

Typesetting captions with the caption package*

Axel Sommerfeldt
caption@sommerfee.de

2006/01/12

Abstract

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

Contents

1	Introduction	2
2	Using the package	3
3	Options	3
3.1	Formatting	3
3.2	Justification	6
3.3	Fonts	7
3.4	Margins and further paragraph options	8
3.5	Styles	10
3.6	Skips	10
4	Useful stuff	11
5	Do it yourself!	14
5.1	Examples	16
6	Using non-standard document classes	17
7	Using other packages	18
7.1	The float package	18
7.2	The listings package	19
7.3	The longtable package	19

*This package has version number v3.0j, last revised 2006/03/21.

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

7.4	The rotating package	19
7.5	The sidecap package	20
7.6	The supertabular package	20
7.7	Known incompatibilities	20
8	Compatibility to older versions	20
8.1	The caption package version 1.x	20
8.2	The caption2 package version 2.x	22
9	Further reading	23
10	Thanks	23
11	The Implementation	24
11.1	Kernel	24
11.2	Main package	38

1 Introduction

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

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

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

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

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

2 Using the package

`\usepackage` Insert

```
\usepackage[<options>]{caption}[2006/03/21]
```

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 L ^A T _E X document classes.)
	<code>hang</code>	Indents the caption text, so it will ‘hang’ under the first line of the text.
	<code>...</code>	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>`.

Three 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 three standard caption label formats:

<code>default</code>	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.)
----------------------	--

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

yields captions like this one:

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

`labelsep=` With the options

```
labelsep=<label separator name>
```

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

<code>none</code>	There is no caption separator. This option only makes sense when used together with other options like <code>labelformat=empty</code> .
<code>colon</code>	The caption label and text will be separated by a colon and a space. (This is the default one.)
<code>period</code>	The caption label and text will be separated by a period and a space.
<code>space</code>	The caption label and text will be separated by a single space.
<code>quad</code>	The caption label and text will be separated by a <code>\quad</code> .
<code>newline</code>	The caption label and text will be separated by a line break (<code>\\</code>).
New feature v3.0h <code>endash</code>	The caption label and text will be separated by an en-dash, surrounded by spaces (<code>--</code>).
<code>...</code>	Own separators can be defined using <code>\DeclareCaptionLabelSeparator</code> . (See section 5: “ <i>Do it yourself</i> ”)

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.
<code>RaggedRight</code>	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. This difference is that this time the word breaking algorithm of \TeX will work inside the caption.
<code>raggedleft</code>	Each line of the caption will be moved to the right margin.
<code>...</code>	Own justifications can be defined using <code>\DeclareCaptionJustification</code> . (See section 5: “ <i>Do it yourself</i> ”)

Two 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 L^AT_EX document classes (`article`, `report`, and `book`) 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>* you switch off the extra centering:

```
singlelinecheck=false
```

Doing so the above short caption would look like

Figure 13: A short caption.

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

3.3 Fonts

`font=` There are three font options which affects different parts of the caption: One affecting the whole caption (`font`), one which only affects the caption label and separator (`labelfont`) and at last one which only affects the caption text (`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:

```
scriptsize    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>	Bold series
<code>rm</code>	Roman family
<code>sf</code>	Sans Serif family
<code>tt</code>	Typewriter family
<code>...</code>	Own font options can be defined using <code>\DeclareCaptionFont</code> . (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` yield 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 using
`width=` the options

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


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

Two examples illustrating this:

```
margin=10pt
```

Figure 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 `\abovecaptionskip`
`belowskip=` and `\belowcaptionskip`. The standard L^AT_EX document classes `article`, `report` and `book` set `\abovecaptionskip` to 10pt and `\belowcaptionskip` to 0pt.

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

```
aboveskip=<amount> and
belowskip=<amount> .
```

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

Please compare the spacing in these small tables:

Table 1: A table		A B	
		C D	
		A B	
		C D	

Table 2: A table

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

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

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

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

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

tableposition=

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

E.g.,

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

New feature
v3.0a

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

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

is equivalent to

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

4 Useful stuff

`\caption` The command

```
\caption[⟨lst.entry⟩]{⟨heading⟩}
```

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

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

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

```

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

```

looks like

A table

A	B
C	D

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

```

\begin{table}
\caption*{A table}
\begin{tabular}{cc}
A & B \\
C & D \\
\end{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{<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 so a figure or table with a `\ContinuedFloat` in it gets the same reference number as the figure or table before.

An example:

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

gives the following result:

Table 7: A table
 ...
 Table 7: A table (cont.)

`\captionsetup` We already know the `\captionsetup` command (see section 2: “Using the package”), but this time we get enlighten about the optional argument *float type*. Remember, the syntax of this command is

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

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

```
\captionsetup[figure]{options}
```

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

Here comes an example to illustrate this:

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

gives captions like this:

Figure 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{<Typ>} .`

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 only two: `figure` and `table`. But as we will see later that some \LaTeX packages exist (like the `float`, `longtable`, and `sidecap` package for example) who can define 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 defined inside `caption.sty` 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`
`\bothIfSecond`

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

`\bothIfFirst{<first arg>}{<second arg>}` and
`\bothIfSecond{<first arg>}{<second arg>}`

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

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

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

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

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

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

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

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

Again an easy example taken from `caption.sty` 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}
```

yields captions with all lines moved to the left margin.

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

```
\DeclareCaptionFont{<name>}{<code>}
```

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

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

New description
v3.0h

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

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

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

```
\DeclareCaptionStyle{<name>}[<additional options>]{<options>}
```

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

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

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

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

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

5.1 Examples

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

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

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

Figure 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 an appropriate definition is

```
\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 options like `tablecaptionabove` and commands like `\captionabove`, `\captionbelow`, `\captionformat`, `\figureformat`, `\tableformat`, `\setcapindent`, `\setcaphanging`, `\captionstyle` etc. will not work anymore. So make a wise decision!

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

7 Using other packages

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

These are the packages the caption package is adapted to:

<code>float</code>	Gives you the possibility to define new floating environments
<code>hypcap</code>	Adjusting hyperref anchors of captions
<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} .
```

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

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

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

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

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

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

For further information about the supported packages please take a look at the documentation belonging to it or buy yourself The L^AT_EX 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 L^AT_EX. While the latter one is only a recommendation to L^AT_EX to set the float “here”, the `H` forces the float to appear exactly at the spot where it occurs in your input file and nowhere else.

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

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

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

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

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

```
#include <stdio.h>

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

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

```
\DeclareCaptionStyle{ruled}{{options}}
```

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

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

7.2 The listings package

New description
v3.0b

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

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

7.3 The longtable package

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

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

7.4 The rotating package

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

7.5 The sidecap package

New description
v3.0b

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

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

listof= Using the sidecap package you will probably notice that suppressing the entry in the list of figures or tables with `\caption[]{\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.

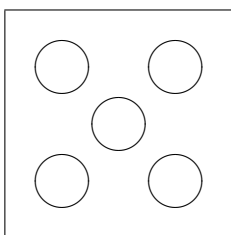


Figure 31: A small example with the caption beside the figure.

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 \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`, `hvfloating`, 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 yield to 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
normal	format=plain
hang	format=hang
isu	format=hang
center	justification=centering
centerlast	justification=centerlast
nooneline	singlelinecheck=off
scriptsize	font=scriptsize
footnotesize	font=footnotesize
small	font=small
normalsize	font=normalsize
large	font=large
Large	font=Large
up	labelfont=up
it	labelfont=it
sl	labelfont=sl
sc	labelfont=sc
md	labelfont=md
bf	labelfont=bf
rm	labelfont=rm
sf	labelfont=sf
tt	labelfont=tt

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}
```

should now be written 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` who 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 help you removing these errors. If you have problems migrating from `caption2` to `caption` please don’t hesitate to send me an e-mail.

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>online</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*”)

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

`\captionstyle`, `\captionlabeldelim`, `\captionlabelsep`,
`\captionindent`, `\captionlabelfalse`, `\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 yield to 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 could 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 T_EX FAQ - Frequently asked questions about T_EX and L^AT_EX:

<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 L^AT_EX 2_ε 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 Hinz, 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, and Uwe Siart 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 \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 \LaTeX document – it just provides functionality which can be used by \LaTeX 2_ϵ packages which typesets captions, like the caption package or the subfig package.

The caption package itself redefines the \LaTeX commands `\caption`, `\@caption`, and `\@makecaption` and maps the latter one to `\caption@@make`, giving the user the possibility to control the captions of the floating environments `figure` and `table`. Furthermore it does similar to the caption stuff coming from other packages (like the `longtable` 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 \LaTeX itself or a \LaTeX 2_ϵ package and the caption stuff offered by the caption kernel.

11.1 Kernel

Identification

```
1 \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2 \ProvidesPackage{caption3}[2006/03/16 v3.0j caption3 kernel (AR)]
3 {+debug}\PackageWarning{caption3}{DEBUG VERSION}
```

Generic helpers

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

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

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

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

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

```
11 \def\bothIfFirst#1#2{%
12   \protected@edef\caption@tempa{#1}%
```



```

13 \ifx\caption@tempa\@empty\else
14   #1#2%
15 \fi}

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

```

`\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.

```

21 \def\caption@ifinlist#1#2{%
22   \let\next\@secondoftwo
23   \edef\caption@tempa{#1}%
24   \@for\caption@tempb:=#2\do{%
25     \ifx\caption@tempa\caption@tempb
26       \let\next\@firstoftwo
27     \fi}%
28   \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>}

```

```

29 \def\caption@setbool#1#2{%
30   \caption@ifinlist{#2}{1,true,yes,on}{%
31     \expandafter\let\csname caption@if#1\endcsname\@firstoftwo
32   }{\caption@ifinlist{#2}{0,false,no,off}{%
33     \expandafter\let\csname caption@if#1\endcsname\@secondoftwo
34   }}%
35   \PackageError{caption}{Undefined boolean value `#2'}{\caption@eh}%
36   }}

37 \def\caption@ifbool#1{\@nameuse{caption@if#1}}
38 \def\caption@undefbool#1{\@nameundef{caption@if#1}}

```

Using the keyval package

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

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

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

```

40 \providecommand*\undefine@key[2]{%
41   \@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 L^AT_EX 2_ε), 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.

```

42 \newcommand\DeclareCaptionOption{%
43   \@ifstar{\caption@declareoption\AtEndOfPackage}%
44   {\caption@declareoption\@gobble}}
45 \newcommand*\caption@declareoption[2]{%
46   #1{\undefine@key{caption}{#2}}\define@key{caption}{#2}}
47 \@onlypreamble\DeclareCaptionOption
48 \@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`.

```

49 \newcommand\captionsetup{\@ifnextchar[\caption@setuptype\caption@setup}
50 \newcommand\caption@typ@{\caption@typ@} % This saves 74 words of TeX memory
51 \def\caption@setuptype[#1]#2{%
52   \@ifundefined{\caption@typ@#1}%
53   {\@namedef{\caption@typ@#1}{#2}}%
54   {\expandafter\l@addto@macro\csname\caption@typ@#1\endcsname{, #2}}}
55 \newcommand\caption@setup{\caption@setkeys{caption}}

```

`\caption@setkeys` This one simply calls `\setkeys` but lets error messages refer to the caption package instead of the keyval package.

```

56 \newcommand*\caption@setkeys[2]{%
57   \let\caption@KV@errx\KV@errx
58   \let\caption@KV@err\KV@err
59   \def\KV@errx#1{\PackageError\caption@package{##1}\@ehc}%
60   \let\KV@err\KV@errx
61   \setkeys{#1}{#2}%
62   \let\KV@errx\caption@KV@errx
63   \let\KV@err\caption@KV@err}
64 \newcommand\caption@package{caption}

```

`\caption@settype` `\caption@settype[<package>]{<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.)

```

65 \newcommand\caption@settype{%
66   \@ifnextchar[\caption@@settype{\caption@@settype{caption}}]
67 \def\caption@@settype[#1]#2{%
68   \@ifundefined{\caption@typ@#2}{%}{%
69     \def\caption@package{#1}%
70     \caption@esetup{\csname\caption@typ@#2\endcsname}%
71     \def\caption@package{caption}}}%

```

`\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.

```

72 \newcommand*\caption@esetup[1]{%

```

```

73 \edef\caption@tempa{\noexpand\caption@setup{#1}}%
74 \caption@tempa}

\clearcaptionsetup \clearcaptionsetup{<type>}
This removes the saved option list associated with <type>.
75 \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>.
76 \newcommand*\showcaptionsetup[2][\@firstofone]{%
77 \GenericWarning{}}{%
78 #1 Caption Info: KV list on `#2'\MessageBreak
79 #1 Caption Data: (%
80 \@ifundefined{\caption@typ@#2}{%
81 % empty -- print nothing
82 }{%
83 \@nameuse{\caption@typ@#2}%
84 }%
85 )}}

```

Errors

```

\caption@eh At the moment we only offer this simple error message as generic helper for the user.
86 \newcommand\caption@eh{%
87 If you do not understand this error, please take a closer look\MessageBreak
88 at the documentation of the 'caption' package.\MessageBreak
89 \@ehc}

```

Margin resp. width

```

\captionmargin \captionmargin and \captionwidth contain the extra margin resp. the total
\captionmarginx width used for captions. Please never set these values in a direct way, they are just acces-
\captionwidth sible 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.
90 \newdimen\captionmargin
91 \newdimen\captionmarginx
92 \newdimen\captionwidth

93 \DeclareCaptionOption{margin}{\setcaptionmargin{#1}}
94 \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.
95 \newcommand*\setcaptionmargin[1]{%
96 \captionwidth\z@
97 \caption@@setmargin#1,#1,\@nil\@@}
98 \def\caption@@setmargin#1,#2,#3\@@{%
99 \setlength\captionmargin{#1}%

```

```

100 \setlength\captionmarginx{#2}%
101 \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.

```

```

102 \newcommand\setcaptionwidth{%
103 \setlength\captionwidth}

```

Indentions

```

\captionindent These are the indentions we support.
\captionparindent 104 \newdimen\captionindent
\captionhangindent 105 \newdimen\captionparindent
106 \newdimen\captionhangindent

```

```

107 \DeclareCaptionOption{indent}[\leftmargini]{\setlength\captionindent{#1}}% obsolete
108 \DeclareCaptionOption{indention}[\leftmargini]{\setlength\captionindent{#1}}
109 \DeclareCaptionOption{hangindent}{\setlength\captionhangindent{#1}}
110 \DeclareCaptionOption{parindent}{\setlength\captionparindent{#1}}
111 \DeclareCaptionOption{parskip}{\l@addto@macro\caption@@par{\setlength\parskip{#1}}}

```

Styles

```

\DeclareCaptionStyle \DeclareCaptionStyle{<name>}[<single-line-list-of-KV>]{<list-of-KV>}
112 \newcommand*\DeclareCaptionStyle[1]{%
113 \@ifnextchar[\caption@declarestyle{#1}]{\caption@declarestyle{#1}[]}}
114 \def\caption@declarestyle#1[#2]#3{%
115 \global\@namedef{caption@sls@#1}{#2}%
116 \global\@namedef{caption@sty@#1}{#3}}
117 \@onlypreamble\DeclareCaptionStyle
118 \@onlypreamble\caption@declarestyle
119 \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.

```

120 \newcommand\caption@setstyle{%
121 \@ifstar{\caption@@setstyle\@gobble}{\caption@@setstyle\@firstofone}}
122 \newcommand*\caption@@setstyle[2]{%
123 \@ifundefined{caption@sty@#2}%
124 {#1{\PackageError{caption}{Undefined caption style `#2'}{\caption@eh}}}%
125 {\expandafter\let\expandafter\caption@sls\csname caption@sls@#2\endcsname
126 \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!

```

```

127 \newcommand\caption@setdefault{\captionsetup{%
128   format=default,labelformat=default,labelsep=default,%
129   justification=default,font=default,labelfont=default,textfont=default,%
130   margin=0pt,indent=0pt,parindent=0pt,hangindent=0pt,%
131   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 \LaTeX document classes: If the caption fits in one single line, it is typeset centered.

```

132 \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.

```

133 \newcommand\DeclareCaptionFormat{%
134   \@ifstar{\caption@declareformat\@gobble}%
135           {\caption@declareformat\@firstofone}}
136 \newcommand\caption@declareformat[3]{%
137   \global\expandafter\let\csname caption@ifh@#2\endcsname#1%
138   \global\long\@namedef{caption@fmt@#2}##1##2##3{#3}}
139 \@onlypreamble\DeclareCaptionFormat
140 \@onlypreamble\caption@declareformat

```

```

141 \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`).

```

142 \newcommand*\caption@setformat[1]{%
143   \@ifundefined{caption@fmt@#1}%
144   {\PackageError{caption}{Undefined caption format ‘#1’}{\caption@eh}}%
145   {\expandafter\let\expandafter\caption@ifh\csname caption@ifh@#1\endcsname
146    \expandafter\let\expandafter\caption@fmt\csname caption@fmt@#1\endcsname}}

```

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

```

147 \DeclareCaptionFormat{plain}{#1#2#3\par}
148 \DeclareCaptionFormat{hang}{%
149   \@hangfrom{#1#2}%
150   \advance\captionparindent\hangindent
151   \advance\captionhangindent\hangindent
152   \caption@@par
153   #3\par}

```

‘default’ usually maps to ‘plain’.

```

154 \def\caption@fmt@default{\caption@fmt@plain}
155 \def\caption@ifh@default{\caption@ifh@plain}

```

Label formats

```

DeclareCaptionLabelFormat \DeclareCaptionLabelFormat{<name>}{<code with #1 and #2>}
156 \newcommand*{\DeclareCaptionLabelFormat}[2]{%
157   \global\@namedef{caption@lfmt@#1}##1##2{#2}}
158 \@onlypreamble\DeclareCaptionLabelFormat

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

```

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

Selecting a caption label format simply means saving the code (in `\caption@lfmt`).

```

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

```

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

```

164 \DeclareCaptionLabelFormat{empty}{}
165 \DeclareCaptionLabelFormat{simple}{\bothIfFirst{#1}{\nobreakspace}{#2}}
166 \DeclareCaptionLabelFormat{parens}{\bothIfFirst{#1}{\nobreakspace}{#2}}

```

‘default’ usually maps to ‘simple’.

```
167 \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.

```

```

168 \newcommand\DeclareCaptionLabelSeparator{%
169   \@ifstar{\caption@declarelabelseparator\@gobble}%
170           {\caption@declarelabelseparator\@firstofone}}
171 \newcommand\caption@declarelabelseparator[3]{%
172   \global\expandafter\let\csname caption@iflfl@#2\endcsname#1%
173   \global\long\@namedef{caption@lsep@#2}{#3}}
174 \@onlypreamble\DeclareCaptionLabelSeparator
175 \@onlypreamble\caption@declarelabelseparator

176 \DeclareCaptionOption{labelsep}{\caption@setlabelseparator{#1}}
177 \DeclareCaptionOption{labelseparator}{\caption@setlabelseparator{#1}}

```

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

Selecting a caption label separator simply means saving the code (in `\caption@lsep`).

```

178 \newcommand*{\caption@setlabelseparator}[1]{%
179   \@ifundefined{caption@lsep@#1}%
180     {\PackageError{caption}{Undefined caption label separator `#1'}{\caption@eh}}%
181     {\expandafter\let\expandafter\caption@iflfl\csname caption@iflfl@#1\endcsname
182     \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’.

```

183 \DeclareCaptionLabelSeparator{none}{}
184 \DeclareCaptionLabelSeparator{colon}{: }

```

```

185 \DeclareCaptionLabelSeparator{period}{. }
186 \DeclareCaptionLabelSeparator{space}{ }
187 \DeclareCaptionLabelSeparator*{quad}{\quad}
188 \DeclareCaptionLabelSeparator*{newline}{\\}
189 \DeclareCaptionLabelSeparator*{endash}{\space\textendash\space}

‘default’ usually maps to ‘colon’.

190 \def\caption@lsep@default{\caption@lsep@colon}
191 \def\caption@iflf@default{\caption@iflf@colon}

```

Justifications

```

\DeclareCaptionJustification \DeclareCaptionJustification{<name>}{<code>}
192 \newcommand*\DeclareCaptionJustification[2]{%
193   \global\@namedef{caption@hj@#1}{#2}}
194 %\newcommand\DeclareCaptionJustification{\DeclareCaptionFont}
195 \@onlypreamble\DeclareCaptionJustification

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

```

```

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

```

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

```

197 \newcommand*\caption@setjustification[1]{%
198   \@ifundefined{caption@hj@#1}%
199     {\PackageError{caption}{Undefined caption justification `#1'}{\caption@eh}}%
200     {\expandafter\let\expandafter\caption@hj\csname caption@hj@#1\endcsname}}
201 %\newcommand\caption@setjustification{\caption@setfont{#1}}

```

These are the pre-defined justification code snippets.

```

202 \DeclareCaptionJustification{justified}{}
203 \DeclareCaptionJustification{centering}{\centering}
204 \DeclareCaptionJustification{centerfirst}{\caption@centerfirst}
205 \DeclareCaptionJustification{centerlast}{\caption@centerlast}
206 \DeclareCaptionJustification{raggedleft}{\raggedleft}
207 \DeclareCaptionJustification{raggedright}{\raggedright}

```

‘default’ usually maps to ‘justified’.

```

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

```

\caption@centerfirst Please blame Frank Mittelbach for \caption@centerfirst and Anne Brüggemann-Klein[12] for \caption@centerlast :-)

```

209 \newcommand\caption@centerfirst{%
210   \edef\caption@normaladjust{%
211     \leftskip\the\leftskip
212     \rightskip\the\rightskip
213     \parfillskip\the\parfillskip\relax}%
214   \leftskip\z@\@plus -1fil%
215   \rightskip\z@\@plus 1fil%
216   \parfillskip\z@skip
217   \noindent\hskip\z@\@plus 2fil%
218   \@setpar{\@@par\@restorepar\caption@normaladjust}}
219 \newcommand\caption@centerlast{%
220   \leftskip\z@\@plus 1fil%
221   \rightskip\z@\@plus -1fil%
222   \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.

```
223 \DeclareCaptionJustification{Centering}{%
224   \caption@ragged\Centering\centering}
225 \DeclareCaptionJustification{RaggedLeft}{%
226   \caption@ragged\RaggedLeft\raggedleft}
227 \DeclareCaptionJustification{RaggedRight}{%
228   \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.)

```
229 \newcommand*\caption@ragged[2]{%
230   \@ifundefined{caption\string#1}{%
231     \PackageWarning{caption}{%
232       Cannot locate the 'ragged2e' package, therefore\MessageBreak
233       substituting \string#2 for \string#1\MessageBreak}%
234     \global\@namedef{caption\string#1}{}}%
235   #2}

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

Fonts

```
\DeclareCaptionFont \DeclareCaptionFont{<name>}{<code>}

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

241 \DeclareCaptionOption{font}{\caption@setfont{font}{#1}}
242 \DeclareCaptionOption{labelfont}{\caption@setfont{labelfont}{#1}}
243 \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.

```
244 \newcommand*\caption@setfont[2]{%
245   \let\caption@tempa\@empty
246   \begingroup
247 %     \define@key{caption@fnt}{default}[]{%
248 %       \global\expandafter\let\expandafter\caption@tempa
249 %       \csname caption#1@default\endcsname}%
250     \caption@setkeys{caption@fnt}{#2}%
251   \endgroup
252   \expandafter\let\csname caption#1\endcsname\caption@tempa}

253 \DeclareCaptionFont{default}{}
```


These are the pre-defined font code snippets.

```

254 \DeclareCaptionFont{scriptsize}{\scriptsize}
255 \DeclareCaptionFont{footnotesize}{\footnotesize}
256 \DeclareCaptionFont{small}{\small}
257 \DeclareCaptionFont{normalsize}{\normalsize}
258 \DeclareCaptionFont{large}{\large}
259 \DeclareCaptionFont{Large}{\Large}

260 \DeclareCaptionFont{up}{\upshape}
261 \DeclareCaptionFont{it}{\itshape}
262 \DeclareCaptionFont{sl}{\slshape}
263 \DeclareCaptionFont{sc}{\scshape}
264 \DeclareCaptionFont{md}{\mdseries}
265 \DeclareCaptionFont{bf}{\bfseries}
266 \DeclareCaptionFont{rm}{\rmfamily}
267 \DeclareCaptionFont{sf}{\sffamily}
268 \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.

```

269 \providecommand\captionsize{}

270 \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.

```

271 \@ifundefined{abovecaptionskip}{%
272   \newlength\abovecaptionskip\setlength\abovecaptionskip{10\p@}}{}
273 \@ifundefined{belowcaptionskip}{%
274   \newlength\belowcaptionskip\setlength\belowcaptionskip{0\p@}}{}

275 \DeclareCaptionOption{aboveskip}{\setlength\abovecaptionskip{#1}}
276 \DeclareCaptionOption{belowskip}{\setlength\belowcaptionskip{#1}}
277 \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.)

```

278 \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.*

```

279 \newcommand*\caption@setposition[1]{%
280   \caption@ifinlist{#1}{d,default}{}%
281   \def\caption@position{\caption@defaultpos}%
282 }{\caption@ifinlist{#1}{t,top,above}{%
283   \let\caption@position\@firstoftwo
284 }{\caption@ifinlist{#1}{b,bottom,below}{%
285   \let\caption@position\@secondoftwo
286 }{\caption@ifinlist{#1}{a,auto}{%
287   \let\caption@position\@undefined
288 }{%
289   \PackageError{caption}{Undefined caption position `#1'}{\caption@eh}%
290 }}}
```

`\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^AT_EX document classes.

```

291 %\caption@setdefaultpos{b}% default = bottom
292 \let\caption@defaultpos\@secondoftwo
```

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

```

293 \newcommand\caption@iftop{%
294   \ifx\caption@position\@undefined
295     \expandafter\@secondoftwo
296   \else
297     \expandafter\caption@position
298   \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.

```

299 \newcommand\caption@fixposition{%
300   \ifx\caption@position\@undefined
301     \caption@autoposition
302   \fi}
```

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

```

303 \newcommand\caption@autoposition{%
304   \ifvmode
305     <+debug> \edef\caption@tempa{\the\prevdepth}%
306     <+debug> \PackageInfo{caption}{\protect\prevdepth=\caption@tempa}%
307   % \caption@setposition{\ifdim\prevdepth>-\p@ b\else t\fi}%
308   \ifdim\prevdepth>-\p@
309     \let\caption@position\@secondoftwo
310   \else
311     \let\caption@position\@firstoftwo
312   \fi
313 \else
314 <+debug> \PackageInfo{caption}{no \protect\prevdepth}%
```

```

315 % \caption@setposition{b}%
316 \let\caption@position\@secondoftwo
317 \fi}

```

Hooks

```

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

```

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

```

318 \newcommand\caption@beginhook{}
319 \newcommand\caption@endhook{}
320 \newcommand\AtBeginCaption{\l@addto@macro\caption@beginhook}
321 \newcommand\AtEndCaption{\l@addto@macro\caption@endhook}

```

Miscellaneous options

```

322 \DeclareCaptionOption{listof}{\caption@setbool{lof}{#1}}
323 \DeclareCaptionOption{singlelinecheck}{\caption@setbool{slc}{#1}}
324 \DeclareCaptionOption{strut}{\caption@setbool{strut}{#1}}

```

Debug options

Please note that these options are usually not available.

```

325 <+debug>\DeclareCaptionOption{showposition}{\caption@setbool{showpos}{#1}}
326 <+debug>\captionsetup{showposition=0}

```

Initialization of parameters

```

327 \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.)
328 \newif\ifcaption@star

```

Typesetting the caption

```

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

```

```

\caption@@make \caption@@make{<caption label>}{<caption text>}
331 \newcommand\caption@@make[2]{%
332 \begingroup
333 \caption@beginhook
334 \caption@calcmargin

```

Special single-line treatment (option `singlelinecheck=`)

```

335 \caption@ifslc{\ifx\caption@sls\@empty}else
336 \caption@slc{#1}{#2}\captionwidth\relax
337 \fi}{}%

```

Typeset the left margin (option `margin=`)

```

338 \@tempdima\captionmargin
339 \caption@ifh{\advance\@tempdima by \captionindent}%
340 \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*`.)

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

Typeset the indentation (option `indention=`)

```
344 \caption@ifh{%
345 % \ifdim\captionindent=\z@
346 % \leavevmode
347 % \else
348 % \hskip-\captionindent}%
349 % \fi}%

```

Typeset the caption itself

```
350 \caption@@@make{#1}{#2}%
351 \caption@endbox

```

Typeset the right margin (option `margin=`)

```
352 \@tempdima\captionmargin
353 \advance\@tempdima by \captionmarginx
354 \hskip\@tempdima
355 \caption@endhook
356 \endgroup
357 \global\caption@starfalse}

```

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

```
358 \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.

```
359 \@tempdima\hsize
360 \ifnum\@listdepth>0\relax
361 \advance\@tempdima by -\leftmargin
362 \advance\@tempdima by -\rightmargin
363 \fi
364 \ifdim\captionwidth=\z@
365 \captionwidth\@tempdima
366 \advance\captionwidth by -2\captionmargin
367 \advance\captionwidth by -\captionmarginx
368 \else
369 \captionmargin\@tempdima
370 \advance\captionmargin by -\captionwidth
371 \divide\captionmargin by 2
372 \captionmarginx\z@
373 \fi
374 \<+debug> \PackageInfo{caption}{%
375 \<+debug> \protect\hsize=\the\hsize,
376 \<+debug> \protect\margin=\the\captionmargin,
377 \<+debug> \protect\marginx=\the\captionmarginx,
378 \<+debug> \protect\width=\the\captionwidth}%

```

```

379 }

\caption@slc This one does the single-line-check.
380 \newcommand\caption@slc[4]{%
381   \caption@startslc
382   \sbox\@tempboxa{\caption@@@make{#1}{#2}}%
383   \ifdim\wd\@tempboxa >#3%
384     \caption@endslc
385   \else
386     \caption@endslc
387     \caption@esetup\caption@sls
388     #4%
389   \fi}

\caption@startslc Re-define anything which would disturb the single-line-check.
390 \newcommand\caption@startslc{%
391   \begingroup
392   \let\label\@gobble
393   \let\@footnotetext\@gobble\let\@endnotetext\@gobble
394   \def\stepcounter##1{\advance\c@##1\endcsname\@ne\relax}%
395   \let\caption@hj\relax}

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

\caption@startbox These macros start and end the box which surrounds the caption paragraph.
\caption@endbox 398 \newcommand*\caption@startbox[1]{\vbox\bgroup\hsize#1}%
399 %\newcommand*\caption@startbox[1]{\vbox\bgroup\setlength\hsize{#1}\@parboxrestore}
400 \newcommand*\caption@endbox{\egroup}
401 %\newcommand*\caption@endbox{\@finalstrut\strutbox\@par\egroup}

\caption@@@make \caption@@@make{\caption label}{\caption text}
This one finally typesets the caption paragraph, without margin and indentation.
402 \newcommand\caption@@@make[2]{%
  If the label is empty, we use no caption label separator.
403   \sbox\@tempboxa{#1}%
404   \ifdim\wd\@tempboxa=\z@
405     \let\caption@lsep\relax
406   \fi

  If the text is empty, we use no caption label separator, too.
  Note: Unfortunately this only works under certain circumstances. Therefore an additional
  check inside \caption will be introduced in the upcoming version v3.1 of the caption
  package.
407   \caption@ifempty{#2}{%
408     \let\caption@lsep\relax
409 %   \let\caption@ifstrut\@secondoftwo
410   }%

  Take care that \captionparindent and \captionhangindent will be used to
  typeset the paragraph.
411   \@setpar{\@par\caption@par}\caption@par

```

```

412 \caption@hj\captionsize\captionfont\caption@fmt
413 {\ifcaption@star\else{\captionlabelfont#1}\fi}%
414 {\ifcaption@star\else{\caption@iflfl\captionlabelfont\caption@lsep}\fi}%
415 {{\captiontextfont
416 \caption@ifstrut{\vrule\@height\ht\strutbox\@width\z@}{}}%
417 \nobreak\hskip\z@skip
418 #2%
419 % \caption@ifstrut{\vrule\@height\z@\@depth\dp\strutbox\@width\z@}{}}%
420 \caption@ifstrut{\@finalstrut\strutbox}{}}%
421 \par}}}
```

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

```

422 \newcommand\caption@ifempty[1]{%
423   \def\caption@tempa{#1}%
424   \def\caption@tempb{\ignorespaces}%
425   \ifx\caption@tempa\caption@tempb
426     \let\caption@tempa\@empty
427   \fi
428   \ifx\caption@tempa\@empty
429     \expandafter\@firstofone
430   \else
431     \expandafter\@gobble
432   \fi}

```

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

```
433 \newcommand*\caption@@par{%
434   \parindent\captionparindent\hangindent\captionhangindent}%
```

Identification

```
435 \NeedsTeXFormat{LaTeX2e}[1994/12/01]
436 \ProvidesPackage{caption}[2006/03/21 v3.0j Customising captions (AR)]
437 \ifdefined\debug\PackageWarning{caption}{DEBUG VERSION}
```

```
438 \RequirePackage{caption3}[2006/01/12] % needs v3.0i or newer
```

[illegible]

```
444 \DeclareCaptionOption*{figureposition}{\captionsetup[figure]{position=#1}}
```

```
445 \DeclareCaptionOption*{tableposition}{\captionsetup[table]{position=#1}}
```

caption v1.x compatibility options

```
446 \DeclareCaptionOption*{normal}[]{\caption@setformat{normal}}
447 \DeclareCaptionOption*{isu}[]{\caption@setformat{hang}}
448 \DeclareCaptionOption*{hang}[]{\caption@setformat{hang}}
449 \DeclareCaptionOption*{center}[]{\caption@setjustification{centering}}
450 \DeclareCaptionOption*{anne}[]{\caption@setjustification{centerlast}}
451 \DeclareCaptionOption*{centerlast}[]{\caption@setjustification{centerlast}}

452 \DeclareCaptionOption*{scriptsize}[]{\def\captionfont{\scriptsize}}
453 \DeclareCaptionOption*{footnotesize}[]{\def\captionfont{\footnotesize}}
454 \DeclareCaptionOption*{small}[]{\def\captionfont{\small}}
455 \DeclareCaptionOption*{normalsize}[]{\def\captionfont{\normalsize}}
456 \DeclareCaptionOption*{large}[]{\def\captionfont{\large}}
457 \DeclareCaptionOption*{Large}[]{\def\captionfont{\Large}}

458 \DeclareCaptionOption*{up}[]{\l@addto@macro\captionlabelfont\upshape}
459 \DeclareCaptionOption*{it}[]{\l@addto@macro\captionlabelfont\itshape}
460 \DeclareCaptionOption*{sl}[]{\l@addto@macro\captionlabelfont\slshape}
461 \DeclareCaptionOption*{sc}[]{\l@addto@macro\captionlabelfont\scshape}
462 \DeclareCaptionOption*{md}[]{\l@addto@macro\captionlabelfont\mdseries}
463 \DeclareCaptionOption*{bf}[]{\l@addto@macro\captionlabelfont\bfseries}
464 \DeclareCaptionOption*{rm}[]{\l@addto@macro\captionlabelfont\rmfamily}
465 \DeclareCaptionOption*{sf}[]{\l@addto@macro\captionlabelfont\sffamily}
466 \DeclareCaptionOption*{tt}[]{\l@addto@macro\captionlabelfont\ttfamily}

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

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

Some caption2 v2.x compatibility options

```
470 \DeclareCaptionOption*{flushleft}[]{\caption@setjustification{raggedright}}
471 \DeclareCaptionOption*{flushright}[]{\caption@setjustification{raggedleft}}
472 \DeclareCaptionOption*{oneline}[]{\caption@setbool{slc}{1}}
473 \DeclareCaptionOption*{ignoreLTcapwidth}[]{}
```

Some KOMA-Script compatibility stuff

```
474 \@ifundefined{scr@caption}{}{
475   \DeclareCaptionOption*{onelinecaption}[]{\onelinecaptionstrue}
476   \DeclareCaptionOption*{noonelinecaption}[]{\onelinecaptionsfalse}
477   \DeclareCaptionOption*{tablecaptionabove}[]{\captionsetup[table]{position=t}}
478   \DeclareCaptionOption*{tablecaptionbelow}[]{\captionsetup[table]{position=b}}

\onelinecaptionsfalse
\onelinecaptionstrue
479   \def\onelinecaptionstrue{\caption@setbool{slc}{1}}
480   \def\onelinecaptionsfalse{\caption@setbool{slc}{0}}
```

\captionabove
\captionbelow

Original code:

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

```

481 \def\captionabove{%
482   \caption@setposition{t}\let\caption@setposition\@gobble
483   \scr@caption}
484 \def\captionbelow{%
485   \caption@setposition{b}\let\caption@setposition\@gobble
486 \scr@caption}

487 }

```

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.

```

488 \newcommand*\caption@declarepackage[1]{%
489   \caption@setbool{pkt@#1}{1}%
490   \DeclareCaptionOption*{#1}{\caption@setbool{pkt@#1}{#1}}}
491 \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.)

```

492 \newcommand\caption@ifpackage[3]{%
493   <+debug> \edef\caption@tempa{%
494   <+debug>   \caption@ifbool{pkt@#1}%
495   <+debug>   {\@ifundefined{#2}{\AtBeginDocument}{firstofone}}%
496   <+debug>   {gobble}}%
497   <+debug> \PackageInfo{caption}{#1 = \caption@ifbool{pkt@#1}{1}{0} %
498   <+debug>   {\@ifundefined{#2}{not }{loaded} -> \caption@tempa)}%
499   \caption@ifbool{pkt@#1}{%
500     \@ifundefined{#2}%
501     {\let\caption@tempa\AtBeginDocument}%
502     {\let\caption@tempa\@firstofone}%
503   }{%
504     \let\caption@tempa\@gobble
505   }%
506   \caption@tempa{\@ifundefined{#2}{}{#3}}%
507   \caption@undefbool{pkt@#1}}
508 \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.

```

509 \caption@declarepackage{caption}

```


These are the packages we support:

```

510 \caption@declarepackage{float}
511 \caption@declarepackage{floatrow}
512 \caption@declarepackage{hyperref}
513 \caption@declarepackage{hypcap}
514 \caption@declarepackage{listings}
515 \caption@declarepackage{longtable}
516 \caption@declarepackage{picins}
517 \caption@declarepackage{rotating}
518 \caption@declarepackage{sidecap}
519 \caption@declarepackage{supertabular}

```

`\ProcessOptionsWithKV` We process our options using the keyval package, so we use this one instead of `\ProcessOptions` offered by L^AT_EX 2_ε. (This code was taken from the hyperref package.)

```

520 \def\ProcessOptionsWithKV#1{%
521   \let\@tempc\relax
522   \let\caption@tempa\@empty
523   \@for\CurrentOption:=\@classoptionslist\do{%
524     \@ifundefined{KV@#1@\CurrentOption}{%
525       }{%
526         \@ifundefined{KV@#1@\CurrentOption @default}{%
527           \PackageInfo{#1}{Global option '\CurrentOption' ignored}%
528         }{%
529           \PackageInfo{#1}{Global option '\CurrentOption' processed}%
530           \edef\caption@tempa{\caption@tempa,\CurrentOption,}%
531           \@expandtwoargs\@removeelement\CurrentOption
532             \@unusedoptionlist\@unusedoptionlist
533         }%
534       }%
535     }%
536   \edef\caption@tempa{%
537     \noexpand\caption@setkeys{#1}{%
538       \caption@tempa\@optionlist{\@currname.\@currentx}%
539     }%
540   }%
541   \caption@tempa
542   \let\CurrentOption\@empty
543   \AtEndOfPackage{\let\@unprocessedoptions\relax}}

```

```
544 \ProcessOptionsWithKV{caption}
```

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

```

545 \caption@ifbool{pkt@caption}{}{\endinput}
546 \caption@undefbool{pkt@caption}

```

Useful stuff

```

\captionof \captionof(*){<type>}[<lst_entry>]{<heading>}
547 \def\captionof{\ifstar{\caption@of{\caption*}}{\caption@of{caption}}}
548 \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.

```
549 \DeclareCaptionOption{type}{\def\@capttype{#1}}
```

```
\ContinuedFloat \ContinuedFloat[\langle type \rangle]
```

This mainly decreases the appropriate counter by -1 .

```
550 \providecommand\ContinuedFloat{%
551   \@ifnextchar[%
552     \@ContinuedFloat
553     {\ifx\@capttype\@undefined
554       \latex@error{\noexpand\ContinuedFloat outside float}\@ehd
555     }else
556       \@ContinuedFloat[\@capttype]%
557     \fi}}
558 \def\@ContinuedFloat[#1]{%
559   \addtocounter{#1}\m@ne
560   \caption@ContinuedFloat{#1}%
561   \caption@@ContinuedFloat{#1}}
```

```
\caption@ContinuedFloat \caption@ContinuedFloat{\langle type \rangle}
\caption@resetContinuedFloat \caption@resetContinuedFloat{\langle type \rangle}
```

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.)

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

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

```
564 \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{\langle type \rangle} performs these tasks:
```

- Call `\caption@resetContinuedFloat` (see above) and start a new group
- Execute the options set with `\captionsetup[\langle type \rangle]`
- Define `\fnum@{\langle type \rangle}` 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`)

```
565 \newcommand*\caption@begin[1]{%
566   \caption@resetContinuedFloat{#1}%
567   \begingroup
568   \caption@setfloattype{#1}%
569   \ifx\caption@lfmt\caption@lfmt@default\else
570     \@namedef{fnum@#1}{%
571       \caption@lfmt{\caption@floatname{#1}}{\@nameuse{the#1}}}%
572   \fi
```

```

573 \caption@fixposition
574 \global\let\caption@fixedposition\caption@position}

\caption@beginex \caption@beginex{<type>}{<list entry>}
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.
575 \newcommand\caption@beginex[2]{%
576 \caption@begin{#1}%
577 \caption@iflof%
578 {\def\caption@tempa{#2}}%
579 {\let\caption@tempa\@empty}%
580 \ifx\caption@tempa\@empty
581 \long\def\addcontentsline##1##2##3{}%
582 \fi}

\caption@end \caption@end closes the group.
583 \newcommand*\caption@end{%
584 \endgroup
585 \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.
586 \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{...}.
587 \newcommand*\caption@letfloattype[2]{%
588 \def\caption@setfloattype##1{%
589 \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.
590 \newcommand*\caption@floatname[1]{\@nameuse{#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.

```

591 \AtBeginDocument{%

```

We only patch `\caption` and `\@caption` if the `captcont` package (which brings its 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.

```

592 \ifundefined{cc@caption}{%

\caption Here comes our definition of \caption and \caption*. (We set \caption@startrue
globally so it works with the sidecap package, too.)
593 \let\caption@old\caption
594 \def\caption{\caption@caption\caption@old}%
595 \def\caption@caption#1{%
596 \ifstar{\ContinuedFloat\global\caption@startrue#1[]}{#1}}%

\@caption Our definition of \@caption simply calls the old definition, nested by \caption@beginex
and \caption@end.
597 \let\caption@@old\@caption
598 \long\def\@caption#1[#2]#3{%
599 \caption@beginex{#1}{#2}%
600 \caption@@old{#1}[]{#2}]{#3}%
601 \caption@end}%

602 }{%

Minimum captcont package support:
We define \caption@caption here so it's there but does not make any harm.
603 \PackageInfo{caption}{captcont package v2.0 detected}%
604 \def\caption@caption#1[#1]%
605 }%
606 }

\@makecaption \@makecaption{<label>}{<text>}
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.

```

607 \renewcommand\@makecaption[2]{%
608 \caption@iftop{\vskip\belowcaptionskip}{\vskip\abovecaptionskip}%
609 <+debug> \caption@ifbool{showpos}{%
610 <+debug> \llap{$\caption@iftop\downarrow\uparrow$ }}{}%
611 \caption@@make{#1}{#2}%
612 \caption@iftop{\vskip\abovecaptionskip}{\vskip\belowcaptionskip}}

```

KOMA-Script classes support

```
613 \AtBeginDocument{\@ifundefined{scr@caption}}{%
614   \PackageInfo{caption}{KOMA-Script class detected}%
\scr@caption We update the definition of \scr@caption so it actually reflects our definition of
\caption.
615   \let\scr@caption\caption
616 }}
```

french(le) package support

```
617 \AtBeginDocument{\@ifundefined{f@ffrench}}{%
618   \PackageInfo{caption}{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.
619   \@ifundefined{GOfrench}%
620     {\let\caption@tempa\@firstofone}%
621     {\def\caption@tempa{\g@addto@macro\GOfrench}}%
622   \caption@tempa{%

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

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

\f@ffrench \f@ffrench and \f@tfrench reflect \fnum@figure and \fnum@table when
\f@tfrench 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.
626     \let\@eatDP\@undefined
627     \let\caption@tempa\@empty
628     \ifx\f@ffrench\fnum@figure
629       \l@addto@macro\caption@tempa{\let\fnum@figure\f@ffrench}%
630     \fi
631     \ifx\f@tfrench\fnum@table
632       \l@addto@macro\caption@tempa{\let\fnum@table\f@tfrench}%
633     \fi
634     \def\f@ffrench{\ifx\listoffigures\relax\else\figurename~\thefigure\fi}%
635     \def\f@tfrench{\ifx\listoftables\relax\else\tablename~\thetable\fi}%
636     \caption@tempa

637 }}
```

float package support

The float package usually do not use the L^AT_EX 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.

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

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

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

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

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

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

`\captionof` 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.

```

643 \renewcommand*\caption@of[2]{%
644   \@ifundefined{fst@#2}{}{%
645     \let\caption@of@float\@firstofone
646     \@nameuse{fst@#2}\@float@setevery{#2}}%
647   \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)

```

648 \let\caption@@float\float@caption
649 \long\def\float@caption#1[#2]#3{%
650   \caption@beginex{#1}{#2}%
651   \let\@fs@capt\caption@@make
652   \caption@@float{#1}[{#2}]{#3}%
653   \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.

```

654     \@ifundefined{hyper@@anchor}{}{%
655       \begingroup
656         \@tempdima\prevdepth
657         \nointerlineskip
658         \let\leavevmode\relax
659         \hyper@@anchor\@currentHref\relax
660         \prevdepth\@tempdima
661       \endgroup}%
662   \def\caption@@make##1##2{\unvbox\@floatcapt}%
663   \@makecaption{}{}%
664   \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>]`.

```

665 \let\caption@float@setevery\@float@setevery
666 \def\@float@setevery#1{%
667   \caption@float@setevery{#1}%

```

L^AT_EX 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.

```

668   \def\caption@floatname##1{\@nameuse{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 be 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.)

```

669   \expandafter\let\expandafter\caption@tempa\csname @float@c@#1\endcsname
670   \ifx\caption@tempa\float@caption
671   \else\ifx\caption@tempa\@caption
672   \else\ifx\caption@tempa\caption@@float
673 {+debug}      \PackageInfo{caption}{\protect\@float@c@#1\space := \protect\float@ca
674   \expandafter\let\csname @float@c@#1\endcsname\float@caption
675   \else
676 {+debug}      \PackageInfo{caption}{\protect\@float@c@#1\space := \protect\@caption
677   \expandafter\let\csname @float@c@#1\endcsname\@caption
678   \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`.

```

679   \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`)

```

680   \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.

```

681   \expandafter\let\expandafter\caption@fst\csname fst@#1\endcsname
682   \edef\caption@fst{\noexpand\string\expandafter\noexpand\caption@fst}%
683   \edef\caption@fst{\noexpand\@gobblefour\caption@fst}%
684 %   \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.

```

685   \caption@setstyle*\caption@fst
686   \caption@settype\caption@fst

```



```

687     \fi}%

\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.
688   \g@addto@macro\fs@plaintop{\def\@fs@mid{\vspace\abovecaptionskip\relax}}%
689   \g@addto@macro\fs@boxed{\def\@fs@mid{\kern\abovecaptionskip\relax}}%
690 }

```

The skip between ‘boxed’ floats and their caption defaults to 2pt.

```
691 \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’.

```

692 \caption@ifbool{ruled}{%
693   \captionsetup[ruled]{margin=0pt,singlelinecheck=0}%
694 }{%
695   \DeclareCaptionStyle{ruled}{labelfont=bf,labelsep=space,strut=0}%
696 }
697 \caption@undefbool{ruled}

```

floatrow package support

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

```

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

```

`\caption@of` Captions typeset with `\captionof` should have the correct layout, so we have to ‘activate’ this layout here with `\flrow@setlist`.

(Please note that this version of `\captionof` has the same restrictions than the `\captionof` offered for floating environments defined with the float package, see above.)

```

700   \renewcommand*\caption@of[2]{%
701     \def\@captive{#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.

```

702   \renewcommand*\caption@floatname[1]{%
703     \@nameuse{\@ifundefined{fname@#1}{#1name}{fname@#1}}}%
704 }

```

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.

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

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

```
707 \def\caption@ContinuedFloat#1{%
708   \ifundefined{theH#1}{}{%
709     \ifundefined{CF@#1}{}%
710     \expandafter\newcount\csname CF@#1\endcsname
711     \caption@resetContinuedFloat{#1}{}%
712     \global\advance\csname CF@#1\endcsname\@ne\relax
713     \expandafter\l@addto@macro\csname theH#1\endcsname{%
714       .\expandafter\@arabic\csname CF@#1\endcsname}%
715     \let\caption@resetContinuedFloat\@gobble
716   }%

```

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

```
717 \def\caption@resetContinuedFloat#1{%
718   \ifundefined{CF@#1}{}{\global\csname CF@#1\endcsname\z@\relax}}%
719 }
```

hycap package support

When the hycap package is used the following problems occur:

1. The hycap 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 hycap 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 hycap does not work if `\Hy@hypertexnamesfalse` is set. This come in handy; we set it locally to avoid duplicated hyperref labels which could occur if `\ContinuedFloat` will be used.)
4. `\capstart` will call `\H@refstepcounter` to increase the float number. This collides with a following `\ContinuedFloat`, too, so we have to move this call from here to `\caption`. (Since we set `\Hy@hypertexnamesfalse` we can do this without problems.)

```

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

\capstart Here comes our version of \capstart:
722   \let\caption@capstart\capstart
723   \def\capstart{%
    First of all we update \hc@org@caption to correct the problem that the hypcap pack-
    age has saved an older definition of \caption.
724     \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.
725     \begingroup
726       \let\H@refstepcounter\@gobble
727       \Hy@hypertexnamesfalse
728       \caption@capstart
729       \global\let\caption@currentHref\@currentHref
730     \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 ex-
    pected.
731     \def\caption{%
732       \let\caption\hc@org@caption
733       \H@refstepcounter\@capttype
734       \caption@caption\hc@caption}}%

\hc@@caption Here comes our version of \hc@@caption:
735   \let\caption@hc@@caption\hc@@caption
736   \long\def\hc@@caption#1[#2]#3{%
737     \caption@beginex{#1}{#2}%
    Beside the usual \caption@begin and \caption@end stuff (to support local op-
    tions etc.) we make sure our saved hyperref reference will be used.
738     \let\caption@hyper@makecurrent\hyper@makecurrent
739     \def\hyper@makecurrent\@capttype{%
740       \let\hyper@makecurrent\caption@hyper@makecurrent
741       \global\let\@currentHref\caption@currentHref}%
742     \caption@hc@@caption{#1}[{#2}]{#3}%
743     \caption@end}%
744 }

```

listings package support

```

745 \caption@ifpackage{listings}{lst@MakeCaption}{%
746   \PackageInfo{caption}{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`.

```

747 \let\caption@lst@MakeCaption\lst@MakeCaption
748 \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.
749   \def\caption@autoposition{\caption@setposition{#1}}%
750   \caption@begin{lstlisting}%
751     \caption@lst@MakeCaption{#1}%
752   \caption@end}%
753 }
```

longtable package support

```

754 \caption@ifpackage{longtable}{LT@makecaption}{%
755   \PackageInfo{caption}{longtable package v3.15 (or newer) detected}%
\LT@makecaption \LT@makecaption{<cmd>}{<label>}{<text>}
Original code:
\def\LT@makecaption#1#2#3{%
  \LT@mccl\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}}}
```

```

756 \def\LT@makecaption#1#2#3{%
757   \caption@LT@make{%
  We set \ifcaption@star according the 1st argument.
758   \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`.)

```

759   \caption@letfloattype{longtable}{%
760     \ifdim\LTcapwidth=4in \else
761       \setcaptionwidth\LTcapwidth
762     \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.)

```

763 %   \caption@setdefaultpos{t}%
764   \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.

```
765 \def\caption@autoposition{%
766 \caption@setposition{\ifcase\LT@rows t\else b\fi}}%
767 \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...)

```
768 \vskip-\ht\strutbox
```

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

```
769 \caption@iftop{\vskip\belowcaptionskip}{\vskip\abovecaptionskip}%
770 \caption@@make{#2}{#3}\endgraf
771 \caption@iftop{\vskip\abovecaptionskip}{\vskip\belowcaptionskip}%
772 \caption@end}}%
```

`\caption@LT@make` Typesets the caption as centered `\multicolumn...`

```
773 \newcommand\caption@LT@make[1]{%
774 \LT@mcol\LT@cols c{\hbox to\z@{\hss\parbox[t]{hsize{#1}\hss}}}%
775 }
```

picins package support

```
776 \caption@ifpackage{picins}{piccaption}{%
777 \PackageInfo{caption}{picins package v3.0 (or newer) detected}%
```

`\piccaption` Original code:

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

778 \def\piccaption{\@dblarg\@piccaption}
779 % \def\piccaption{\caption@caption{\@dblarg\@piccaption}}
```

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

```
780 }
```

rotating package support

```
781 \caption@ifpackage{rotating}{@rotcaption}{%
782 \PackageInfo{caption}{rotating package v2.0 (or newer) detected}%
```

`\rotcaption` Make `\rotcaption*` work.

```
783 \def\rotcaption{\let\@makecaption\@makerotcaption\caption}%
784 % \let\@rotcaption\@undefined
```

`\rotcaptionof` Make `\rotcaptionof(*)` work.

```
785 \def\rotcaptionof{\@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.)

```
786 \long\def\@makerotcaption#1#2{%
787   \ifdim\captionwidth=\z@
788     \setcaptionwidth{.8\textheight}%
789     \caption@slc{#1}{#2}{.8\vsiz}{%
790       \let\caption@makerot\caption@@make
791       \setcaptionmargin\z@
792 %       \setlength\captionindent\z@
793 %       \def\caption@startbox##1{\hbox\bgroup\hsize=.8\textheight}%
794 %       \def\caption@endbox{\egroup}%
795 %       (not needed because \rotatebox uses an \hbox anyway)
796       \let\caption@startbox\@gobble
797       \let\caption@endbox\relax}%
798     \caption@setbool{slc}{0}% been there, done that
799   \fi
800   \rotatebox{90}{\caption@makerot{#1}{#2}}%
801   \nobreak\hspace{12pt}}%
802 \newcommand\caption@makerot[2]{%
803   \begin{minipage}\captionwidth\caption@@make{#1}{#2}\end{minipage}}%
804 }
```

sidecap package support

```
805 \caption@ifpackage{sidecap}{endSC@FLOAT}{%
806   \PackageInfo{caption}{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.)

```
807 \@ifundefined{caption@caption}%
808   {\let\caption@tempa\AtBeginDocument}%
809   {\let\caption@tempa\@firstofone}%
810   \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`.

```
811 \let\caption@SC@zfloat\SC@zfloat
812 \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.

```
813 \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.

```
814 \global\let\SC@CAPsetup\@empty
815 \def\captionsetup##1{\g@addto@macro\SC@CAPsetup{,##1}}%
```

Make \caption* work.

```
816 \let\caption@old\caption
817 % \def\caption{\renewcommand\captionsetup[1]{}\caption@caption\caption@old}%
818 \def\caption{\caption@caption\caption@old}%
819 }%
```

\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.

```
820 \let\caption@endSC@FLOAT\endSC@FLOAT
821 \def\endSC@FLOAT{%
```

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

```
822 \let\@captype\SC@captype
```

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

```
823 \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

```
824 \caption@letfloattype{SC\@captype}{%
825 \@listdepth\z@
826 \setcaptionmargin\z@
827 \@ifundefined{SC@justify}{}{%
828 \ifx\SC@justify\@empty\else
829 \let\caption@hj\SC@justify
830 \let\SC@justify\@empty
831 \fi}}%
```

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

```
832 \long\def\caption@ifempty##1{%
833 \ifx\SC@CAPtext\@empty
834 \expandafter\@firstofone
835 \else
836 \expandafter\@gobble
837 \fi}%
```

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

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

```
839 }
```

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.

```
840 \def\caption@setSTposition{%
```

```
841 \caption@setposition{\if@topcaption t\else b\fi}}
```

```
842 \caption@ifpackage{supertabular}{ST@caption}{%
```

```
843 \PackageInfo{caption}{supertabular package detected}%
```

`\tablecaption` Make `\topcaption*` and `\bottomcaption*` work.

```
844 \let\caption@tablecaption\tablecaption
```

```
845 \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}
```

```
846 \let\caption@ST\ST@caption
```

```
847 \long\def\ST@caption#1[#2]#3{\par%
```

```
848 \caption@letfloattype{supertabular}{}%
```

```
849 \let\caption@fixposition\caption@setSTposition
```

```
850 \caption@beginex{#1}{#2}%
```

```
851 \addcontentsline{\csname ext@#1\endcsname}{#1}%
```

```
852 {\protect\numberline{%
```

```
853 \csname the#1\endcsname}{\ignorespaces #2}}%
```

```
854 \@parboxrestore
```

```
855 \normalsize
```

```
856 \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
```

```
857 \caption@end}%
```

```
858 }
```


References

- [1] Frank Mittelbach and Michel Goossens: *The L^AT_EX Companion (2nd. Ed.)*, Addison-Wesley, 2004.
- [2] Anselm Lingnau: *An Improved Environment for Floats*, 2001/11/08
- [3] Olga Lapko: *The floatrow package documentation*, 2005/05/22
- [4] Sebastian Rahtz: *Hypertext marks in L^AT_EX*, 2003/11/30
- [5] Heiko Oberdiek: *The hypcap package – Adjusting anchors of captions* 2001/08/27
- [6] Carsten Heinz: *The Listings Package*, 2004/02/13
- [7] David Carlisle: *The longtable package*, 2000/10/22
- [8] Sebastian Rahtz and Leonor Barroca: *A style option for rotated objects in L^AT_EX*, 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