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

*This package has version number v3.0i, last revised 2006/01/12.

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

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}[2006/01/12]
```

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

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

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

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

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

So

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

and

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

are equal in their results.

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

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

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

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

3 Options

3.1 Formatting

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

The *caption format* determines how this information will be presented; it is specified with the option

`format=`⟨*format name*⟩ ,

having the name of the caption format as its argument.

There are two standard caption formats:

New description v3.0h

`plain`	Typesets the captions as a normal paragraph. (This is the default behaviour, it is adapted from the standard LaTeX document classes.)
`hang`	Indents the caption text, so it will 'hang' under the first line of the text.
`...`	Own formats can be defined using `\DeclareCaptionFormat`. (See section 5: *"Do it yourself"*)

An example: Specifying the option

```
format=hang
```

yields captions like this:

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

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

`indention=`⟨*amount*⟩.

Three examples:

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

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

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

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

`labelformat=` With the option

    labelformat=⟨*label format name*⟩

New description v3.0e

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

`default`	The caption label will be typeset as specified by the document class, usually this means the name and the number (like `simple`). (This is the default behaviour.)
`empty`	The caption label will be empty. This option only makes sense when used together with other options like `labelsep=none`.
`simple`	The caption label will be typeset as a name and a number.
`parens`	The number of the caption label will be typeset in parentheses.
`...`	Own label formats can be defined using `\DeclareCaptionLabelFormat`. (See section 5: *"Do it yourself"*)

An example: Using the options

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

yields captions like this one:

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

`labelsep=` With the options

    labelsep=⟨*label separator name*⟩

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

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

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

Three examples:

```
labelsep=period
```

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

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

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

```
labelsep=endash
```

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

3.2 Justification

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

`justification=`⟨*justification name*⟩ .

You can choose one of the following:

`justified`	Typesets the caption as a normal paragraph. (This is the default.)
`centering`	Each line of the caption will be centered.
`centerlast`	The last line of each paragraph of the caption text will be centered.
`centerfirst`	Only the first line of the caption will be centered.
`raggedright`	Each line of the caption will be moved to the left margin.

`RaggedRight`	Each line of the caption will be moved to the left margin, too. But this time the command `\RaggedRight` of the ragged2e package will be used to achieve this. This difference is that this time the word breaking algorithm of TEX will work inside the caption.
`raggedleft`	Each line of the caption will be moved to the right margin.
`...`	Own justifications can be defined using `\DeclareCaptionJustification`. (See section 5: *"Do it yourself"*)

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 LATEX 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=`
`labelfont=`
`textfont=`

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

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

And these are the available font options:

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

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

If you use only one of these options you can omit the braces; e.g., the options `font={small}` and `font=small` 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=` `width=` For all captions you can specify *either* an extra margin *or* a fixed width. You do this using the options

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

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

Two examples illustrating this:

```
margin=10pt
```

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

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

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

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

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

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

Please compare the spacing in these small tables:

Table 1: A table

A	B
C	D

A	B
C	D

Table 2: A table

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

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

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

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

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

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

Table 3: A table

A	B
C	D

A	B
C	D

Table 4: A table

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

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

E.g.,

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

New feature v3.0a

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

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

is equivalent to

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

4 Useful stuff

`\caption` The command

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

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

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

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

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

looks like

A table

A	B
C	D

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

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

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

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

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

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

typesets captions like this:

Figure 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 enviroments 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:regeln:

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

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

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

New feature v3.0b

If you use one of the above packages together with the caption package you get the additional possibility to set up captions with

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

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

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

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

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

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

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

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

7.1 The float package

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

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

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

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

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

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

```
#include <stdio.h>

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

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

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

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

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

7.2 The listings package

New description v3.0b

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

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

7.3 The longtable package

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

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

7.4 The rotating package

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

7.5 The sidecap package

New description v3.0b

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

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

`listof=`

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

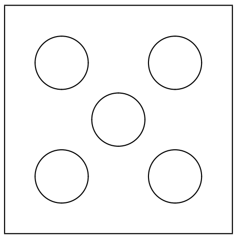

Figure 31: A small example with the caption beside the figure.

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, hvfloat, and nonfloat

8 Compatibility to older versions

8.1 The caption package version 1.*x*

This version of the caption package still supports the old options and commands provided by the version 1.*x* of this package. So there shouldn't occur any problems compiling old documents, but please don't mix old options and commands with the new ones. This isn't supported and can 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 *v*1.*x*	caption *v*3.*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`

caption *v*1.*x*	caption *v*3.*x*
`small`	`font=small`
`normalsize`	`font=normalsize`
`large`	`font=large`
`Large`	`font=Large`
`up`	`labelfont=up`
`it`	`labelfont=it`
`sl`	`labelfont=sl`
`sc`	`labelfont=sc`
`md`	`labelfont=md`
`bf`	`labelfont=bf`
`rm`	`labelfont=rm`
`sf`	`labelfont=sf`
`tt`	`labelfont=tt`

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

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

(See section 5: *"Do it yourself"*)

Setting the margin for captions was done in *v*1.*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*
`flushleft`	`justification=raggedright`
`flushright`	`justification=raggedleft`
`oneline`	`singlelinecheck=on`

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 `\onelinecaptions-false` (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 *v*2.*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 TEX FAQ - Frequently asked questions about TEX and LATEX:

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

- A French FAQ can be found at

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

- epslatex from Keith Reckdahl contains many tips around including graphics in LATEX 2ε 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, and Florian Keiler 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 2ε packages which typesets captions, like the caption package or the subfig package.
The caption package itself redefines the LaTeX commands `\caption`, `\@caption`, and `\@makecaption` and maps the latter one to `\caption@@make`, giving the user the possibility to control the captions of the floating environments `figure` and `table`. Furthermore it does similar to the caption stuff coming from other packages (like the longtable 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 2ε itself or a LaTeX 2ε package and the caption stuff offered by the caption kernel.

11.1 Kernel

Identification

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{caption3}[2006/01/12 v3.0i caption3 kernel (AR)]
⟨+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*⟩`]`.

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

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

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

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

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

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

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

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

would expand to `yes`.

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

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

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

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

```
\def\caption@ifbool#1{\@nameuse{caption@if#1}}
```

```
\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.

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

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

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

`\DeclareCaptionOption` `\DeclareCaptionOption{`⟨*option*⟩`}{`⟨*code*⟩`}`
`\DeclareCaptionOption*{`⟨*option*⟩`}{`⟨*code*⟩`}`
We declare our options using these commands (instead of using `\DeclareOption` offered by LaTeX 2ε), 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.

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

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

```
\newcommand\captionsetup{\@ifnextchar[\caption@setuptype\caption@setup}
\newcommand\caption@typ@{caption@typ@} % This saves 74 words of TeX memory
\def\caption@setuptype[#1]#2{%
  \@ifundefined{\caption@typ@#1}%
    {\@namedef{\caption@typ@#1}{#2}}%
    {\expandafter\l@addto@macro\csname\caption@typ@#1\endcsname{,#2}}}
\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.

```
\newcommand*\caption@setkeys[2]{%
  \let\caption@KV@errx\KV@errx
  \let\caption@KV@err\KV@err
  \def\KV@errx##1{\PackageError\caption@package{##1}\@ehc}%
  \let\KV@err\KV@errx
  \setkeys{#1}{#2}%
  \let\KV@errx\caption@KV@errx
  \let\KV@err\caption@KV@err}
\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.)

```
\newcommand\caption@settype{\@ifnextchar[\caption@@settype\caption@@@settype}
\def\caption@@settype[#1]#2{%
  \def\caption@package{#1}%
  \caption@@@settype{#2}%
  \def\caption@package{caption}}
\newcommand*\caption@@@settype[1]{%
  \@ifundefined{\caption@typ@#1}{}{%
    \caption@esetup{\csname\caption@typ@#1\endcsname}}}
```

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

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

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

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

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

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

Errors

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

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

Margin resp. width

`\captionmargin`
`\captionmarginx`
`\captionwidth`

`\captionmargin` and `\captionwidth` contain the extra margin resp. the total width used for captions. Please never set these values in a direct way, they are just accessible in user documents to provide compatibility to `caption.sty` *v*1.*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`.

```
\newdimen\captionmargin
\newdimen\captionmarginx
\newdimen\captionwidth
```

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

`\setcaptionmargin` `\setcaptionmargin{`⟨*amount*⟩`}`

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

`\setcaptionwidth` `\setcaptionwidth{`⟨*amount*⟩`}`

```
\newcommand\setcaptionwidth{%
  \setlength\captionwidth}
```

Indentions

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

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

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

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

There is an option clash between the KOMA-Script document classes and the caption kernel, both define the options `parindent` and `parskip` but with different meaning. Furthermore the ones defined by the caption kernel take a value as parameter but the KOMA-Script ones do not. So we need special versions of the options `parindent` and `parskip` here, ones who determine if a value is given (and therefore should be treated as our option) or not (and therefore should be ignored by us).

```
  \let\caption@KV@parindent\KV@caption@parindent
  \DeclareCaptionOption{parindent}[]{%
    \def\caption@tempa{#1}%
    \ifx\caption@tempa\@empty
      \PackageInfo{caption3}{Option `parindent' ignored}%
    \else
      \caption@KV@parindent{#1}%
    \fi}%
  \let\caption@KV@parskip\KV@caption@parskip
  \DeclareCaptionOption{parskip}[]{%
    \def\caption@tempa{#1}%
    \ifx\caption@tempa\@empty
      \PackageInfo{caption3}{Option `parskip' ignored}%
    \else
      \caption@KV@parskip{#1}%
    \fi}%
}
```

Styles

`\DeclareCaptionStyle`

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

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

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

`\caption@setstyle`

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

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

`\caption@setdefault`

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

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

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

Formats

`\DeclareCaptionFormat`

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

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

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

`\caption@setformat`

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

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

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

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

‘default’ usually maps to ‘plain’.

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

Label formats

`\DeclareCaptionLabelFormat`

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

```
\newcommand*\DeclareCaptionLabelFormat[2]{%
```

```
  \global\@namedef{caption@lfmt@#1}##1##2{#2}}
\@onlypreamble\DeclareCaptionLabelFormat
```

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

\caption@setlabelformat

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

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

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

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

‘default’ usually maps to ‘simple’.

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

Label separators

eclareCaptionLabelSeparator

`\DeclareCaptionLabelSeparator{`⟨*name*⟩`}{`⟨*code*⟩`}`
`\DeclareCaptionLabelSeparator*{`⟨*name*⟩`}{`⟨*code*⟩`}`
The starred form causes the label separator to be typeset *without* using `\captionlabelfont`.

```
\newcommand\DeclareCaptionLabelSeparator{%
  \@ifstar{\caption@declarelabelseparator\@gobble}%
          {\caption@declarelabelseparator\@firstofone}}
\newcommand\caption@declarelabelseparator[3]{%
  \global\expandafter\let\csname caption@iflf@#2\endcsname#1%
  \global\long\@namedef{caption@lsep@#2}{#3}}
\@onlypreamble\DeclareCaptionLabelSeparator
\@onlypreamble\caption@declarelabelseparator
```

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

\caption@setlabelseparator

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

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

```
\DeclareCaptionLabelSeparator{none}{}
```

```
\DeclareCaptionLabelSeparator{colon}{: }
\DeclareCaptionLabelSeparator{period}{. }
\DeclareCaptionLabelSeparator{space}{ }
\DeclareCaptionLabelSeparator*{quad}{\quad}
\DeclareCaptionLabelSeparator*{newline}{\\}
\DeclareCaptionLabelSeparator*{endash}{\space\textendash\space}
```

‘default’ usually maps to ‘colon’.

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

```
%\@ifundefined{captionseparator}{}{% new v3.1: french(le) support
%  \DeclareCaptionLabelSeparator{default}{\captionseparator\space}}
%\@ifundefined{CaptionSeparator}{}{% new v3.1: frenchb support
%  \DeclareCaptionLabelSeparator{default}{\CaptionSeparator}}
```

Justifications

DeclareCaptionJustification

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

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

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

\caption@setjustification

`\caption@setjustification{`⟨*name*⟩`}`

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

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

These are the pre-defined justification code snippets.

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

‘default’ usually maps to ‘justified’.

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

\caption@centerfirst
\caption@centerlast

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

```
\newcommand\caption@centerfirst{%
  \edef\caption@normaladjust{%
```

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

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

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

`\caption@ragged`

`\caption@ragged` will be basically defined as

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

but with 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.)

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

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

Fonts

`\DeclareCaptionFont`

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

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

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

\caption@setfont `\caption@setfont{`⟨*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.

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

\caption@setdefaultfont `\caption@setdefaultfont{`⟨*command*⟩`}{`⟨*code*⟩`}`
(new v3.1)

```
%\newcommand\caption@setdefaultfont[1]{\long\@namedef{caption#1@default}}
%\@onlypreamble\caption@setdefaultfont
\DeclareCaptionFont{default}{}
```

These are the pre-defined font code snippets.

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

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

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

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

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

```
% new v3.1 (french(le)/frenchb)
%\ifx\captionfont\emph
%% \caption@setdefaultfont{labelfont}{\scshape}
%  \caption@setdefaultfont{textfont}{\em}
%\else
%  \def\@tempa{\itshape\@cfORI}
% \ifx\captionfont\@tempa
%%   \caption@setdefaultfont{labelfont}{\scshape}
%    \caption@setdefaultfont{textfont}{\em}
%  \fi
%\fi
```

Vertical spaces before and after captions

\abovecaptionskip
\belowcaptionskip

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

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

```
\DeclareCaptionOption{aboveskip}{\setlength\abovecaptionskip{#1}}
\DeclareCaptionOption{belowskip}{\setlength\belowcaptionskip{#1}}
\DeclareCaptionOption{skip}{\setlength\abovecaptionskip{#1}}
```

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 `longtable`s, this is doomed to fail, so it's not documented in the user part of the documentation.)

```
\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.*

```
\newcommand*\caption@setposition[1]{%
  \caption@ifinlist{#1}{d,default}{%
    \def\caption@position{\caption@defaultpos}%
  }{\caption@ifinlist{#1}{t,top,above}{%
    \let\caption@position\@firstoftwo
  }{\caption@ifinlist{#1}{b,bottom,below}{%
    \let\caption@position\@secondoftwo
  }{\caption@ifinlist{#1}{a,auto}{%
    \let\caption@position\@undefined
  }{%
    \PackageError{caption}{Undefined caption position `#1'}{\caption@eh}%
  }}}}}
```

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

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

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

```
\newcommand\caption@iftop{%
  \ifx\caption@position\@undefined
    \expandafter\@secondoftwo
  \else
    \expandafter\caption@position
  \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.

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

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

```
\newcommand\caption@autoposition{%
  \ifvmode
⟨+debug⟩    \edef\caption@tempa{\the\prevdepth}%
⟨+debug⟩    \PackageInfo{caption}{\protect\prevdepth=\caption@tempa}%
%    \caption@setposition{\ifdim\prevdepth>-\p@ b\else t\fi}%
    \ifdim\prevdepth>-\p@
      \let\caption@position\@secondoftwo
    \else
      \let\caption@position\@firstoftwo
    \fi
  \else
⟨+debug⟩    \PackageInfo{caption}{no \protect\prevdepth}%
%    \caption@setposition{b}%
    \let\caption@position\@secondoftwo
  \fi}
```

Hooks

`\AtBeginCaption` `\AtEndCaption` `\AtBeginCaption {`⟨*code*⟩`}`
`\AtEndCaption {`⟨*code*⟩`}`
These hooks can be used analogous to `\AtBeginDocument` and `\AtEndDocument`.

```
\newcommand\caption@beginhook{}
\newcommand\caption@endhook{}
```

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

Miscellaneous options

```
\DeclareCaptionOption{listof}{\caption@setbool{lof}{#1}}
\DeclareCaptionOption{singlelinecheck}{\caption@setbool{slc}{#1}}
\DeclareCaptionOption{strut}{\caption@setbool{strut}{#1}}
```

Debug options

Please note that these options are usually not available.

```
⟨+debug⟩\DeclareCaptionOption{showposition}{\caption@setbool{showpos}{#1}}
⟨+debug⟩\captionsetup{showposition=0}
```

Initialization of parameters

```
\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`.)

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

Typesetting the caption

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

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

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

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

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

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

Typeset the left margin (option `margin=`)

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

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

Typeset the indention (option `indention=`)

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

Typeset the caption itself

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

```
  \caption@endbox
```

Typeset the right margin (option `margin=`)

```
  \@tempdima\captionmargin
  \advance\@tempdima by \captionmarginx
  \hskip\@tempdima
```

```
  \caption@endhook
  \endgroup
```

```
  \global\caption@starfalse}
```

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

```
\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.

```
  \@tempdima\hsize
  \ifnum\@listdepth>0\relax
    \advance\@tempdima by -\leftmargin
    \advance\@tempdima by -\rightmargin
  \fi
```

```
  \ifdim\captionwidth=\z@
    \captionwidth\@tempdima
    \advance\captionwidth by -2\captionmargin
    \advance\captionwidth by -\captionmarginx
  \else
    \captionmargin\@tempdima
    \advance\captionmargin by -\captionwidth
    \divide\captionmargin by 2
    \captionmarginx\z@
  \fi
```

```
⟨+debug⟩  \PackageInfo{caption}{%
⟨+debug⟩    \protect\hsize=\the\hsize,
⟨+debug⟩    \protect\margin=\the\captionmargin,
⟨+debug⟩    \protect\marginx=\the\captionmarginx,
⟨+debug⟩    \protect\width=\the\captionwidth}%
```

```
}
```

`\caption@slc` This one does the single-line-check.

```
\newcommand\caption@slc[4]{%
  \caption@startslc
  \sbox\@tempboxa{\caption@@@make{#1}{#2}}%
  \ifdim\wd\@tempboxa >#3%
    \caption@endslc
  \else
    \caption@endslc
    \caption@esetup\caption@sls
    #4%
  \fi}
```

`\caption@startslc` Re-define anything which would disturb the single-line-check.

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

`\caption@endslc` This ends the single-line-check.

```
\newcommand\caption@endslc{%
  \endgroup}
```

`\caption@startbox` `\caption@endbox` These macros start and end the box which surrounds the caption paragraph.

```
\newcommand*\caption@startbox[1]{\vbox\bgroup\hsize#1}%
%\newcommand*\caption@startbox[1]{\vtop\bgroup\hsize#1}% changed v3.1
%\newcommand*\caption@startbox[1]{\vbox\bgroup\setlength\hsize{#1}\@parboxrestore}%
\newcommand*\caption@endbox{\egroup}
%\newcommand*\caption@endbox{\@finalstrut\strutbox\@@par\egroup}
```

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

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

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

```
  \sbox\@tempboxa{#1}%
  \ifdim\wd\@tempboxa=\z@
    \let\caption@lsep\relax
  \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 *v*3.1 of the caption package.

```
  \caption@ifempty{#2}{%
    \let\caption@lsep\relax
%   \let\caption@ifstrut\@secondoftwo
  }%
```

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

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

Finally the caption will be typeset.

```
  \caption@hj\captionsize\captionfont\caption@fmt
    {\ifcaption@star\else{\captionlabelfont#1}\fi}%
    {\ifcaption@star\else{\caption@iflf\captionlabelfont\caption@lsep}\fi}%
    {{\captiontextfont
      \caption@ifstrut{\vrule\@height\ht\strutbox\@width\z@}{}%
      \nobreak\hskip\z@skip
      #2%
%     \caption@ifstrut{\vrule\@height\z@\@depth\dp\strutbox\@width\z@}{}%
      \caption@ifstrut{\@finalstrut\strutbox}{}%
      \par}}}
```

`\caption@ifempty`

`\caption@ifempty{`⟨*text*⟩`}{`⟨*if-clause*⟩`}`
This one tests if the ⟨*text*⟩ is actually empty.
Note: This will be done without expanding the text, therefore this is far away from being bullet-proof.

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

`\caption@@par`

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

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

11.2 Main package

Identification

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{caption}[2006/01/12 v3.0i Customising captions (AR)]
⟨+debug⟩\PackageWarning{caption}{DEBUG VERSION}
```

Loading the caption kernel

```
\RequirePackage{caption3}[2005/12/04]
```

Option for configuration files

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

Options for **`figure`** and **`table`**

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

caption *v*1*.x* compatibility options

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

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

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

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

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

Some caption2 *v*2*.x* compatibility options

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

Some KOMA-Script compatibility stuff

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

  \DeclareCaptionOption*{onelinecaption}[]{\onelinecaptionstrue}
  \DeclareCaptionOption*{noonelinecaption}[]{\onelinecaptionsfalse}
  \DeclareCaptionOption*{tablecaptionabove}[]{\captionsetup[table]{position=t}}
  \DeclareCaptionOption*{tablecaptionbelow}[]{\captionsetup[table]{position=b}}
```

\onelinecaptionsfalse
\onelinecaptionstrue

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

\captionabove
\captionbelow

```
  \def\captionabove{\caption@setposition{t}\scr@caption}
  \def\captionbelow{\caption@setposition{b}\scr@caption}
```

```
}
```

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.

```
\newcommand*\caption@declarepackage[1]{%
  \caption@setbool{pkt@#1}{1}%
  \DeclareCaptionOption*{#1}{\caption@setbool{pkt@#1}{##1}}}
\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.)

```
\newcommand\caption@ifpackage[3]{%
⟨+debug⟩  \edef\caption@tempa{%
⟨+debug⟩    \caption@ifbool{pkt@#1}%
⟨+debug⟩      {\@ifundefined{#2}{AtBeginDocument}{firstofone}}%
⟨+debug⟩      {gobble}}%
⟨+debug⟩  \PackageInfo{caption}{#1 = \caption@ifbool{pkt@#1}{1}{0} %
⟨+debug⟩        (\@ifundefined{#2}{not }{}loaded -> \caption@tempa)}%
  \caption@ifbool{pkt@#1}{%
    \@ifundefined{#2}%
      {\let\caption@tempa\AtBeginDocument}%
      {\let\caption@tempa\@firstofone}%
  }{%
    \let\caption@tempa\@gobble
  }%
  \caption@tempa{\@ifundefined{#2}{}{#3}}%
  \caption@undefbool{pkt@#1}}
\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 superseeded by the subfig *package option* `caption=false` *which causes that only the caption kernel* caption3 *is loaded.*

```
\caption@declarepackage{caption}
```

These are the packages we support:

```
\caption@declarepackage{float}
\caption@declarepackage{floatrow}
\caption@declarepackage{hyperref}
\caption@declarepackage{hypcap}
\caption@declarepackage{listings}
\caption@declarepackage{longtable}
\caption@declarepackage{rotating}
\caption@declarepackage{sidecap}
\caption@declarepackage{supertabular}
```

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

```
\def\ProcessOptionsWithKV#1{%
  \let\@tempc\relax

  \let\caption@tempa\@empty
  \@for\CurrentOption:=\@classoptionslist\do{%
    \@ifundefined{KV@#1@\CurrentOption}%
    {}%
    {%
      \edef\caption@tempa{\caption@tempa,\CurrentOption,}%
      \@expandtwoargs\@removeelement\CurrentOption
        \@unusedoptionlist\@unusedoptionlist
    }%
  }%
  \edef\caption@tempa{%
    \noexpand\caption@setkeys{#1}{%
      \caption@tempa\@ptionlist{\@currname.\@currext}%
    }%
  }%
  \caption@tempa

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

\ProcessOptionsWithKV{caption}
```

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

```
\caption@ifbool{pkt@caption}{}{\endinput}
\caption@undefbool{pkt@caption}
```

Useful stuff

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

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

```
\DeclareCaptionOption{type}{\def\@captype{#1}}
```

\ContinuedFloat `\ContinuedFloat[⟨type⟩]`
This mainly decreases the appropriate counter by -1.

```
\providecommand\ContinuedFloat{%
  \@ifnextchar[%]
    \@ContinuedFloat
    {\ifx\@captype\@undefined
       \@latex@error{\noexpand\ContinuedFloat outside float}\@ehd
     \else
       \@ContinuedFloat[\@captype]%
     \fi}}
\def\@ContinuedFloat[#1]{%
  \addtocounter{#1}\m@ne
  \caption@ContinuedFloat{#1}%
  \caption@@ContinuedFloat{#1}}
```

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

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

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

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

Internal helpers

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

- Call `\caption@resetContinuedFloat` (see above) and start a new group
- Execute the options set with `\captionsetup[⟨type⟩]`
- Define `\fnum@⟨type⟩` if the caption label format is set to non-default

- Override the `position=` setting, if necessary (for example if set to `auto` or used inside a `supertabular`)

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

`\caption@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`.

```
\newcommand\caption@beginex[2]{%
  \caption@begin{#1}%
  \caption@iflof%
    {\def\caption@tempa{#2}}%
    {\let\caption@tempa\@empty}%
  \ifx\caption@tempa\@empty
    \long\def\addcontentsline##1##2##3{}%
  \fi}
```

`\caption@end`

`\caption@end` closes the group.

```
\newcommand*\caption@end{%
  \endgroup
  \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.

```
\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 options 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{...}`.

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

```
\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.

```
\AtBeginDocument{%
```

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

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

`\caption`

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

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

`\@caption`

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

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

```
  }{%
```

Minimum captcont package support:
We define `\caption@caption` here so it's there but does not make any harm.

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

\@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.

```
\renewcommand\@makecaption[2]{%
  \caption@iftop{\vskip\belowcaptionskip}{\vskip\abovecaptionskip}%
⟨+debug⟩  \caption@ifbool{showpos}{%
⟨+debug⟩    \llap{$\caption@iftop\downarrow\uparrow$ }}{}%
  \caption@@make{#1}{#2}%
  \caption@iftop{\vskip\abovecaptionskip}{\vskip\belowcaptionskip}}
```

KOMA-Script classes support

```
\AtBeginDocument{\@ifundefined{scr@caption}{}{%
  \PackageInfo{caption}{KOMA-Script class detected}%
```

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

```
  \let\scr@caption\caption
}}
```

french(le) package support

```
\AtBeginDocument{\@ifundefined{f@ffrench}{}{%
  \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.

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

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

```
    \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.

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

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

```
    \let\@eatDP\@undefined
    \let\caption@tempa\@empty
    \ifx\f@ffrench\fnum@figure
      \l@addto@macro\caption@tempa{\let\fnum@figure\f@ffrench}%
    \fi
    \ifx\f@tfrench\fnum@table
      \l@addto@macro\caption@tempa{\let\fnum@table\f@tfrench}%
    \fi
    \def\f@ffrench{\ifx\listoffigures\relax\else\figurename~\thefigure\fi}%
    \def\f@tfrench{\ifx\listoftables\relax\else\tablename~\thetable\fi}%
    \caption@tempa
```

```
}}}
```

float package support

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

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

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

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

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

Since \float@caption puts the float contents into a savebox we need a special version of \captionof which 'unfolds' this box afterwards, so the caption actually gets typeset. Furthermore we have to typeset the spacing above and below the caption for ourself, since this space is not part of the box.
Please note that this version of \captionof only works *outside* floating environments defined with the float package, so for example a \captionof{Program} used within a 'standard' figure or a minipage will work fine, but not within a re-styled figure or an Example environment defined with \newfloat. (We don't check for this so you'll get wired errors if you try to do so!)

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

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

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

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

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

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

```
      \caption@of@float{%
```

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.

```
      \@ifundefined{hyper@@anchor}{}{%
        \begingroup
          \@tempdima\prevdepth
          \nointerlineskip
          \let\leavevmode\relax
          \hyper@@anchor\@currentHref\relax
          \prevdepth\@tempdima
        \endgroup}%
      \def\caption@@make##1##2{\unvbox\@floatcapt}%
      \@makecaption{}{}}%
  \caption@end}%
```

`\@float@setevery`

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

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

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

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

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

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

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

```
  \expandafter\let\expandafter\caption@tempa\csname @float@c@#1\endcsname
  \ifx\caption@tempa\float@caption
  \else\ifx\caption@tempa\@caption
  \else\ifx\caption@tempa\caption@@float
⟨+debug⟩      \PackageInfo{caption}{\protect\@float@c@#1\space := \protect\float@capt
```

```
      \expandafter\let\csname @float@c@#1\endcsname\float@caption
    \else
⟨+debug⟩      \PackageInfo{caption}{\protect\@float@c@#1\space := \protect\@caption}
      \expandafter\let\csname @float@c@#1\endcsname\@caption
    \fi\fi\fi
```

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

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

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

```
      \caption@setfloatposition
```

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

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

`\fst@Program` will be defined as

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

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

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

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

```
      \caption@setstyle*\caption@fst
      \caption@settype\caption@fst
    \fi}%
```

`\fs@plaintop`
`\fs@boxed`

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

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

}
```

The skip between 'boxed' floats and their caption defaults to `2pt`.

```
\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 *v*1.*x* option `ruled` instead: The current caption settings will be used, but without margin and without ‘single-line-check’.

```
\caption@ifbool{ruled}{%
  \captionsetup[ruled]{margin=0pt,singlelinecheck=0}%
}{%
  \DeclareCaptionStyle{ruled}{labelfont=bf,labelsep=space,strut=0}%
}
\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:

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

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

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

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

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

}
```

hyperref package support

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

$\langle\textit{type}\rangle\,.\,\langle\textit{type \#}\rangle\,.\,\langle\textit{continue \#}\rangle$

will be used instead of

$\langle\textit{type}\rangle\,.\,\langle\textit{type \#}\rangle$.

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

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

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

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

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

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

}
```

hypcap package support

When the hypcap package is used the following problems occur:

1. The hypcap package uses `\capstart`, `\hc@caption`, and `\hc@@caption` instead of `\caption` and `\@caption`. So we have to patch these macros, too.

2. `\caption` will be saved to `\hc@org@caption` when the hypcap package is loaded. We have to change this so our definition of `\caption` will always be used.

3. Both, `\capstart` and `\hc@@caption`, call `\hyper@makecurrent`. But since we offer `\ContinuedFloat` the float counters could have changed between these both calls! So we fix this by saving the hyperref reference (= `\@currentHref`) in `\capstart` and restoring it later on in `\hc@@caption`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
      \let\caption@hyper@makecurrent\hyper@makecurrent
      \def\hyper@makecurrent\@captype{%
        \let\hyper@makecurrent\caption@hyper@makecurrent
        \global\let\@currentHref\caption@currentHref}%
      \caption@hc@@caption{#1}[{#2}]{#3}%
    \caption@end}%
}
```

listings package support

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

`\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`.

```
  \let\caption@lst@MakeCaption\lst@MakeCaption
  \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.

```
    \def\caption@autoposition{\caption@setposition{#1}}%

    \caption@begin{lstlisting}%
      \caption@lst@MakeCaption{#1}%
    \caption@end}%

}
```

longtable package support

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

`\LT@makecaption` `\LT@makecaption{`⟨*cmd*⟩`}{`⟨*label*⟩`}{`⟨*text*⟩`}`
Original code:

```
  \def\LT@makecaption#1#2#3{%
    \LT@mcol\LT@cols c{\hbox to\z@{\hss\parbox[t]\LTcapwidth{%
      % Based on article class "\@makecaption", "#1" is "\@gobble" in star
      % form, and "\@firstofone" otherwise.
      \sbox\@tempboxa{#1{#2: }#3}%
      \ifdim\wd\@tempboxa>\hsize
        #1{#2: }#3%
      \else
        \hbox to\hsize{\hfil\box\@tempboxa\hfil}%
      \fi
      \endgraf\vskip\baselineskip}%
    \hss}}}
```

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

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

```
      \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`.)

```
    \caption@letfloattype{longtable}{%
      \ifdim\LTcapwidth=4in \else
        \setcaptionwidth\LTcapwidth
      \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.)

```
%    \caption@setdefaultpos{t}%
    \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.

```
    \def\caption@autoposition{%
      \caption@setposition{\ifcase\LT@rows t\else b\fi}}%
```

```
    \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...)

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

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

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

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

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

```
}
```

rotating package support

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

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

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

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

```
  \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\vsize
    \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.)

```
  \long\def\@makerotcaption#1#2{%
    \ifdim\captionwidth=\z@
      \setcaptionwidth{.8\textheight}%
      \caption@slc{#1}{#2}{.8\vsize}{%
        \let\caption@makerot\caption@@make
        \setcaptionmargin\z@
%       \setlength\captionindent\z@
%       \def\caption@startbox##1{\hbox\bgroup\hsize=.8\textheight}%
%       \def\caption@endbox{\egroup}%
%         (not needed because \rotatebox uses an \hbox anyway)
        \let\caption@startbox\@gobble
        \let\caption@endbox\relax}%
      \caption@setbool{slc}{0}% been there, done that
    \fi
    \rotatebox{90}{\caption@makerot{#1}{#2}}%
    \nobreak\hspace{12pt}}%
  \newcommand\caption@makerot[2]{%
    \begin{minipage}\captionwidth\caption@@make{#1}{#2}\end{minipage}}%
}
```

sidecap package support

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

```
  \@ifundefined{caption@caption}%
    {\let\caption@tempa\AtBeginDocument}%
    {\let\caption@tempa\@firstofone}%
  \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`.

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

```
  \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.

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

Make `\caption*` work.

```
  \let\caption@old\caption
% \def\caption{\renewcommand\captionsetup[1]{}\caption@caption\caption@old}%
  \def\caption{\caption@caption\caption@old}%
}%
```

\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.

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

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

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

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

```
  \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

```
  \caption@letfloattype{SC\@captype}{%
    \@listdepth\z@
    \setcaptionmargin\z@
    \@ifundefined{SC@justify}{}{%
      \ifx\SC@justify\@empty\else
        \let\caption@hj\SC@justify
        \let\SC@justify\@empty
      \fi}}%
```

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

```
    \long\def\caption@ifempty##1{%
      \ifx\SC@CAPtext\@empty
        \expandafter\@firstofone
      \else
        \expandafter\@gobble
      \fi}%
```

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

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

```
}
```

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.

```
\def\caption@setSTposition{%
  \caption@setposition{\if@topcaption t\else b\fi}}
```

```
\caption@ifpackage{supertabular}{ST@caption}{%
  \PackageInfo{caption}{supertabular package detected}%
```

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

```
  \let\caption@tablecaption\tablecaption
  \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}
```

```
  \let\caption@ST\ST@caption
  \long\def\ST@caption#1[#2]#3{\par%
    \caption@letfloattype{supertabular}{}%
    \let\caption@fixposition\caption@setSTposition
    \caption@beginex{#1}{#2}%
      \addcontentsline{\csname ext@#1\endcsname}{#1}%
```

```
                       {\protect\numberline{%
                           \csname the#1\endcsname}{\ignorespaces #2}}%
      \@parboxrestore
      \normalsize
      \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
    \caption@end}%

}
```

References

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

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

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

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

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

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

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

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

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

[10] Steven D. Cochran: *The subfig package*, 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