \caption@lsep).
244 \newcommand*\caption@setlabelseparator[1]{%
```

```

245 \@ifundefined{caption@lsep@#1}%
246     {\caption@error{Undefined caption label separator `#1'}}%
247     {\expandafter\let\expandafter\caption@iflfl\csname caption@iflfl@#1\endcsname
248         \expandafter\let\expandafter\caption@lsep\csname caption@lsep@#1\endcsname}}

```

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

```

249 \DeclareCaptionLabelSeparator{none}{}
250 \DeclareCaptionLabelSeparator{colon}{: }
251 \DeclareCaptionLabelSeparator{period}{. }
252 \DeclareCaptionLabelSeparator{space}{ }
253 \DeclareCaptionLabelSeparator*{quad}{\quad}
254 \DeclareCaptionLabelSeparator*{newline}{\\}
255 \DeclareCaptionLabelSeparator*{endash}{\space\textendash\space}

```

‘default’ usually maps to ‘colon’.

```

256 \def\caption@lsep@default{\caption@lsep@colon}
257 \def\caption@iflfl@default{\caption@iflfl@colon}

```

### Text formats

```

\DeclareCaptionTextFormat \DeclareCaptionTextFormat{<name>}{<code with #1>}
258 \newcommand*\DeclareCaptionTextFormat[2]{%
259     \global\long\@namedef{caption@tfmt@#1}##1{#2}}
260 \@onlypreamble\DeclareCaptionTextFormat

261 \DeclareCaptionOption{textformat}{\caption@settextformat{#1}}

```

```

\caption@settextformat \caption@settextformat{<name>}

```

Selecting a caption text format simply means saving the code (in \caption@tfmt).

```

262 \newcommand*\caption@settextformat[1]{%
263     \@ifundefined{caption@tfmt@#1}%
264     {\caption@error{Undefined caption text format `#1'}}%
265     {\expandafter\let\expandafter\caption@tfmt\csname caption@tfmt@#1\endcsname}}

```

There are two pre-defined text formats, called ‘simple’ and ‘period’.

```

266 \DeclareCaptionTextFormat{simple}{#1}
267 \DeclareCaptionTextFormat{period}{#1.}

```

‘default’ usually maps to ‘simple’.

```

268 \def\caption@tfmt@default{\caption@tfmt@simple}

```

### Justifications

```

\DeclareCaptionJustification \DeclareCaptionJustification{<name>}{<code>}
269 \newcommand*\DeclareCaptionJustification[2]{%
270     \global\@namedef{caption@hj@#1}{#2}}
271 %\newcommand\DeclareCaptionJustification{\DeclareCaptionFont}
272 \@onlypreamble\DeclareCaptionJustification

```

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

```
\caption@setjustification \caption@setjustification{<name>}
```

Selecting a caption justification simply means saving the code (in \caption@hj).

```
274 \newcommand*\caption@setjustification[1]{%
275   \@ifundefined{caption@hj@#1}%
276     {\caption@error{Undefined caption justification `#1'}}%
277     {\expandafter\let\expandafter\caption@hj\csname caption@hj@#1\endcsname}}
278 %\newcommand\caption@setjustification{\caption@setfont{@hj}}
```

These are the pre-defined justification code snippets.

```
279 \DeclareCaptionJustification{justified}{}
280 \DeclareCaptionJustification{centering}{\centering}
281 \DeclareCaptionJustification{centerfirst}{\centerfirst}
282 \DeclareCaptionJustification{centerlast}{\centerlast}
283 \DeclareCaptionJustification{raggedleft}{\raggedleft}
284 \DeclareCaptionJustification{raggedright}{\raggedright}
```

‘default’ usually maps to ‘justified’.

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

```
\centerfirst Please blame Frank Mittelbach for \caption@centerfirst :-)
```

```
286 \providecommand\centerfirst{%
287   \let\\\@centercr
288   \edef\caption@normaladjust{%
289     \leftskip\the\leftskip
290     \rightskip\the\rightskip
291     \parfillskip\the\parfillskip\relax}%
292   \leftskip\z@\@plus -1fil%
293   \rightskip\z@\@plus 1fil%
294   \parfillskip\z@skip
295   \noindent\hskip\z@\@plus 2fil%
296   \@setpar{\@par\@restorepar\caption@normaladjust}}
```

```
\centerlast This is based on code from Anne Brüggemann-Klein[12]
```

```
297 \providecommand\centerlast{%
298   \let\\\@centercr
299   \leftskip\z@\@plus 1fil%
300   \rightskip\z@\@plus -1fil%
301   \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.

```
302 \DeclareCaptionJustification{Centering}{%
303   \caption@ragged\Centering\centering}
304 \DeclareCaptionJustification{RaggedLeft}{%
305   \caption@ragged\RaggedLeft\raggedleft}
306 \DeclareCaptionJustification{RaggedRight}{%
307   \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 an additional warning if the ragged2e package is not avail. (This warning will be
typeout only one time per option, that's why we need the caption\string#1 stuff.)
308 \newcommand*\caption@ragged[2]{%
309   \@ifundefined{caption\string#1}{%
310     \PackageWarning{caption}{%
311       Cannot locate the 'ragged2e' package, therefore\MessageBreak
312       substituting \string#2 for \string#1\MessageBreak}%
313     \global\@namedef{caption\string#1}}}%
314   #2}

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

```

## Fonts

```

\DeclareCaptionFont \DeclareCaptionFont{<name>}{<code>}
317 \newcommand\DeclareCaptionFont[2]{%
318   \define@key{caption@fnt}{#1}[]{\g@addto@macro\caption@tempa{#2}}%
319   \@onlypreamble\DeclareCaptionFont

320 \DeclareCaptionOption{font}{\caption@setfont{font}{#1}}
321 \DeclareCaptionOption{labelfont}{\caption@setfont{labelfont}{#1}}
322 \DeclareCaptionOption{textfont}{\caption@setfont{textfont}{#1}}

```

`\caption@setfont` `\caption@setfont{<name>}{<keyval-list of names>}`  
 Selecting a caption font means saving all the code snippets (in `\caption#1`). Because we use `\setkeys` recursive here we need to do this inside an extra group and collect all the code snippets in `\caption@tempa` first.

```

323 \newcommand*\caption@setfont[2]{%
324   \let\caption@tempa\@empty
325   \begingroup
326   % \define@key{caption@fnt}{default}[]{%
327   %   \global\expandafter\let\expandafter\caption@tempa
328   %   \csname caption#1@default\endcsname}%
329   \caption@setkeys[caption]{caption@fnt}{#2}%
330   \endgroup
331   \expandafter\let\csname caption#1\endcsname\caption@tempa}

332 \DeclareCaptionFont{default}{}

```

These are the pre-defined font code snippets.

```

333 \DeclareCaptionFont{scriptsize}{\scriptsize}
334 \DeclareCaptionFont{footnotesize}{\footnotesize}
335 \DeclareCaptionFont{small}{\small}
336 \DeclareCaptionFont{normalsize}{\normalsize}

```

```

337 \DeclareCaptionFont{large}{\large}
338 \DeclareCaptionFont{Large}{\Large}

339 \DeclareCaptionFont{up}{\upshape}
340 \DeclareCaptionFont{it}{\itshape}
341 \DeclareCaptionFont{sl}{\slshape}
342 \DeclareCaptionFont{sc}{\scshape}
343 \DeclareCaptionFont{md}{\mdseries}
344 \DeclareCaptionFont{bf}{\bfseries}
345 \DeclareCaptionFont{rm}{\rmfamily}
346 \DeclareCaptionFont{sf}{\sffamily}
347 \DeclareCaptionFont{tt}{\ttfamily}

```

`\captionsize` The old versions *v1.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.

```

348 \providecommand\captionsize{}

349 \DeclareCaptionOption{size}{\caption@setfont{size}{#1}}

```

### Vertical spaces before and after captions

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

```

350 \@ifundefined{abovecaptionskip}{%
351   \newlength\abovecaptionskip\setlength\abovecaptionskip{10\p@}}{}
352 \@ifundefined{belowcaptionskip}{%
353   \newlength\belowcaptionskip\setlength\belowcaptionskip{0\p@}}{}

354 \DeclareCaptionOption{aboveskip}{\setlength\abovecaptionskip{#1}}
355 \DeclareCaptionOption{belowskip}{\setlength\belowcaptionskip{#1}}
356 \DeclareCaptionOption{skip}{\setlength\abovecaptionskip{#1}}

```

### 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 a specific package like the `float` package) instead. The user can put the `\caption` command wherever 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.)

```

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

```

`\caption@setposition` `\caption@setposition{position}`  
 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.*

```

358 \newcommand*\caption@setposition[1]{%
359   \caption@ifinlist{#1}{d,default}{%
360     \def\caption@position{\caption@defaultpos}%
361   }{\caption@ifinlist{#1}{t,top,above}{%
362     \let\caption@position\@firstoftwo
363   }{\caption@ifinlist{#1}{b,bottom,below}{%
364     \let\caption@position\@secondoftwo
365   }{\caption@ifinlist{#1}{a,auto}{%
366     \let\caption@position\@undefined
367   }{%
368     \caption@error{Undefined caption position `#1'}%
369   }}}}%

```

`\caption@defaultpos` The default ‘position’ is usually ‘bottom’, this means that the (larger) skip will be typeset above the caption. This corresponds to the `\@makecaption` implementation in the standard L<sup>A</sup>T<sub>E</sub>X document classes.

```

370 %\caption@setdefaultpos{b}% default = bottom
371 \let\caption@defaultpos\@secondoftwo

```

`\caption@iftop` `\caption@iftop{<true-code>}{<false-code>}`  
 (If the position= is set to auto we assume a bottom position.)

```

372 \newcommand\caption@iftop{%
373   \ifx\caption@position\@undefined
374     \expandafter\@secondoftwo
375   \else
376     \expandafter\caption@position
377   \fi}

```

`\caption@fixposition` `\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.

```

378 \newcommand\caption@fixposition{%
379   \ifx\caption@position\@undefined
380     \caption@autoposition
381   \fi}

```

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

```

382 \newcommand\caption@autoposition{%
383   \ifvmode
384     \edef\caption@tempa{\the\prevdepth}%
385     \caption@Debug{\protect\prevdepth=\caption@tempa}%
386 %   \caption@setposition{\ifdim\prevdepth>-\p@ b\else t\fi}%
387     \ifdim\prevdepth>-\p@
388       \let\caption@position\@secondoftwo
389     \else
390       \let\caption@position\@firstoftwo
391     \fi
392   \else

```



```

393 \caption@Debug{no \protect\prevdepth}%
394 % \caption@setposition{b}%
395 \let\caption@position\@secondoftwo
396 \fi}

```

## Hooks

```

\AtBeginCaption \AtBeginCaption {<code>}
\AtEndCaption \AtEndCaption {<code>}

```

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

```

397 \newcommand\caption@beginhook{}
398 \newcommand\caption@endhook{}
399 \newcommand\AtBeginCaption{\l@addto@macro\caption@beginhook}
400 \newcommand\AtEndCaption{\l@addto@macro\caption@endhook}

```

## Miscellaneous options

```

401 \DeclareCaptionOption{listof}{\caption@setbool{lof}{#1}}
402 \DeclareCaptionOption{singlelinecheck}{\caption@setbool{slc}{#1}}
403 \DeclareCaptionOption{strut}{\caption@setbool{strut}{#1}}

```

## Debug option

```

404 \DeclareCaptionOption{debug}[1]{%
405 \caption@setbool{debug}{#1}%
406 \caption@ifbool{debug}{%
407 \def\caption@Debug{\PackageInfo{caption3}}%
408 }{%
409 \let\caption@Debug\@gobble
410 }}
411 \setkeys{caption}{debug=0}

```

## Initialization of parameters

```

412 \captionsetup{style=default,position=default,listof=1}

```

```

\ifcaption@star If the starred form of \caption is used, this will be set to true. (It will be reset to
false at the end of \caption@@make.)
413 \newif\ifcaption@star

```

## Typesetting the caption

```

\caption@make \caption@make{<float name>}{<ref. number>}{<text>}
414 \newcommand\caption@make[2]{%
415 \caption@@make{\caption@lfmt{#1}{#2}}

```

```

\caption@@make \caption@@make{<caption label>}{<caption text>}
416 \newcommand\caption@@make[2]{%
417 \begingroup
418 \caption@beginhook
419 \caption@calcmargin

```

Special single-line treatment (option singlelinecheck=)

```
420 \caption@ifslc{\ifx\caption@sls\@empty\else
421   \caption@slc{#1}{#2}\captionwidth\relax
422 \fi}{}%
```

Typeset the left margin (option margin=)

```
423 \@tempdima\captionmargin
424 \caption@ifh{\advance\@tempdima by \captionindent}%
425 \hskip\@tempdima
```

We actually use a `\vbox` of width `\captionwidth - \captionindent` to typeset the caption (Note: `\captionindent` is *not* supported if the caption format was defined with `\DeclareCaptionFormat*`.)

```
426 \@tempdima\captionwidth
427 \caption@ifh{\advance\@tempdima by -\captionindent}%
428 \captionbox\@tempdima{%
```

Typeset the indentation (option indentation=)

Bugfix 04-05-05: `\hskip-\captionindent` replaced by `\ifdim\captionindent=\z@...`

```
429 \caption@ifh{%
430   \ifdim\captionindent=\z@
431     \leavevmode
432   \else
433     \hskip-\captionindent
434   \fi}%
```

Typeset the caption itself and close the `\captionbox`

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

Typeset the right margin (option margin=)

```
436 \@tempdima\captionmargin
437 \advance\@tempdima by \captionmarginx
438 \hskip\@tempdima
439 \caption@endhook
440 \endgroup
441 \global\caption@starfalse}
```

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

```
442 \newcommand\caption@calcmargin{%
```

*Note:* Inside a list environment `\linewidth` do not contain the proper value, because `\caption` calls `\@parboxrestore` which resets `\linewidth` to `\hsize`. Therefore we have to calculate the proper line width on our own in this case.

```
443 \@tempdima\hsize
444 \ifnum\@listdepth>0\relax
445   \advance\@tempdima by -\leftmargin
446   \advance\@tempdima by -\rightmargin
447 \fi
```

```

448 \ifdim\captionwidth=\z@
449   \captionwidth\@tempdima
450   \advance\captionwidth by -2\captionmargin
451   \advance\captionwidth by -\captionmarginx
452 \else
453   \captionmargin\@tempdima
454   \advance\captionmargin by -\captionwidth
455   \divide\captionmargin by 2
456   \captionmarginx\z@
457 \fi

458 \caption@Debug{%
459   \protect\hsize=\the\hsize,
460   \protect\margin=\the\captionmargin,
461   \protect\marginx=\the\captionmarginx,
462   \protect\width=\the\captionwidth}%
463 }

\caption@slc  This one does the single-line-check.
464 \newcommand\caption@slc[4]{%
465   \caption@startslc
466   \sbox\@tempboxa{\caption@@@make{#1}{#2}}%
467   \ifdim\wd\@tempboxa >#3%
468     \caption@endslc
469   \else
470     \caption@endslc
471     \caption@esetup\caption@sls
472     #4%
473   \fi}

\caption@startslc  Re-define anything which would disturb the single-line-check.
474 \newcommand\caption@startslc{%
475   \begingroup
476   \let\label\@gobble
477   \let\@footnotetext\@gobble\let\@endnotetext\@gobble
478   \def\stepcounter##1{\advance\c@##1\endcsname\@ne\relax}%
479   \let\caption@hj\relax}

\caption@endslc  This ends the single-line-check.
480 \newcommand\caption@endslc{%
481   \endgroup}

\captionbox  This macro defines the box which surrounds the caption paragraph.
482 \newcommand\captionbox{\parbox[b]{}

\caption@@@make  \caption@@@make{\caption label}{\caption text}
This one finally typesets the caption paragraph, without margin and indentation.
483 \newcommand\caption@@@make[2]{%
```

If the label is empty, we use no caption label separator.

```
484 \sbox\@tempboxa{#1}%
485 \ifdim\wd\@tempboxa=\z@
486 \let\caption@lsep\relax
487 \fi
```

If the text is empty, we use no caption label separator, too.

```
488 \caption@ifempty{#2}{%
489 \let\caption@lsep\relax
490 \let\caption@ifstrut\@secondoftwo
491 }%
```

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

```
492 \@setpar{\@@par\caption@@par}\caption@@par
```

Finally the caption will be typeset.

```
493 \caption@hj\captionsize\captionfont\caption@fmt
494 {\ifcaption@star\else{\captionlabelfont#1}\fi}%
495 {\ifcaption@star\else{\caption@iflf\captionlabelfont\caption@lsep}\fi}%
496 {\captiontextfont
497 \caption@ifstrut{\vrule\@height\ht\strutbox\@width\z@}{}}%
498 \nobreak\hskip\z@skip
499 \caption@tfmt{#2}%
500 \caption@ifstrut{\vrule\@height\z@\@depth\dp\strutbox\@width\z@}{}}%
501 \caption@ifstrut{\@finalstrut\strutbox}{}}%
502 \par}}}
```

```
\caption@ifempty \caption@ifempty{<text>}{<if-clause>}
```

This one tests if the *<text>* is actually empty.

*Note:* This will be done without expanding the text, therefore this is far away from being bullet-proof.

```
503 \newcommand\caption@ifempty[1]{%
504 \def\caption@tempa{#1}%
505 \def\caption@tempb{\ignorespaces}%
506 \ifx\caption@tempa\caption@tempb
507 \let\caption@tempa\@empty
508 \fi
509 \ifx\caption@tempa\@empty
510 \expandafter\@firstofone
511 \else
512 \expandafter\@gobble
513 \fi}
```

```
\caption@@par \caption@@par
```

This command will be executed with every `\par` inside the caption.

```
514 \newcommand*\caption@@par{%
515 \parindent\captionparindent\hangindent\captionhangindent}%

```

## subfig package support

This is a very small bugfix for v1.2 and v1.3 of the subfig package, making `\subfig` robust. I do this here because it's caption related stuff and I get all the bug reports ; -)

```
516 \AtBeginDocument{%
517   \def\@tempa{\@ifstar\sf@@subref\sf@subref}%
518   \ifx\subref\@tempa
519     \PackageInfo{caption3}{subref 1.2 or 1.3 detected}%
520     \DeclareRobustCommand* \subref{\@ifstar\sf@@subref\sf@subref}%
521   \fi
522 }
```

## 11.2 Main package

### Identification

```
523 \NeedsTeXFormat{LaTeX2e}[1994/12/01]
524 \ProvidesPackage{caption}[2007/03/09 v3.0o Customising captions (AR)]
```

### Loading the caption kernel

```
525 \RequirePackage{caption3}[2007/01/31] % needs v3.01 or newer
```

### Option for configuration files

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

```

### Options for figure and table

```
531 \DeclareCaptionOption*{figureposition}{\captionsetup[figure]{position=#1}}
532 \DeclareCaptionOption*{tableposition}{\captionsetup[table]{position=#1}}
533 \DeclareCaptionOption*{figurename}{\captionsetup[figure]{name=#1}}
534 \DeclareCaptionOption*{tablename}{\captionsetup[table]{name=#1}}

```

### caption v1.x compatibility options

```
535 \DeclareCaptionOption*{normal}[]{\caption@setformat{normal}}
536 \DeclareCaptionOption*{isu}[]{\caption@setformat{hang}}
537 \DeclareCaptionOption*{hang}[]{\caption@setformat{hang}}
538 \DeclareCaptionOption*{center}[]{\caption@setjustification{centering}}
539 \DeclareCaptionOption*{anne}[]{\caption@setjustification{centerlast}}
540 \DeclareCaptionOption*{centerlast}[]{\caption@setjustification{centerlast}}

541 \DeclareCaptionOption*{scriptsize}[]{\def\captionfont{\scriptsize}}
542 \DeclareCaptionOption*{footnotesize}[]{\def\captionfont{\footnotesize}}
543 \DeclareCaptionOption*{small}[]{\def\captionfont{\small}}
544 \DeclareCaptionOption*{normalsize}[]{\def\captionfont{\normalsize}}
545 \DeclareCaptionOption*{large}[]{\def\captionfont{\large}}
546 \DeclareCaptionOption*{Large}[]{\def\captionfont{\Large}}

```

```

547 \DeclareCaptionOption*{up}[]{\l@addto@macro\captionlabelfont\upshape}
548 \DeclareCaptionOption*{it}[]{\l@addto@macro\captionlabelfont\itshape}
549 \DeclareCaptionOption*{sl}[]{\l@addto@macro\captionlabelfont\slshape}
550 \DeclareCaptionOption*{sc}[]{\l@addto@macro\captionlabelfont\scshape}
551 \DeclareCaptionOption*{md}[]{\l@addto@macro\captionlabelfont\mdseries}
552 \DeclareCaptionOption*{bf}[]{\l@addto@macro\captionlabelfont\bfseries}
553 \DeclareCaptionOption*{rm}[]{\l@addto@macro\captionlabelfont\rmfamily}
554 \DeclareCaptionOption*{sf}[]{\l@addto@macro\captionlabelfont\sffamily}
555 \DeclareCaptionOption*{tt}[]{\l@addto@macro\captionlabelfont\ttfamily}

556 \DeclareCaptionOption*{nooneline}[]{\caption@setbool{slc}{0}}
557 \caption@setbool{ruled}{0}
558 \DeclareCaptionOption*{ruled}[]{\caption@setbool{ruled}{1}}

```

### Some caption2 v2.x compatibility options

```

559 \DeclareCaptionOption*{flushleft}[]{\caption@setjustification{raggedright}}
560 \DeclareCaptionOption*{flushright}[]{\caption@setjustification{raggedleft}}
561 \DeclareCaptionOption*{oneline}[]{\caption@setbool{slc}{1}}
562 \DeclareCaptionOption*{ignoreLTcapwidth}[]{}

```

### Some KOMA-Script compatibility options

```

563 \DeclareCaptionOption*{onelinecaption}[]{\caption@setbool{slc}{1}}
564 \DeclareCaptionOption*{noonelinecaption}[]{\caption@setbool{slc}{0}}
565 \DeclareCaptionOption*{tablecaptionabove}[]{\captionsetup[table]{position=t}}
566 \DeclareCaptionOption*{tablecaptionbelow}[]{\captionsetup[table]{position=b}}

```

### Generic package support

`\caption@declarepackage`    `\caption@declarepackage{<package name>}`

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

```

567 \newcommand*\caption@declarepackage[1]{%
568   \caption@setbool{pkt@#1}{1}%
569   \DeclareCaptionOption*{#1}{\caption@setbool{pkt@#1}{##1}}
570 \AtEndOfPackage{\let\caption@declarepackage\@undefined}

```

`\caption@ifpackage`    `\caption@ifpackage{<package name>}{<package macro>}{<code>}`

If a certain package support is requested the appropriate code will be used. ‘Requested’ means that the option belonging to it is set to true and the macro called *<package macro>* is defined. (If *<package macro>* is not yet defined we use `\AtBeginDocument` here, so the package could be loaded after this package, too.)

```

571 \newcommand\caption@ifpackage[3]{%
572   \edef\caption@tempa{%
573     \caption@ifbool{pkt@#1}%
574     {\@ifundefined{#2}{AtBeginDocument}{firstofone}}%
575     {gobble}}%
576   \caption@Info{#1=\caption@ifbool{pkt@#1}{1}{0} %
577     (\@ifundefined{#2}{not }{}loaded->\caption@tempa)}%

```

```

578 \caption@ifbool{pkt@#1}{%
579   \@ifundefined{#2}%
580   {\let\caption@tempa\AtBeginDocument}%
581   {\let\caption@tempa\@firstofone}%
582 }{%
583   \let\caption@tempa\@gobble
584 }%
585 \caption@tempa{\@ifundefined{#2}{}{#3}}%
586 \caption@undefbool{pkt@#1}
587 \AtEndOfPackage{\let\caption@ifpackage\@undefined}

```

You can also switch the caption support off using the package option `caption=false`. This may look strange, but there are certain circumstances where this could be useful. Such a situation might be the usage of the `subfig` package without disturbing the main caption code of the document class.

*Note: This mechanism is obsolete now, it has been superseded by the `subfig` package option `caption=false` which causes that only the caption kernel `caption3` is loaded.*

```

588 \caption@declarepackage{caption}

```

These are the packages we support:

```

589 \caption@declarepackage{float}
590 \caption@declarepackage{floatrow}
591 \caption@declarepackage{hyperref}
592 \caption@declarepackage{hypcap}
593 \caption@declarepackage{listings}
594 \caption@declarepackage{longtable}
595 \caption@declarepackage{picins}
596 \caption@declarepackage{rotating}
597 \caption@declarepackage{sidecap}
598 \caption@declarepackage{supertabular}

```

## Processing of options

```

599 \DeclareCaptionOption{verbose}[1]{%
600   \caption@setbool{verbose}{#1}%
601   \caption@ifbool{verbose}{%
602     \def\caption@Info{\PackageInfo{caption}}%
603   }{%
604     \let\caption@Info\@gobble
605   }
606   \setkeys{caption}{verbose=0}
607 \caption@ProcessOptions{caption}

```

If the option `caption=false` was given we stop processing this file immediately.

```

608 \caption@ifbool{pkt@caption}{}{\endinput}
609 \caption@undefbool{pkt@caption}

```

## Useful stuff

```

\captionof \captionof(*){<type>}[<lst_entry>]{<heading>}

```

```

610 \def\captionof{\@ifstar{\caption@of{\caption*}}{\caption@of\caption}}
611 \newcommand*\caption@of[2]{\def\@capttype{#2}#1}

```

**Note:** Like `\captionof` the option `type=` should only be used inside a group or environment and does not check if the argument is a valid floating environment or not.

```

612 \DeclareCaptionOption{name}{\caption@setfloatname\@capttype{#1}}
613 \DeclareCaptionOption{type}{\def\@capttype{#1}}

```

`\ContinuedFloat`    `\ContinuedFloat[<type>]`

This mainly decreases the appropriate counter by  $-1$ .

```

614 \providecommand\ContinuedFloat{%
615   \@ifnextchar[%
616     \@ContinuedFloat
617     {\ifx\@capttype\@undefined
618       \@latex@error{\noexpand\ContinuedFloat outside float}\@ehd
619     }
620     \@ContinuedFloat[\@capttype]%
621   \fi}}
622 \def\@ContinuedFloat[#1]{%
623   \addtocounter{#1}\m@ne
624   \caption@ContinuedFloat{#1}%
625   \caption@@ContinuedFloat{#1}}

```

`\caption@ContinuedFloat`    `\caption@ContinuedFloat{<type>}`  
`\caption@resetContinuedFloat`    `\caption@resetContinuedFloat{<type>}`

The first one will be called inside `\ContinuedFloat`, the second one inside `\caption`. Usually they do nothing but this changes if the `hyperref` package is loaded. (See `hyperref` package support for details.)

```

626 \let\caption@ContinuedFloat\@gobble
627 \let\caption@resetContinuedFloat\@gobble

```

`\caption@@ContinuedFloat`    This hook is for foreign packages which link themselves into `\ContinuedFloat`, for example the `subfig` package[10].

```

628 \providecommand*\caption@@ContinuedFloat[1]{}

```

### Internal helpers

`\caption@begin`    Our handling of `\caption` will always be surrounded by `\caption@begin` (or `\caption@beginex`) and `\caption@end`.  
`\caption@begin{<type>}` performs these tasks:

- Call `\caption@resetContinuedFloat` (see above) and start a new group
- Execute the options set with `\captionsetup[<type>]`
- Define `\fnum@<type>` if the caption label format is set to non-default
- Override the `position=` setting, if necessary (for example if set to `auto` or used inside a `supertabular`)



```

629 \newcommand*\caption@begin[1]{%
630   \caption@resetContinuedFloat{#1}%
631   \begingroup
632   \caption@setfloattype{#1}%
633   \ifx\caption@lfmt\caption@lfmt@default\else
634     \@namedef{fnum{#1}}{%
635       \caption@lfmt{\caption@floatname{#1}}{\@nameuse{the#1}}}%
636   \fi
637   \caption@fixposition
638   \global\let\caption@fixedposition\caption@position}

\caption@beginex \caption@beginex{<type>}{<list entry>}{<heading>}
performs the same tasks as \caption@begin and additionally: Redefine \addcontentsline
if no list-of entry is requested, that means either the argument <list entry> is empty or
listof= was set to false.
639 \newcommand\caption@beginex[3]{%
640   \caption@begin{#1}%
641   \caption@iflof%
642     {\def\caption@tempa{#2}}%
643     {\let\caption@tempa\@empty}%
644   \ifx\caption@tempa\@empty
645     \long\def\addcontentsline##1##2##3{% There is no \@gobblethree...
646   \fi
647   \caption@ifempty{#3}{\let\caption@ifempty\@secondoftwo}%
648 }

\caption@end \caption@end closes the group.
649 \newcommand*\caption@end{%
650   \endgroup
651   \let\caption@position\caption@fixedposition}

\caption@setfloattype \caption@setfloattype{<type>}
sets 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 or sidecap package is loaded.
652 \let\caption@setfloattype\caption@settype

\caption@letfloattype \caption@letfloattype{<type>}{<extra code>}
redefines \caption@setfloattype so it does not only \caption@settype{<type>}
but two additional tasks: Executing extra code given as second argument and execute op-
tions with \caption@settype{#1} afterwards.
You can find an example of its usage in the longtable support, where this macro is
called so \captionsetup[longtable]{...} can be used to setup options for
longtables which have a higher priority than the options which have been setup with
\captionsetup[table]{...} or \setlength\LTcapwidth{...}.

```

```

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

```

```
\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 changed later on.

```

656 \newcommand*\caption@floatname[1]{\@nameuse{#1name}}
657 \newcommand*\caption@setfloatname[1]{\@namedef{#1name}}%

```

### Caption support

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

```
658 \AtBeginDocument{%
```

We only patch `\caption` and `\@caption` if the `captcont` package (which brings it's own definition of `\caption*`) is not used. It does not make much sense using the actual version of the `caption` package with the `captcont` package, but this was different in the old (`v1.x`) days so we take care to be backward compatible.

```

659   \@ifundefined{cc@caption}{%
660     \@ifundefined{caption@old}{%

```

```
\caption Here comes our definition of \caption and \caption*. (We set \caption@startrue globally so it works with the sidecap package, too.)
```

```

661   \let\caption@old\caption
662   \def\caption{\caption@caption\caption@old}%
663   \def\caption@caption#1{%
664     \ifstar{\ContinuedFloat\global\caption@startrue#1[]}{#1}}%

```

```
\@caption Our definition of \@caption simply calls the old definition, nested by \caption@beginex and \caption@end.
```

```

665   \let\caption@@old\@caption
666   \long\def\@caption#1[#2]#3{%
667     \caption@beginex{#1}{#2}{#3}%
668     \caption@@old{#1}[{#2}]{#3}%
669     \caption@end}%

```

```
670 }{}{}{}%
```

Minimum `captcont` package support:

We define `\caption@caption` here so it's there but does not make any harm.

```

671   \caption@Info{captcont package v2.0 detected}%
672   \def\caption@caption#1{#1}%
673   }%
674 }

```

```
\@makecaption \@makecaption{\langle label \rangle}{\langle text \rangle}
```

The 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}
```

We do basically the same, but take care of the position= setting and use \caption@@make from the caption kernel to actually typeset the caption.

```
675 \renewcommand\@makecaption[2]{%
676   \caption@iftop{\vskip\belowcaptionskip}{\vskip\abovecaptionskip}%
677   \caption@@make{#1}{#2}%
678   \caption@iftop{\vskip\abovecaptionskip}{\vskip\belowcaptionskip}}
```

### KOMA-Script classes support

```
679 \ifundefined{scr@caption}{}{%
680   \caption@Info{KOMA-Script class detected}
```

```
\onelinecaptionsfalse
\onelinecaptionstrue
681 \def\onelinecaptionstrue{\caption@setbool{slc}{1}}
682 \def\onelinecaptionsfalse{\caption@setbool{slc}{0}}
683 \ifonelinecaptions
684   \onelinecaptionstrue
685 \else
686   \onelinecaptionsfalse
687 \fi
```

\captionabove Original code:

```
\captionbelow
\newcommand{\captionabove}{\@captionabovetrue\scr@caption}
\newcommand{\captionbelow}{\@captionabovefalse\scr@caption}
```

```
688 \AtBeginDocument{\let\scr@caption\caption}
689 \def\captionabove{%
690   \caption@setposition{t}\let\caption@setposition\@gobble
691   \scr@caption}
692 \def\captionbelow{%
693   \caption@setposition{b}\let\caption@setposition\@gobble
694   \scr@caption}
695 }
```

### french(le) package support

```
696 \AtBeginDocument{\@ifundefined{f@ffrench}}{%
697   \caption@Info{french(le) package detected}%
```

If `\GOfrench` is defined as `\relax` all the re-definitions regarding captions have already been done, so we can do our patches immediately. Otherwise we must add our stuff to `\GOfrench`.

```
698   \@ifundefined{GOfrench}%
699     {\let\caption@tempa\@firstofone}%
700     {\def\caption@tempa{\g@addto@macro\GOfrench}}%
701   \caption@tempa{%
```

`\@cnORI` We update the definition of `\@cnORI` so it actually reflects our definition of `\caption`.

```
702   \let\@cnORI\caption
```

`\@tablescaption` The **french(le)** package sets `\caption` to `\@tablescaption` at `\begin{table}` for special treatment of footnotes. Therefore we have to patch `\@tablescaption` so `\caption*` will work inside the table environment.

```
703   \let\caption@tablescaption\@tablescaption
704   \def\@tablescaption{\caption@caption\caption@tablescaption}%
```

`\f@ffrench` `\f@tfrench` `\f@ffrench` and `\f@tfrench` reflect `\fnum@figure` and `\fnum@table` when used in french mode. These contain additional code which typesets the caption separator `\captionseparator` instead of the usual colon. Because this breaks with our `\@makecaption` code we have to remove this additional code here.

```
705   \let\@eatDP\@undefined
706   \let\caption@tempa\@empty
707   \ifx\f@ffrench\fnum@figure
708     \l@addto@macro\caption@tempa{\let\fnum@figure\f@ffrench}%
709   \fi
710   \ifx\f@tfrench\fnum@table
711     \l@addto@macro\caption@tempa{\let\fnum@table\f@tfrench}%
712   \fi
713   \def\f@ffrench{\ifx\listoffigures\relax\else\figurename~\thefigure\fi}%
714   \def\f@tfrench{\ifx\listoftables\relax\else\tablename~\thetable\fi}%
715   \caption@tempa

716 }}}}
```

### float package support

The float package usually do not use the L<sup>A</sup>T<sub>E</sub>X 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 select 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 with skips.

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

719 \caption@ifpackage{float}{\float@setevery}{%
720   \caption@Info{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.

```
721 \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.

```
722 \renewcommand*\caption@of[2]{%
723   \@ifundefined{fst@#2}{}{%
724     \let\caption@of@float\@firstofone
725     \@nameuse{fst@#2}\@float@setevery{#2}}%
726   \def\@capttype{#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 requested by `\captionof`. (see above)

```

727 \let\caption@@float\float@caption
728 \long\def\float@caption#1[#2]#3{%
729   \caption@beginex{#1}{#2}{#3}%
730   \let\@fs@capt\caption@@make
731   \caption@@float{#1}[{#2}]{#3}%
732   \caption@of@float{%

```

If the `hyperref` package is loaded, we need to set the appropriate anchor for ourself. To do so without adding extra vertical space we need to save (and restore) `\prevdepth` and switch off the interline skip.

```

733   \ifundefined{hyper@@anchor}{}{%
734     \begingroup
735     \@tempdima\prevdepth
736     \nointerlineskip
737     \let\leavevmode\relax
738     \hyper@@anchor\@currentHref\relax
739     \prevdepth\@tempdima
740     \endgroup}%
741   \def\caption@@make##1##2{\unvbox\@floatcapt}%
742   \makecaption{}{}%
743   \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 adaptations and to setup the proper caption style (if defined) and additional settings declared with `\captionsetup[<float style>]`.

```

744 \let\caption@float@setevery\@float@setevery
745 \def\@float@setevery#1{%

```

$\text{\LaTeX}$  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.

```

746 \def\caption@floatname##1{\@nameuse{fname@#1}}%
747 \def\caption@setfloatname##1{\@namedef{fname@#1}}%

```

Both `\newfloat` and `\restylefloat` save 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.)

```

748 \expandafter\let\expandafter\caption@tempa\csname @float@c@#1\endcsname
749 \ifx\caption@tempa\float@caption
750 \else\ifx\caption@tempa\caption
751 \else\ifx\caption@tempa\caption@@float
752 \caption@Info{\protect\@float@c@#1\space := \protect\float@caption}%
753 \expandafter\let\csname @float@c@#1\endcsname\float@caption
754 \else
755 \caption@Info{\protect\@float@c@#1\space := \protect\caption}%
756 \expandafter\let\csname @float@c@#1\endcsname\caption
757 \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`.

```

758 \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`)

```

759 \caption@setfloatposition

```

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.

```

760 \expandafter\let\expandafter\caption@fst\csname fst@#1\endcsname
761 \edef\caption@fst{\noexpand\string\expandafter\noexpand\caption@fst}%
762 \edef\caption@fst{\noexpand\@gobblefour\caption@fst}%
763 % \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.

```

764 \caption@setstyle*\caption@fst
765 \caption@settype\caption@fst

```

```

766     \fi
767     \caption@float@setevery{#1}}%

\fs@plaintop  The float styles plaintop and boxed don't use our skip which can be set with skip=
\fs@boxed    : plaintop uses \belowcaptionskip instead of \abovecaptionskip, and
               boxed uses a fixed space of 2pt. So we patch the according float style macros here to
               change this.
768     \g@addto@macro\fs@plaintop{\def\@fs@mid{\vspace\abovecaptionskip\relax}}%
769     \g@addto@macro\fs@boxed{\def\@fs@mid{\kern\abovecaptionskip\relax}}%

770 }

The skip between 'boxed' floats and their caption defaults to 2pt.
771 \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 some
caption parameters 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'.
772 \caption@ifbool{ruled}{%
773   \captionsetup[ruled]{margin=0pt,singlelinecheck=0}%
774 }{%
775   \DeclareCaptionStyle{ruled}{labelfont=bf,labelsep=space,strut=0}%
776 }
777 \caption@undefbool{ruled}

floatrow package support

The floatrow package is already adapted for usage with the caption package. So the main
work has already been done by Mrs.Lapko, there are only two little things we have to take
care about:
778 \caption@ifpackage{floatrow}{\flrow@setlist}{%
779   \caption@Info{floatrow package v0.1f (or newer) detected}%

\caption@of  Captions typeset with \captionof should have the correct layout, so we have to 'acti-
               vate' 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.)
780   \renewcommand*\caption@of[2]{%
781     \def\@captype{#2}\flrow@setlist{{#2}}#1}%

\caption@floatname  The floatrow package follows the same naming convention as the float package; so we
                    have to adapt \caption@floatname here, too.
782   \renewcommand*\caption@floatname[1]{%
783     \@nameuse{\@ifundefined{fname@#1}{#1name}{fname@#1}}}%
784   \renewcommand*\caption@setfloatname[1]{%
785     \@namedef{\@ifundefined{fname@#1}{#1name}{fname@#1}}}%

786 }

```



## 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

$$\langle type \rangle . \langle type \# \rangle . \langle continue \# \rangle$$

will be used instead of

$$\langle type \rangle . \langle type \# \rangle .$$

(This implementation follows an idea from Steven Douglas Cochran.)

Note: This does not help if `\Hy@naturalnamestrue` is set.

```
787 \caption@ifpackage{hyperref}{\theHfigure}{%
788   \caption@Info{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`.

```
789   \def\caption@ContinuedFloat#1{%
790     \@ifundefined{theH#1}{}{}%
791     \@ifundefined{CF@#1}{}%
792     \expandafter\newcount\csname CF@#1\endcsname
793     \caption@resetContinuedFloat{#1}{}%
794     \global\advance\csname CF@#1\endcsname\@ne\relax
795     \expandafter\l@addto@macro\csname theH#1\endcsname{%
796       .\expandafter\@arabic\csname CF@#1\endcsname}%
797     \let\caption@resetContinuedFloat\@gobble
798   }%

```

```
\caption@resetContinuedFloat
```

If a continuation counter is defined, we reset it.

```
799   \def\caption@resetContinuedFloat#1{%
800     \@ifundefined{CF@#1}{}{\global\csname CF@#1\endcsname\z@\relax}}%
801 }
```

## 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.<sup>4</sup> 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.)

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

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

```
804 \let\caption@capstart\capstart
805 \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`.

```
806   \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`.

```
807   \begingroup
808     \let\H@refstepcounter\@gobble
809     \Hy@hypertexnamesfalse
810     \caption@capstart
811     \global\let\hc@currentHref\@currentHref
812   \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.

```
813   \def\caption{%
814     \let\caption\hc@org@caption
815     \H@refstepcounter\@captype
816     \caption@caption\hc@caption}}%
```

---

<sup>4</sup>This issue was fixed in `hypcap` v1.6

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

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

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

```
820 \let\caption@hyper@makecurrent\hyper@makecurrent
821 \def\hyper@makecurrent\@capttype{%
822 \let\hyper@makecurrent\caption@hyper@makecurrent
823 \global\let\@currentHref\hc@currentHref}%
824 \caption@hc@@caption{#1} [{#2}] {#3}%
825 \caption@end}%
826 }
```

### listings package support

```
827 \caption@ifpackage{listings}{\lst@MakeCaption}{%
828 \caption@Info{listings package v1.2 (or newer) detected}%
```

`\lst@MakeCaption` To support the `listings` package we need to redefine `\lst@MakeCaption` so the original stuff is nested with `\caption@begin` and `\caption@end`.

```
829 \let\caption@lst@MakeCaption\lst@MakeCaption
830 \def\lst@MakeCaption#1{%
```

If the `position=` is set to `auto`, we take over the `captionpos=` setting from the `listings` package. Note that we won't do this otherwise, so `listings` settings like `abovecaptionskip=0pt`, `belowcaptionskip=10pt`, `captionpos=t` will *not* cause different outputs with or without the `caption` package loaded.

```
831 \def\caption@autoposition{\caption@setposition{#1}}%
832 \caption@begin{lstlisting}%
833 \caption@lst@MakeCaption{#1}%
834 \caption@end}%
835 }
```

### longtable package support

```
836 \caption@ifpackage{longtable}{\LT@makecaption}{%
837 \caption@Info{longtable package v3.15 (or newer) detected}%
```

`\LT@makecaption` `\LT@makecaption{<cmd>}{<label>}{<text>}`

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}}}

```

```

838 \def\LT@makecaption#1#2#3{%
839   \caption@LT@make{%

```

We set `\ifcaption@star` according to the 1st argument.

```

840   \caption@startrue#1\caption@starfalse

```

If `\LTcapwidth` is not set to its default value 4in we assume that it shall overwrite our own setting. (But `\captionsetup[longtable]{width=...}` will overwrite `\LTcapwidth`.)

```

841   \caption@letfloattype{longtable}{%
842     \ifdim\LTcapwidth=4in \else
843       \setcaptionwidth\LTcapwidth
844     \fi}%

```

The default `position=` setting for longtables is `top`. (This emulates the standard behaviour of the `longtable` package which has no skip above the caption but a skip below it.)

```

845 %   \caption@setdefaultpos{t}%
846   \let\caption@defaultpos\@firstoftwo

```

`position=auto` is a bad idea for longtables, but we do our very best. This works quite well for captions inside the longtable contents, but not for captions inside the longtable (end)foot.

```

847   \def\caption@autoposition{%
848     \caption@setposition{\ifcase\LT@rows t\else b\fi}}%
849   \caption@begin{table}%

```

The following skip has the purpose to correct the height of the `\parbox[t]`. Usually it's the height of the very first line, but because of our extra skips (`\abovecaptionskip` and `\belowcaptionskip`) it's always 0pt. (A different idea would be typesetting the first skip outside the longtable column with `\noalign{\vskip...}`, but this means we have to move `\caption@begin` to some other place because it does not work in tabular mode...)

```

850   \vskip-\ht\strutbox

```

This should look familiar. We do our skips and use `\caption@@make` to typeset the caption itself.

```

851   \caption@iftop{\vskip\belowcaptionskip}{\vskip\abovecaptionskip}%
852   \caption@@make{#2}{#3}\endgraf
853   \caption@iftop{\vskip\abovecaptionskip}{\vskip\belowcaptionskip}%
854   \caption@end}}%

```

```

\caption@LT@make Typesets the caption as centered \multicolumn...
855 \newcommand\caption@LT@make[1]{%
856 \LT@mcol\LT@cols c{\hbox to\z@{\hss\parbox[t]{\hsize{#1}\hss}}}%
857 }

picins package support

858 \caption@ifpackage{picins}{piccaption}{%
859 \caption@Info{picins package v3.0 (or newer) detected}%

\piccaption Original code:

\def\piccaption{\@ifnextchar [{\@piccaption}{\@piccaption[]}}

860 \def\piccaption{\@dblarg\@piccaption}
861 % \def\piccaption{\caption@caption{\@dblarg\@piccaption}}

TODO: Make \piccaption[] {...} and \piccaption{} work
862 }

rotating package support

863 \caption@ifpackage{rotating}{@rotcaption}{%
864 \caption@Info{rotating package v2.0 (or newer) detected}%

\rotcaption Make \rotcaption* work.

865 \def\rotcaption{\let\@makecaption\@makerotcaption\caption}%
866 % \let\@rotcaption\@undefined

\rotcaptionof Make \rotcaptionof(*) work.

867 \def\rotcaptionof{%
868 \@ifstar{\caption@of{\rotcaption*}}{\caption@of\rotcaption}}%

\@makerotcaption Original (bugfixed) code:

\long\def\@makerotcaption#1#2{%
\setbox\@tempboxa\hbox{#1: #2}%
\ifdim \wd\@tempboxa > .8\vsiz
\rotatebox{90}{%
\begin{minipage}{.8\textheight}#1: #2\end{minipage}%
}%\par % <== \par removed (AR)
\else%
\rotatebox{90}{\box\@tempboxa}%
\fi
\nobreak\hspace{12pt}% <== \nobreak added (AR)
}

```

Our version emulates this behaviour, but if `width=` is set, the rotated caption is always typeset as `minipage`. (Note that `margin=` is not supported here.)

```

869 \long\def\@makerotcaption#1#2{%
870     \ifdim\captionwidth=\z@
871         \setcaptionwidth{.8\textheight}%
872         \caption@slc{#1}{#2}{.8\vsizerelax}%
873         \let\caption@makerot\caption@@make
874         \setcaptionmargin\z@
875         \setlength\captionindent\z@
876     \else
877         \long\def\captionbox##1##2{\hbox{\hsize=.8\textheight\relax##2}}%
878         (not needed because \rotatebox uses an \hbox anyway)
879         \let\captionbox\@secondoftwo%
880         \caption@setbool{slc}{0}% been there, done that
881     \fi
882     \rotatebox{90}{\caption@makerot{#1}{#2}}%
883     \nobreak\hspace{12pt}}%
884 \newcommand\caption@makerot[2]{%
885     \begin{minipage}\captionwidth\caption@@make{#1}{#2}\end{minipage}}%

```

### sidecap package support

```

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

```

`\SC@caption` First of all, we let `sidecap` use an actual definition of `\caption`.  
(This is only required for version 1.5d of the `sidecap` package.)

```

888 \@ifundefined{caption@caption}%
889     {\let\caption@tempa\AtBeginDocument}%
890     {\let\caption@tempa\@firstofone}%
891 \caption@tempa{\let\SC@caption=\caption}%

```

`\SC@zfloat` This macro will be called at the start of the environment, here is a good opportunity to do some adaptations to `\caption` and `\captionsetup`.

```

892 \let\caption@SC@zfloat\SC@zfloat
893 \def\SC@zfloat#1#2#3[#4]{%

```

Note: #2 is either figure or table and will be stored to `\SC@captype` by the original version of `\SC@zfloat`.

```

894     \caption@SC@zfloat{#1}{#2}{#3}[#4]%

```

Since the `sidecap` package uses our `\caption` code outside the floating environment the regular `\captionsetup` will not work. So we need a special version here which saves the given argument list which will be executed later on.

```

895 \global\let\SC@CAPsetup\empty
896 \def\captionsetup##1{\g@addto@macro\SC@CAPsetup{,##1}}%

```

Make `\caption*` work.

```
897 \let\caption@SC\caption
898 % \def\caption{\renewcommand\captionsetup[1]{} \caption@caption\caption@SC}%
899 \def\caption{\caption@caption\caption@SC}%
900 }%
```

`\endSC@FLOAT` This macro will be called at the end of the environment, here we need to setup our stuff before the `sidecap` package actually typesets its caption.

```
901 \let\caption@endSC@FLOAT\endSC@FLOAT
902 \def\endSC@FLOAT{%
```

Note that `\@capttype` isn't defined so far, this will be done inside the original definition of `\endSC@FLOAT`. But we define `\@capttype` already here to make `\captionsetup` work with `\@capttype`-based options (like `type=`).

```
903 \let\@capttype\SC@capttype
```

Here we execute the options set with `\captionsetup` inside this environment.

```
904 \caption@esetup\SC@CAPsetup
```

Before we can typeset the caption we need to set the margin to zero because any extra margin would only be 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. The usage of `\SC@justify` differs from version to version of the `sidecap` package:

Version 1.4: `\SC@justify` is not defined

Version 1.5: `\SC@justify` is `\relax` when not set

Version 1.6: `\SC@justify` is `\@empty` when not set

```
905 \caption@letfloattype{SC\@capttype}{%
906 \listdepth\z@
907 \setcaptionmargin\z@
908 \@ifundefined{SC@justify}{}{%
909 \ifx\SC@justify\@empty\else
910 \let\caption@hj\SC@justify
911 \let\SC@justify\@empty
912 \fi}}%
```

We adapt `\caption@ifempty` so `\caption{}` will work within these environments, too.

```
913 \long\def\caption@ifempty##1{%
914 \ifx\SC@CAPtext\@empty
915 \expandafter\@firstofone
916 \else
917 \expandafter\@gobble
918 \fi}%
```

Finally we call the original definition of `\endSC@FLOAT` which will call our version of `\caption` to typeset the caption.

```
919 \caption@endSC@FLOAT}%
```

```
920 }
```

### supertabular package support

```

\caption@setSTposition  The position= setting will be overwritten by the supertabular package: If \topcaption
                        is used, the position will be top automatically, bottom otherwise.
921 \def\caption@setSTposition{%
922   \caption@setposition{\if@topcaption t\else b\fi}}

923 \caption@ifpackage{supertabular}{ST@caption}{%
924   \caption@Info{supertabular package detected}}%

\tablecaption  Make \topcaption* and \bottomcaption* work.
925   \let\caption@tablecaption\tablecaption
926   \def\tablecaption{\caption@caption\caption@tablecaption}%

\ST@caption  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}

927 \long\def\ST@caption#1[#2]#3{\par%
928   \caption@letfloattype{supertabular}{}%
929   \let\caption@fixposition\caption@setSTposition
930   \caption@beginex{#1}{#2}{#3}%
931   \addcontentsline{\csname ext@#1\endcsname}{#1}%
932     {\protect\numberline{%
933       \csname the#1\endcsname}{\ignorespaces #2}}}%
934   \@parboxrestore
935   \normalsize
936   \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
937   \caption@end}%
938 }

```



## References

- [1] Frank Mittelbach and Michel Goossens: *The L<sup>A</sup>T<sub>E</sub>X 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 & Heiko Oberdiek: *Hypertext marks in L<sup>A</sup>T<sub>E</sub>X*, 2007/01/25
- [5] Heiko Oberdiek: *The hypcap package – Adjusting anchors of captions* 2007/02/19
- [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 L<sup>A</sup>T<sub>E</sub>X*, 1997/09/26
- [9] Rolf Niepraschk und Hubert Gäßlein: *The sidecap package*, 2003/06/06
- [10] Steven D. Cochran: *The subfig package*, 2005/07/05
- [11] Johannes Braams und Theo Jurriens: *The supertabular environment*, 2002/07/19
- [12] Anne Brüggemann-Klein: *Einführung in die Dokumentverarbeitung*, B.G. Teubner, Stuttgart, 1989