

# Reimplementation of $\text{\LaTeX}$ 2 $\epsilon$ 's heading commands using templates

$\text{\LaTeX}$  Project\*

v0.9g 2026-05-25

## Abstract

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Changes to the user interface</b>	<b>3</b>
2.1	New user commands . . . . .	3
2.2	The optional argument . . . . .	3
2.3	Class support . . . . .	4
2.4	Changing heading commands . . . . .	4
2.5	Converting heading commands defined with <code>\@startsection</code> . . . . .	5
2.5.1	Step 1: Getting the template name and the instance values . . . . .	5
2.5.2	Step 2: Get the default template values . . . . .	6
2.6	Step 3: Declare the instances . . . . .	7
2.6.1	Step 4 Redefine the <code>\section</code> command . . . . .	7
<b>3</b>	<b>Setting up <math>\text{\LaTeX}</math> 2<math>\epsilon</math> heading commands in classes</b>	<b>7</b>
<b>4</b>	<b>Open points</b>	<b>8</b>
<b>5</b>	<b>Template types and templates for headings</b>	<b>8</b>
5.1	Template types . . . . .	8
5.1.1	The template type ‘heading’ . . . . .	8
5.1.2	The template type ‘headformat’ . . . . .	9
5.2	Templates . . . . .	10
5.2.1	Templates of type <code>heading</code> . . . . .	10
5.2.2	The <code>heading</code> template ‘ <code>\all</code> ’ . . . . .	10
5.2.3	The <code>heading</code> template ‘display’ . . . . .	11
5.2.4	The <code>heading</code> template ‘runin’ . . . . .	12
5.2.5	Templates of type <code>headformat</code> . . . . .	13
5.2.6	The <code>headformat</code> template ‘hang’ . . . . .	13
5.2.7	The <code>headformat</code> template ‘runin’ . . . . .	14
5.2.8	The <code>headformat</code> template ‘display’ . . . . .	14
5.3	Default instances . . . . .	15

---

\*Initial implementation by Frank Mittelbach.

5.3.1	Instances of <code>heading</code> templates . . . . .	15
5.3.2	Instances of <code>headformat</code> templates . . . . .	15
<b>6</b>	<b>Support for legacy classes and packages</b>	<b>16</b>
6.1	Support classes based on legacy L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> interfaces . . . . .	16
6.2	Support for the <code>titlesec</code> package interfaces . . . . .	16
<b>7</b>	<b>Support for links</b>	<b>16</b>
<b>8</b>	<b>Bookmarks</b>	<b>17</b>
<b>9</b>	<b>Tagging support</b>	<b>17</b>
<b>10</b>	<b>Debugging support</b>	<b>17</b>
<b>11</b>	<b>The Implementation</b>	<b>17</b>
11.1	Debugging . . . . .	17
11.2	Temp variable(s) . . . . .	19
11.3	variable(s) . . . . .	19
11.4	New user commands . . . . .	19
11.5	The L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> parsing interface . . . . .	20
11.6	Templates . . . . .	23
11.6.1	Template types . . . . .	23
11.6.2	heading templates interfaces . . . . .	23
11.6.3	headformat templates interfaces . . . . .	25
11.6.4	heading templates code . . . . .	25
11.6.5	headformat templates code . . . . .	31
11.6.6	Internal commands used by the template code . . . . .	35
11.7	Support for legacy classes and packages . . . . .	38
11.7.1	Instances (sample/default definitions) . . . . .	39
11.7.2	New <code>\@startsection</code> . . . . .	42
11.7.3	New <code>\secdef</code> . . . . .	43
11.7.4	Core L <sup>A</sup> T <sub>E</sub> X . . . . .	44
<b>12</b>	<b>Issues and problems noticed along the way</b>	<b>46</b>
12.1	Local <code>\DeclareInstance</code> . . . . .	46
12.2	<code>\SetCurrentTemplateKeys</code> . . . . .	46
12.3	O-expansion of <code>\UseInstance</code> arguments . . . . .	47
12.4	Switch <code>\openright</code> is not defined by core . . . . .	47
12.5	Indexheading at least for <code>l3doc</code> is wrong now . . . . .	47
	<b>Index</b>	<b>47</b>

# 1 Introduction

This module reimplements tagging aware heading commands with templates. The new implementation is used automatically if `\DocumentMetadata` is used and replaces patches and redefinitions done in the `latex-lab-sec`. In a later step both modules will be merged.

Most of the documentation is of interest only for class and package authors who want to setup their own heading commands but the new implementation changes also the user interface. This is documented first.

## 2 Changes to the user interface

### 2.1 New user commands

`\theheading` This command expands inside a heading to the current number.  
`\partmark` This replaces the fix `\markboth{}{}` or similar in the standard `\part` definition.

### 2.2 The optional argument

The standard heading commands (re)defined by this module use the standard 2e syntax, e.g.,

```
\section[toc]{title}
```

The star-form `\section*{Title}` stops the numbering, toc, bookmark and running header as it was the way for the last 30-odd years. New is that the optional argument is handled as a key/val list if an equal sign at the outer level is detected, otherwise the argument is taken to be the value of the `shorttitle` and passed to the table of contents, the running headers, the bookmarks and used with `\nameref`. Note that this means, that

```
\section*[Title]{Title}
```

will create a toc entry and a bookmark!

The following keys can be used

**toc** This sets the text for the table of contents. It also forces that the entry is created. So

```
\section*[toc=Introduction]{Introduction to this document}
```

will create an unnumbered entry “Introduction” in the table of contents. An empty value (i.e., `toc=`) will suppress the toc entry.

**running** This sets the text for the running header and forces the use of the mark command, so even with a starred section the running header will be updated. An empty value will suppress the call of the mark commands, so a header from a previous section will prevail.

**bookmark** This sets the text for the bookmarks (the PDF outline) if, e.g., `hyperref` is loaded. As with the previous keys an empty value suppresses the bookmark entry.

**nameref** This allows to set the text used for a crossreference with `\nameref`.

**label** This sets a label, so is an alternative to using a `\label` command.

**subtitle, quote** These keys will allow to pass a subtitle and a quote to the underlying code, but currently they are not yet used by the implementations of the standard L<sup>A</sup>T<sub>E</sub>X heading commands.

**shorttitle** This is the key which is used if the optional argument is not of key/val form. So,

```
\section[short]{long title}
```

is the same as

```
\section[shorttitle=short]{long title}
```

The key sets the toc, running, bookmark and nameref text. If **shorttitle** is used together with any of the individual keys (**toc**, **running**, **bookmark**, **nameref**) then they overwrite the default value provided by **shorttitle**.

**numbered, unnumbered** This allow to control the numbering without stopping toc, running head or bookmarks as it would happen with the starred version of the heading command.

**template keys** Any other key present is assumed to be a key that should be passed to the heading instance to overwrite some of its settings. So, e.g.,

```
\section[placement=top,number-format=\fbox{\theheading}]{Title}
```

will force the section to start a new page and put the number into an `\fbox`. For a list of supported keys, check the following parts of the documentation.

## 2.3 Class support

The module redefines all heading commands of the three standard classes, **article**, **report** and **book** to use the new implementation.

Heading commands of other classes that are defined with `\@startsection` are supported (with some restrictions) through a compatibility layer described in the next section.

The heading commands `\part` and `\chapter` are typically defined with `\secdef` and the internal command `\@part` and `\@chapter`. The module redefines `\secdef` to use the standard instances for these commands—this adds tagging support but *changes the layout* until the class provides its own instances.

For the ams-classes, **latex-lab-firstaid** contains instances for `\part` and `\chapter`. They come close to the original layout but are not exactly identical.

Other heading commands defined with `\secdef` will error as the code has no real chance to know how to handle such a heading.

Heading commands which use neither `\@startsection` or `\secdef` (e.g, the `\chapter` command of **memoir**, or the heading commands of KOMA classes) are not changed by this module and so will not be tagged correctly unless they add explicit tagging support.

## 2.4 Changing heading commands

It is possible to change heading commands by editing the instance they use. E.g., in the standard classes one could frame every heading number with

```
\EditInstance{heading}{section}
  {number-format=\fbox{\theheading}}
```

The name of the instance (here `section`) is the same as the heading command.

To support other classes which still setup their heading commands with `\@startsection` the code has a legacy compatibility layer: the `\@startsection` command has been re-defined to setup instances on-the-fly at the first use of a heading command. These instances have names using the the first argument of `\@startsection` with an attached `-@startsection`. As they are created on-the-fly these internal instances can be only edited *after* the first use of the heading command or by declaring (instead of editing) an instance with this name in the preamble. Also as the names of the instances are built from the first argument of `\@startsection` it is not possible to define two different heading commands of the same level with `\@startsection` as that would require two instances with different names. The next section describes how to lift this restrictions by converting such sectioning commands to use the new interfaces.

## 2.5 Converting heading commands defined with `\@startsection`

To convert a heading command defined with `\@startsection` to the new interface suitable instances must be declared. The same needs to happen if one wants to alter the layout of a heading that was defined with `\@startsection` in the document preamble. How this can be done is demonstrated here with the `\section` command of the `ltugboat` class.

### 2.5.1 Step 1: Getting the template name and the instance values

The sectioning commands use two template *types*, `heading` and `headformat`. Compile the following document

```
\DocumentMetadata{}
\documentclass{ltugboat}

\begin{document}

\section{test} % <--- needed so that the instances get defined
\ShowInstanceValues{heading}{section-@startsection}
\ShowInstanceValues{headformat}{section-@startsection}

\end{document}
```

This will show the instance values in the log-file:

```
The instance 'section-@startsection' of type 'heading' has values:
> level => 1
> placement => normal
> mark-cmd => \sectionmark {##1}
> para-indent => false
```

```

> before-vspace => 8.0pt plus 2.0pt minus 2.0pt
> penalty => \c_max_int
> after-penalty-vspace => 0pt
> after-vspace => 4.0pt
> start-code =>
> final-code =>
> heading-decls => \tubsecfmt
> prefix-decls =>
> number-decls =>
> title-decls =>
> subtitle-decls =>
> quote-decls =>
> headformat-instance => section-@startsection
> number-format => \theheading
> contents-extra =>
> name => section
> from template => display.

```

The instance 'section-@startsection' of type 'headformat' has values:

```

> heading-indent => 0pt
> title-format => \tubsecfmt {##1}
> prefix-number-sep => 0pt
> number-title-sep => 1em
> from template => hang.

```

## 2.5.2 Step 2: Get the default template values

The last line in both lists show after the `from template` the names of the templates of each type. That can be used to get the default values of these templates. Compile the following document:

```

\DocumentMetadata{}
\documentclass{ltugboat}

\begin{document}
\ShowTemplateDefaults{heading}{display}
\ShowTemplateDefaults{headformat}{hang}
\end{document}

```

This gives in the log and terminal:

The template 'display' of type 'heading' has default values:

```

> level => 0
> placement => normal
> mark-cmd =>
> para-indent => false
> before-vspace => 0pt
> penalty => \c_max_int
> after-penalty-vspace => 0pt
> after-vspace => 0pt

```

```

> start-code =>
> final-code =>
> heading-decls => \normalfont
> prefix-decls =>
> number-decls =>
> title-decls =>
> subtitle-decls =>
> quote-decls =>
> headformat-instance => std
> number-format => \theheading
> contents-extra => .

```

The template 'hang' of type 'headformat' has default values:

```

> heading-indent => 0pt
> title-format => ##1
> prefix-number-sep => 0pt
> number-title-sep => 1em.

```

## 2.6 Step 3: Declare the instances

By comparing the instance values with the default values one can see which values should be changed in the instance. The names of the new instances can be freely chosen best practice is to use the name of the heading command (without a backslash).

This leads then to these declarations (note that in the `title-format ##1` should be changed into `#1`):

```

\DeclareInstance{heading}{section}{display} % #1=heading, #2=name, #3=template name
{
  level          = 1,
  mark-cmd       = \sectionmark{#1}, % remove second hash
  before-vspace  = 8.0pt plus 2.0pt minus 2.0pt,
  after-vspace   = 4.0pt,
  heading-decls  = \tubsecfmt,
  headformat-instance = section,      % new name
}
\DeclareInstance{headformat}{section}{hang} % #1=heading, #2=name, #3=template name
{
  title-format   = \tubsecfmt {#1}
}

```

### 2.6.1 Step 4 Redefine the \section command

Finally, the heading should be defined to use the new interface.

```

\DeclareDocumentCommand \section {s = {shorttitle} o m}
{ \ParseLaTeXeHeading {section} {#1} {#2} {#3} }

```

## 3 Setting up L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> heading commands in classes

Heading commands are managed by defining an instance of the `heading` template and then using through `\ParseLaTeXeHeading`, e.g.,

```
\DeclareDocumentCommand \part {s ={\shorttitle}o m}
{ \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
```

The name of the `heading` instance is given in the first argument of `\ParseLaTeXeHeading` and it is recommended that classes use the same name as the heading command, to make it easier for users to identify the instance to edit if they want to adapt the command. Examples of instances that mimic the behavior of the heading commands in the standard classes can be found below. Section 2.5 shows how to get instances from commands previously defined with `\@startsection`.

Legacy setup using `\@startsection` remains supported albeit not that performant and with some restrictions for the users, see section 2.4 above and in the documentation below.

In contrast, `\secdef` is only rudimentarily supported. It delegates the layout to two commands which can contain arbitrary code (usually hardwired) so that there is no realistic chance to automatically take this apart and figure out what values should be used for what template parameters. We therefore redefine `\secdef` and map the standard commands `\part` and `\chapter` to the standard instance and error if other uses are detected. Classes using `\secdef` should setup suitable instances that mimic their current layout, an example can be found for the `ams-classes` in `latex-lab-firstaid.dtx`.

## 4 Open points

- There is no good interface yet to change e.g. the font family of all heading commands in one go.
- Support for `titlesec`, see section 6.2.

## 5 Template types and templates for headings

The template(s) for headings expect(s) a large number of positional arguments containing document user data. The document-level command, e.g., `\section` may not offer all of them directly, but may do so via a key value interface or not at all. If no interface is provided then the template is passed `\NoValue`. If it is offered via a key value interface, the template receives whatever is set by the user (and `\NoValue` otherwise).

In my initial implementation I had only the main data as positional arguments, but I came to the conclusion that this scheme here is better going forward.

### 5.1 Template types

The template type `heading` has 10 data arguments, which is more than can be specified as positional arguments. For that reason argument `#9` holds 2 brace groups. The alternative would be to specify such arguments as keys, but for a number of reasons I think the approach to put seldom necessary data arguments all in the last positional argument is actually better (besides being faster).



### 5.1.1 The template type ‘heading’

**Arg: 1** key/value list to alter the default heading parameters

**Arg: 2** unnumbered heading?

**Arg: 3** main title of the heading

**Arg: 4** toc title

**Arg: 5** running title

**Arg: 6** bookmark title

**Arg: 7** nameref text

**Arg: 8** label for the heading in the form `\label{<string>}`

**Arg: 9** { sub title } { quotation }

#### Semantics:

Handles the layout and processing aspects of a heading.

Whether or not the heading is numbered is governed through a boolean, expecting the result of an `s` specification of `\NewDocumentCommand` or equivalent, i.e., expects `\BooleanTrue` or `\BooleanFalse`.

If the title data is also used for bookmarks, toc, running header, and cross references then it has to be given several times which can be arranged for by the parsing interface command that calls the template instance.

If the bookmark, toc, or running argument is set to “empty” then the bookmark, toc or running header should be suppressed by the template. This enables the user on document level to explicitly specify, for example, `[bookmark=]` to suppress the bookmark.

The `nameref` argument is not expected to be empty. Its value should always be used as given if named references are to be generated.

The label argument either holds a `\label` command as provided by the user (or several, see implementation of `\ParseLaTeXeHeading`) or it is empty. I.e., it can be directly executed by a template at the right point where the reference counter (if any) has been set up without the need to check its content.

Argument #9 holds further user data (in brace groups) which are seldom implemented, but if they are the data is available in a positional argument, which then needs to be taken apart using `\@firstoftwo` (for subtitle) or `\@secondoftwo` (for a quotation). If no data is provided `\NoValue` is used to indicate that.

### 5.1.2 The template type ‘headformat’

**Arg: 1** key/value list to alter the default headformat parameters

**Arg: 2** fixed prefix text

**Arg: 3** formatted heading number

**Arg: 4** main title of the heading

**Arg: 5** subtitle

**Arg: 6** quotation

## Semantics:

Handles the layout of just the heading label (prefix and number), main title, subtitle, and quotation but not the spatial relation to previous and following text.

Whether or not the heading is numbered is governed through a boolean, e.g., result of an `s` specification in `\NewDocumentCommand` or equivalent.

If there is no prefix, subtitle or quotation then this is indicated with `\NoValue`.

The template can ignore the `\fixed prefix text` in certain circumstance (or even always). The templates currently implemented to mimic  $\text{\LaTeX} 2_{\epsilon}$  behavior do so in case of unnumbered headings. Same for `\subtitle` and `\quotation`: both are always ignored by the currently defined templates.

Note: instead of unnumbered + counter we could just have a single argument containing the number representation (or `\NoValue` if not used). The reason that there are two is that this allows us to continue to support `\@secntformat`, but I'm not sure this is a good enough reason, so perhaps this is going to change eventually.

## 5.2 Templates

### 5.2.1 Templates of type heading

There are a number of keys that are expected to be recognized by all heading templates (though they may choose not to make use of them). These are listed below instead of being repeated on the actual templates.

All other keys are either attached to the `heading` or to the `headformat` templates. Those that typically vary from heading instance to the next (e.g., `heading-decls`) are all declared in the `heading` templates even if they are actually only used within `headformat` templates. In other words, `headformat` templates have a number of implicit variables that they expect to be set.<sup>1</sup>

### 5.2.2 The heading template ‘`\all`’

#### Attributes:

**name** (*tokenlist*) Referencable name of the heading instance. String that is acceptable in csnames for use in building counter names, etc.

**parent-name** (*tokenlist*) Name of the next higher heading instance. If not given, then the internal heading level of the heading instance is set to 0

**reset-counter** (*tokenlist*) Name of the heading instance that should reset the numbering of this heading level (if any)

**level** (*integer*) Sets the internal heading-level rather than deducing it from **parent-name**. Can be used to specify the top-level heading if not 0, or all headings in legacy implementations, e.g., through `\@startsection`

**placement** (*choice*) Set the heading placement, i.e., the behavior of the heading with respect to page breaks. Allowed values are **page** (heading forms a page if its own), **top** (heading starts a new page), **normal** (heading can appear anywhere on the page). Further possibilities might be **rectopage** and **rectotop** if we implement that. Default: **normal**

---

<sup>1</sup>Maybe questionable

**start-code** (*tokenlist*) Default value is set by the **placement** key. Executed before the heading starts, so can issue, for example, a `\clearpage`

**final-code** (*tokenlist*) Default value is set by the **placement** key. Executed after the heading is typeset. Can set up code for putting the heading on a page by its own, or arrange for paragraph handling of a following paragraph, etc.

**prefix** (*tokenlist*) A fixed string, such as “Chapter”, that can be used together with the number (if any) to form a “heading label”. Usage and placement is up to the template, so it could be placed after the number by the template (in which case it isn’t really a prefix). Default: `\NoValue`

**mark-cmd** (*function(1)*) Function that receives the `<running>` argument and creates a suitable mark insertion Default: `\<name>mark`

**Semantics & Comments:** The above keys should be implemented by all heading templates.

At the moment I have retained the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> interfaces for marks, e.g., one has to set up `\chaptermark`, `\sectionmark`, etc. but I’m not sure this should stay (even though it is certainly simpler from a compatibility perspective).

All templates should set up `\theheading` to correspond to `\the<name>`.

### 5.2.3 The heading template ‘display’

**Attributes:**

**para-indent** (*boolean*) Should the paragraph after the heading be indented?  
Default: `false`

**before-vspace** (*skip*) Vertical space before the heading if there is no page or column break. If there is one it vanishes. In particular this means it will not be used in the heading **placements** `page` or `top` Default: `0pt`

**penalty** (*integer*) Penalty to break before the heading. The default (T<sub>E</sub>X’s largest integer) indicates that no penalty was set in which case `\@secpenalty` is used.  
Default: `1073741823`

**after-penalty-vspace** (*skip*) Vertical space before the heading but after the penalty for the heading. If the penalty results in a page or column break, this space remains at the top of the page Default: `0pt`

**after-vspace** (*skip*) Vertical space after the heading Default: `0pt`

**heading-decls** (*tokenlist*) Declarations (such as font or color settings) applied to all heading elements, i.e., number, title, subtitle, and quotation, if present. Note that color commands (unless `luacolor` is used) introduce a break point before the heading and therefore one should either surround color commands with `\SaveLastSkip` and `\RestoreLastSkip` or to use the keys **prefix-decls**, **number-decls**, **title-decls**, etc. instead. Default: `\normalfont`

**prefix-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading prefix, overwriting the setting of **heading-decls**. Default: `<empty>`

**number-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading number, overwriting the setting of **heading-decls**. Default: *<empty>*

**title-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading title, overwriting the setting of **heading-decls**. Default: *<empty>*

**subtitle-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading subtitle, overwriting the setting of **heading-decls**. Default: *<empty>*

**quote-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading quote, overwriting the setting of **heading-decls**. Default: *<empty>*

**number-format** (*function(1)*) Code that produces a formatted version of the heading number including any ornamentations. Its argument is the counter name for the head. However, instead of using a specific counter representations, e.g., `\thesection`, it can refer to the counter representation for the current heading counter via `\theheading` and ignore the argument. Default: `\theheading`

**contents-extra** (*tokenlist*) Code containing `\addcontents` calls to write to files like `.lot` or `.lot` Default: *<empty>*

**headformat-instance** (*instance*) Template instances of type **headformat** Default: `std`

**Semantics & Comments:** Several of the key names are simply bad and need revision!

## 5.2.4 The heading template ‘runin’

### Attributes:

**before-vspace** (*skip*) Vertical space before the heading if there is no page or column break. If there is one it vanishes. Default: `0pt`

**penalty** (*integer*) Penalty to break before the heading. The default (TeX’s largest integer) indicates that no penalty was set in which case `\@secpenalty` is used. Default: `1073741823`

**after-penalty-vspace** (*skip*) Vertical space before the heading but after the penalty for the heading. If the penalty results in a page or column break, this space remains at the top of the page. It is also applied if the heading is a **page** or **top** heading. Default: `0pt`

**after-vspace** (*skip*) Vertical space after the heading – ignored in runin headings so should be dropped! Default: `0pt`

**heading-decls** (*tokenlist*) Declarations (such as font or color settings) applied to all heading elements, i.e., number, title, subtitle, and quotation, if present. Default: `\normalfont`

**prefix-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading prefix, overwriting the setting of **heading-decls**. Default: *<empty>*

**number-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading number, overwriting the setting of **heading-decls**. Default: *<empty>*

**title-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading title, overwriting the setting of **heading-decls**. Default: *<empty>*

**subtitle-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading subtitle, overwriting the setting of **heading-decls**. Default: *<empty>*

**quote-decls** (*tokenlist*) Declarations (such as font or color settings) applied to the heading quote, overwriting the setting of **heading-decls**. Default: *<empty>*

**number-format** (*function(1)*) Code that produces a formatted version of the heading number including any ornamentations. Its argument is the counter name for the head. However, instead of using a specific counter representations, e.g., `\thesection`, it can refer to the counter representation for the current heading counter via `\theheading` and ignore the argument. Default: `\theheading`

**headformat-instance** (*instance*) Template instances of type **headformat** Default: `std`

**Semantics & Comments:** Several of the key names are simply bad and need revision!

### 5.2.5 Templates of type **headformat**

At the moment all templates of this type assume that placement of prefix, number, title is done in exactly this order. Anything else would require defining further templates.

TODO: consider a more versatile mechanism (like in theorem-like templates) to allow for more flexible placements without the need to define additional templates.

### 5.2.6 The **headformat** template ‘hang’

#### Attributes:

**heading-indent** (*dimen*) Horizontal space before the heading (shifting it to the right in a LR typesetting context) Default: `0pt`

**title-format** (*tokenlist*) Code executed directly before the heading is typeset and after the vertical spacing is done. The code takes an argument, e.g., `\MakeUppercase` Default: *<#1>*

**prefix-number-sep** (*tokenlist*) Token list that can be used in the assignment to a  $\text{\TeX}$  dimension (can contain `em` or `ex` values) specifying the separation between title prefix (if used) and title number. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font. Default: `.5em`

**number-title-sep** (*tokenlist*) Token list that can be used in the assignment to a  $\text{\TeX}$  dimension (can contain `em` or `ex` values) specifying the separation between title number and title text. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font. Default: `1em`

**Semantics & Comments:** Implements a heading layout where number and title start on the same line and the title is hanging off from that if the title has more than one line. It is what L<sup>A</sup>T<sub>E</sub>X always used by default for `\section`, `\subsection`, and `\subsubsection`.

Several of the key names are simply bad and need a revision!

### 5.2.7 The headformat template ‘runin’

**Attributes:**

**heading-indent** (*dimen*) Horizontal space before the heading (shifting it to the right in a LR typesetting context) Default: 0pt

**title-format** (*tokenlist*) Code executed directly before the heading is typeset and after the vertical spacing is done. The code can take an argument, e.g., `\MakeUppercase` Default: `\empty`

**prefix-number-sep** (*tokenlist*) Token list that can be used in the assignment to a T<sub>E</sub>X dimension (can contain `em` or `ex` values) specifying the separation between title prefix (if used) and title number. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font. Default: `.5em`

**number-title-sep** (*tokenlist*) Token list that can be used in the assignment to a T<sub>E</sub>X dimension (can contain `em` or `ex` values) specifying the separation between title number and title text. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font.

In the hang template it is a horizontal dimension!

Default: 1em

**Semantics & Comments:** Implements a heading layout where number and title start on the same line but then continue with the following paragraph on the same line (or the next if the title overflows, i.e., the title is run-in. It is what L<sup>A</sup>T<sub>E</sub>X always used by default for `\paragraph` and `\subparagraph`.

Several of the key names are simply bad and need a revision!

### 5.2.8 The headformat template ‘display’

**Attributes:**

**heading-indent** (*dimen*) Horizontal space before the heading (shifting it to the right in a LR typesetting context) Default: 0pt

**title-format** (*tokenlist*) Code executed directly before the heading is typeset and after the vertical spacing is done. The code can take an argument, e.g., `\MakeUppercase` Default: `\#1`

**prefix-number-sep** (*tokenlist*) Token list that can be used in the assignment to a T<sub>E</sub>X dimension (can contain `em` or `ex` values) specifying the separation between title prefix (if used) and title number. Evaluated after fonts for title text have been set up, i.e., the `em` will be based on the then current font. Default: `.5em`

**number-title-sep** (*tokenlist*) Token list that can be used in the assignment to a  $\TeX$  dimension (can contain **em** or **ex** values) specifying the separation between title number and title text. Evaluated after fonts for title text have been set up, i.e., the **em** will be based on the then current font.

In the display template it is a vertical dimension!

Default: 20 pt

**Semantics & Comments:** Implements a heading layout where number and title are vertically separated. It is what  $\LaTeX$  always used by default for `\chapter` and `\part`. Several of the key names are simply bad and need a revision!

### 5.3 Default instances

These instances are set up to mimic the design of the standard  $\LaTeX$  classes. For other classes they could be adjusted by changing the instance values and/or by basing them on a different template.

#### 5.3.1 Instances of heading templates

**part** (instance of display template) Used for the `\part` command.

**chapter** (instance of display template) Used for the `\chapter` command.

**section** (instance of display template) Used for the `\section` command.

**subsection** (instance of display template) Used for the `\subsection` command.

**subsubsection** (instance of display template) Used for the `\subsubsection` command.

**paragraph** (instance of runin template) Used for the `\paragraph` command.

**subparagraph** (instance of runin template) Used for the `\subparagraph` command.

#### 5.3.2 Instances of headformat templates

**part** (instance of display template) Used in heading instance for `\part`.

**chapter** (instance of display template) Used in heading instance for `\chapter`. By default identical to the one used for `\part`.

**std** (instance of hang template) Used as default in the heading template if nothing else is specified.

**section** (instance of hang template) Used in heading instance for `\section`. By default identical to the **std** instance.

**subsection** (instance of hang template) Used in heading instance for `\subsection`. By default identical to the **std** instance.

**subsubsection** (instance of hang template) Used in heading instance for `\subsubsection`. By default identical to the **std** instance.

**paragraph** (instance of runin template) Used in heading instance for `\paragraph`.

**subparagraph** (instance of runin template) Used in heading instance for `\subparagraph`.

## 6 Support for legacy classes and packages

### 6.1 Support classes based on legacy L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> interfaces

`\@startsection` The `\@startsection` command has the following syntax:

```
\@startsection{name}{level}{indent}{beforeskip}{afterskip}{style}
```

with a special logic that the sign of `<beforeskip>` determines display or runin heading and that of `<afterskip>` whether or not the next paragraph is indented.

All arguments can be cleanly mapped to `heading` and `headformat` templates. Thus, if encountered suitable instances are declared unless already existing.

`\secdef` In contrast `\secdef` only does the parsing and delegates the layout to two commands which can contain arbitrary code (usually hardwired) so that there is no realistic chance to take this apart and figure out what values should be used for what template parameters.

We therefore do not attempt to model this, but instead just use the standard layout from L<sup>A</sup>T<sub>E</sub>X's standard classes for `\chapter` and `\part` if we can identify that the `\secdef` is used to define one of these.

Maybe we can try at some stage to get some static analysis tool going.

### 6.2 Support for the `titlesec` package interfaces

TODO

`\titleformat` The `\titleformat` declaration has the following syntax:

```
\titleformat{cmd}[shape]{format}{label}{sep}{before-code}[after-code]
```

`\titlespacing` The `\titlespacing` declaration has the following syntax:

```
\titlespacing*{cmd}{left-sep}{before-vspace}{after-vspace}[right-sep]
```

## 7 Support for links

All heading commands (numbered and unnumbered) should contain support for active links directly. `hyperref` does not patch the new heading commands! As `\refstepcounter` is not used there is no automatic target.

This means that all definitions should contain at an appropriate place a `\MakeLinkTarget` command. Typically numbered heading commands will use `\MakeLinkTarget{<counter>}` while unnumbered heading commands use `\MakeLinkTarget[<counter>]{}`.

The definition must take care that the target is not separated from the heading by a page break. It also should not affect spacing. The target should be placed so that a jump to the target gives a satisfying user experience, in most cases the best places is at the left margin a bit above the heading.

The targets create also structure destinations that are used by the tagging code. It is therefore important that the targets are in the right place in relation to the tagging commands.



## 8 Bookmarks

Bookmarks are created by the `\addcontentsline` command, more precisely in the new latex-lab toc code a hook with arguments is inserted at the begin of the command which contains the command that creates the bookmark. The arguments of `\addcontentsline` and so also of the hook are the type of the toc, the level name and the (short-)title of the heading. To pass a differing bookmark text the code uses `\texorpdfstring` in the `\addcontentsline`.

UFi:check if this is still the implementation ...

## 9 Tagging support

Tagging of heading commands has to do two tasks:

- surround the whole section (heading and text) with a `Sect` structure
- tag the heading with an `Hn` structure.

This is done with tagging sockets which are documented in the code and in `latex-lab-sec`.

## 10 Debugging support

---

```
\DebugHeadingsOn
\DebugHeadingsOff
\head_debug_on:
\head_debug_off:
```

---

These commands enable/disable debugging messages.

## 11 The Implementation

```
1 <*package>
2 <@@=head>

3 \ProvidesExplPackage
4   {latex-lab-testphase-sec-template}
5   {\ltlabsecIIdate}
6   {\ltlabsecIIversion}
7   {heading implementation}
```

### 11.1 Debugging

```
\g__head_debug_bool

8 \bool_new:N \g__head_debug_bool

(End of definition for \g__head_debug_bool.)
```

```

    \__head_debug:n
\__head_debug_typeout:n
9 \cs_new_eq:NN \__head_debug:n \use_none:n
10 \cs_new_eq:NN \__head_debug_typeout:n \use_none:n

(End of definition for \__head_debug:n and \__head_debug_typeout:n.)

```

```

    \head_debug_on:
    \head_debug_off:
\__head_debug_gset:
11 \cs_new_protected:Npn \head_debug_on:
12 {
13     \bool_gset_true:N \g__head_debug_bool
14     \__head_debug_gset:
15 }

16 \cs_new_protected:Npn \head_debug_off:
17 {
18     \bool_gset_false:N \g__head_debug_bool
19     \__head_debug_gset:
20 }

21 \cs_new_protected:Npn \__head_debug_gset:
22 {
23     \cs_gset_protected:Npe \__head_debug:n ##1
24     { \bool_if:NT \g__head_debug_bool {##1} }
25     \cs_gset_protected:Npe \__head_debug_typeout:n ##1
26     { \bool_if:NT \g__head_debug_bool { \typeout{[head]~ ##1} } }
27 }

(End of definition for \head_debug_on:, \head_debug_off:, and \__head_debug_gset:. These functions
are documented on page 17.)

```

```

    \DebugHeadingsOn
    \DebugHeadingsOff
28 \cs_new_protected:Npn \DebugHeadingsOn { \head_debug_on: }
29 \cs_new_protected:Npn \DebugHeadingsOff { \head_debug_off: }

30 \DebugHeadingsOff

```

(End of definition for \DebugHeadingsOn and \DebugHeadingsOff. These functions are documented on page 17.)

```

\__head_show_arguments:nnnnnn
Some debug data ...
31 \cs_new:Npn \__head_show_arguments:nnnnnn #1#2#3#4#5#6 {
32     \__head_debug_typeout:n{----- Headformat~ instance~ arguments:}
33     \__head_debug_typeout:n{1:~ keys~ =~ \exp_not:n{#1}}
34     \__head_debug_typeout:n{2:~ prefix~ =~ \exp_not:n{#2}}
35     \__head_debug_typeout:n{3:~ number~ =~ \exp_not:n{#3}}
36     \__head_debug_typeout:n{4:~ title~ =~ \exp_not:n{#4}}
37     \__head_debug_typeout:n{5:~ subtitle~ =~ \exp_not:n{#5}}
38     \__head_debug_typeout:n{6:~ quotation~ =~ \exp_not:n{#6}}
39 }

(End of definition for \__head_show_arguments:nnnnnn.)

```

`\_head_show_arguments:nnnnnnnnn`

Some debug data ...

```

40 \cs_new:Npn \_head_show_arguments:nnnnnnnnn #1#2#3#4#5#6#7#8#9 {
41   \_head_debug_typeout:n{-----~ Heading~ instance~ arguments:}
42   \_head_debug_typeout:n{1:~ keys~ =~ \exp_not:n{#1}}
43   \_head_debug_typeout:n{2:~ unnumbered~ =~ \exp_not:n{#2}}
44   \_head_debug_typeout:n{3:~ title~ =~ \exp_not:n{#3}}
45   \_head_debug_typeout:n{4:~ toc~ =~ \exp_not:n{#4}}
46   \_head_debug_typeout:n{5:~ running~ =~ \exp_not:n{#5}}
47   \_head_debug_typeout:n{6:~ bookmark~ =~ \exp_not:n{#6}}
48   \_head_debug_typeout:n{7:~ nameref~ =~ \exp_not:n{#7}}
49   \_head_debug_typeout:n{8:~ label~ =~ \exp_not:n{#8}}
50   \_head_debug_typeout:n{9:~ subtitle | quote ~ =~ \exp_not:n{#9}}
51 }

```

*(End of definition for \\_head\_show\_arguments:nnnnnnnnn.)*

## 11.2 Temp variable(s)

`\l__head_tmpa_tl`

```

52 \tl_new:N\l__head_tmpa_tl

```

*(End of definition for \l\_\_head\_tmpa\_tl.)*

## 11.3 variable(s)

These token lists are automatically declared by the key machinery.

```

53 %\tl_new:N \l__head_bookmark_tl
54 %\tl_new:N \l__head_running_tl
55 %\tl_new:N \l__head_toc_tl
56 %\tl_new:N \l__head_nameref_tl
57 %\tl_new:N \l__head_subtitle_tl
58 %\tl_new:N \l__head_quote_tl
59 %\bool_new:N \l__head_unnumbered_bool

```

`\l__head_label_tl`

`\l__head_instance_keys_tl`

`\l__head_typeset_number_tl`

`\l__head_saved_secnumdepth_tl`

`\l__head_title_tl`

`\l__head_placement_tl`

But this token lists needs a declaration:

```

60 \tl_new:N\l__head_label_tl
61 \tl_new:N\l__head_instance_keys_tl
62 \tl_new:N\l__head_typeset_number_tl
63 \tl_new:N\l__head_saved_secnumdepth_tl
64 \tl_new:N \l__head_title_tl
65 \tl_new:N \l__head_placement_tl

```

*(End of definition for \l\_\_head\_label\_tl and others.)*

## 11.4 New user commands

`\theheading`

This command expands to `\thechapter`, `\thesection` etc in every heading command.

```

66 \tl_new:N\theheading

```

*(End of definition for \theheading. This function is documented on page ??.)*

`\partmark` This replaces the fix `\markboth{}{}` or similar in the standard `\part` definition. As the command is already defined in some classes (like `memoir`), we provide it through a hook.

```
67 \AddToHook{class/after}{\providecommand*\partmark[1]{\markboth{}{}}}
```

(End of definition for `\partmark`. This function is documented on page ??.)

## 11.5 The L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> parsing interface

L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> provides heading commands with a starred form (unnumbered) and one optional argument (to specify an alternative toc/running head). We augment this slightly by supporting a key value interface in the optional argument. This is handled by defining the commands through `\ParseLaTeXeHeading`. This should happen in the document class.

`\ParseLaTeXeHeading`

`\ParseLaTeXeHeading` arguments:

- 1: instance name to use (of type “heading”)
- 2: numbered? boolean
- 3: shorttitle or key/val for layout adjustments and individual title settings
- 4: main title

This command handles the document input that may be hidden in the optional argument, i.e., special data for toc, running (header), bookmark, label, and for more specialized headings subtitle and quote. It also handles the legacy case that the optional argument is just an alternate title to be used for toc, running, and bookmark (through the key shorttitle to which it gets converted).

```
68 \cs_new_protected:Npn \ParseLaTeXeHeading #1 #2 #3 #4 {
69   \__head_debug_typeout:n{=====}
70   \__head_debug_typeout:n{#1:~ \IfBooleanT{#2}{*}
71     \IfValueT{#3}{[\exp_not:n{#3}]}
72     {\exp_not:n{#4}}}
```

We first check if the title argument contains a `\label` command. If yes, we remove it and add it to `\l__head_label_tl` (and if more than one all of them) and save the rest of the main title in `\l__head_title_tl` for later use. If there was no `\label` command then `\l__head_title_tl` is set to #4.

```
73   \__head_find_label:w #4 \label\q_no_value \__head_find_label:w
```

By default toc, running, and bookmark use the data provided by the main title. However, if the second argument is true (i.e., a star was given) they are all suppressed (because this is the L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> logic).

```
74   \IfBooleanTF{#2}
75   {
76     \tl_clear:N \l__head_toc_tl
77     \tl_clear:N \l__head_running_tl
78     \tl_clear:N \l__head_bookmark_tl
79     \bool_set_true:N \l__head_unnumbered_bool
80   }
81   {
```

```

82     \tl_set_eq:NN \l__head_toc_tl      \l__head_title_tl
83     \tl_set_eq:NN \l__head_running_tl  \l__head_title_tl
84     \tl_set_eq:NN \l__head_bookmark_tl \l__head_title_tl
85     \bool_set_false:N \l__head_unnumbered_bool
86 }

```

The nameref always starts out matching the main title.

```

87     \tl_set_eq:NN \l__head_nameref_tl \l__head_title_tl

```

Normally we don't have a subtitle or quote unless they are explicitly given through keys, so we start with `\c_novalue_tl`

```

88     \tl_set_eq:NN \l__head_subtitle_tl \c_novalue_tl
89     \tl_set_eq:NN \l__head_quote_tl    \c_novalue_tl

```

If the optional argument is empty we set the `\l__head_instance_keys_tl` to empty, otherwise process the key/val list setting the keys defined in the module “head”. This overwrites the defaults specified above if keys like “toc” are in the list. All the key/vals unknown by that module are put into `\l__head_instance_keys_tl` which may or may not make this token list non-empty.

If the user has a `label` key and also a `\label` in the main title argument then the latter is ignored (no error checking for that). We could alternatively set all labels we find, perhaps that is the better approach?

```

90     \IfNoValueTF { #3 }
91     { \tl_clear:N \l__head_instance_keys_tl }
92     { \keys_set_known:nnN {heading} { #3 } \l__head_instance_keys_tl }

```

Finally we call the heading instance using the collected mandatory arguments. To simplify downstream processing we pass the values not the container tokenlists resulting from the above processing.<sup>2</sup>

```

93     \__head_debug_typeout:n{use~ 'heading'~ instance:~ #1 }
94     \use:e {
95         \exp_not:N \UseInstance{heading}{#1}
96         { \exp_not:o \l__head_instance_keys_tl }
97         { \bool_if:NTF \l__head_unnumbered_bool \BooleanTrue \BooleanFalse }
98         { \exp_not:o \l__head_title_tl }
99         { \exp_not:o \l__head_toc_tl }
100        { \exp_not:o \l__head_running_tl }
101        { \exp_not:o \l__head_bookmark_tl }
102        { \exp_not:o \l__head_nameref_tl }
103        { \exp_not:o \l__head_label_tl }
104        { { \exp_not:o \l__head_subtitle_tl }
105          { \exp_not:o \l__head_quote_tl } }
106    }
107 }

```

Syntax keys that can appear in the optional argument of headings parsed by `\ParseLaTeXeHeading`. All other keys used there are passed to the heading instance to overwrite instance setting.

```

108 \keys_define:nn {heading} {
109     , shorttitle .meta:n =
110     {toc = {#1} , running = {#1} , bookmark = {#1} , nameref= {#1} }

```

---

<sup>2</sup>I think this is a common requirement and perhaps `\UseInstance` should really o-expand all arguments it receives.

```

111 , bookmark .tl_set:N = \l__head_bookmark_tl
112 , running .tl_set:N = \l__head_running_tl
113 , toc .tl_set:N = \l__head_toc_tl
114 , nameref .tl_set:N = \l__head_nameref_tl
115 %
116 , numbered .bool_set_inverse:N = \l__head_unnumbered_bool
117 , numbered .default:n = true
118 , unnumbered .bool_set:N = \l__head_unnumbered_bool
119 , unnumbered .default:n = true
120 %
121 , subtitle .tl_set:N = \l__head_subtitle_tl
122 , quote .tl_set:N = \l__head_quote_tl

```

We collect labels together with their `\label` command in case there is more than one. This way we can later simply execute `\l__head_label_tl` without any status checks.

```

123 , label .code:n = \tl_put_right:Nn \l__head_label_tl { \label{#1} }
124 }

```

*(End of definition for `\ParseLaTeXeHeading`. This function is documented on page ??.)*

`\__head_find_label:w` A simple-minded check for an existing `\label` command in the main title (could be done better I guess). We add `\label\q_no_value` at the end of the argument, so that we can be sure to find something.

As currently implemented the spaces on both sides of a `\label` command survive if it is removed. This may be an issue in which case this may need some adjustments.

```

125 \cs_new:Npn \__head_find_label:w #1 \label #2#3 \__head_find_label:w {

```

If the token after `\label` is `\q_no_value` then the title had no label and we set the two variables accordingly.

```

126   \quark_if_no_value:nTF {#2}
127   {
128     \__head_debug_typeout:n{---no~label }
129     \tl_clear:N \l__head_label_tl
130     \tl_set:Nn\l__head_title_tl {#1#3}
131   }

```

Otherwise, there was a real `\label` command and #2 holds the label string.

```

132   {
133     \__head_debug_typeout:n{---label~found~#2}
134     \tl_set:Nn\l__head_label_tl { \label{#2} }

```

To construct the value for `\l__head_title_tl` we have to get rid of the two tokens we added at its end, this is done with `\__head_find_label_aux:w`.

```

135     \tl_set:Nn\l__head_title_tl {#1}
136     \__head_find_label_aux:w #3\__head_find_label_aux:w
137   }
138   \__head_debug_typeout:n{---return:~ '\exp_not:o \l__head_title_tl' }
139 }

```

There is at least one more `\label` now and the token after it should be our `\q_no_value`. If not then the title argument had at least 2 labels and we collect the newly found one and recurse.

```

140 \cs_new:Npn \__head_find_label_aux:w #1 \label #2#3 \__head_find_label_aux:w {

```

The #1 may not be everything we have to pick up but we know for sure that it belongs to the title.

```
141 \tl_put_right:Nn \l__head_title_tl { #1 }
```

Now let's see if we are at the end of the argument. If not pick up the newly found \label and recurse on the remaining material.

```
142 \quark_if_no_value:nF {#2}
143 {
144   \__head_debug_typeout:n{---~ extra~ label~ '#2'~ found }
145   \tl_put_right:Nn \l__head_label_tl { \label{#2} }
146   \__head_find_label_aux:w #3\__head_find_label_aux:w
147 }
148 }
```

(End of definition for \\_\_head\_find\_label:w.)

## 11.6 Templates

### 11.6.1 Template types

**heading** (*type*) Templates of type **heading** are used to produce headings. The positional arguments are:

- 1: key/val list
- 2: unnumbered?
- 3: title
- 4: toc
- 5: running
- 6: bookmark
- 7: nameref
- 8: label(s)
- 9: { subtitle } { quote }

```
149 \NewTemplateType{heading}{9}
```

**headformat** (*type*) Templates of type **headformat** are used to produce heading layout from the formatted heading number (if any), the heading title, subtitle and quotation. The positional arguments are:

- 1: key/val list for document-level customizations
- 2: prefix string
- 3: formatted heading number
- 4: title
- 5: subtitle
- 6: quote

```
150 \NewTemplateType{headformat}{6}
```

### 11.6.2 heading templates interfaces

We have two heading templates: `display` and `runin`.

**heading display** (*templ.*) The **display** template produces a display heading, i.e., one that has vertical space before and after it.

```

151 \DeclareTemplateInterface{heading}{display}{9}
152 {
153   , name           : tokenlist
154   , parent-name    : tokenlist
155   , reset-counter  : tokenlist
156   , level          : integer = 0

```

The **placement** key sets default values for the keys **start-code** and **final-code**.

```

157   , placement      : choice {page , top , normal } = normal
158   , mark-cmd       : function(1) =
159   %
160   % many more keys for layout settings are missing for now
161   , para-indent    : choice { true , false } = false

```

We use `\c_max_int` as a fake penalty to indicate that no penalty was given in a key.

```

162   , before-vspace  : skip = Opt
163   , penalty        : integer = \c_max_int
164   , after-penalty-vspace : skip = Opt
165   , after-vspace   : skip = Opt

```

The next two keys are only there to overwrite the **placement** settings with explicit code. Thus, their default value is set up by the **placement** key (which has a default).

```

166   , start-code     : tokenlist =      % no default values!
167   , final-code     : tokenlist =      % no default values!

```

A prefix string such as `\@chappap` could be specified in the next variable. By default is has no value.

```

168   , prefix         : tokenlist = \NoValue
169   , heading-decls  : tokenlist = \normalfont
170   , prefix-decls   : tokenlist =
171   , number-decls   : tokenlist =
172   , title-decls    : tokenlist =
173   , subtitle-decls : tokenlist =
174   , quote-decls    : tokenlist =

175   , headformat-instance : tokenlist = std
176   , number-format      : function(1) = \theheading
177   , contents-extra     : tokenlist =
178 }

```

**heading runin** (*templ.*) This template is similar to the **display** template but implements a **runin** heading, i.e., one where the paragraph text continues on the same line.

Unfortunately, we can't really use `\DeclareTemplateCopy` to set it up because with `\EditTemplateDefaults` we are not able to alter the setup for the **placement** and **para-indent** keys as that involves changing the implementation code. Thus with `\DeclareTemplateCopy` we can copy the interface setup but the code setup still needs to be done using `\DeclareTemplateCode`.

```

179 \DeclareTemplateCopy{heading}{runin}{display}

```



### 11.6.3 headformat templates interfaces

We have three template: display, hang and runin.

headformat display (*templ.*) The display template produces

```
180 \DeclareTemplateInterface{headformat}{display}{6}
181 {
182   , heading-indent      : length = 0pt
183   , title-format        : function(1) = #1
184   , prefix-number-sep   : tokenlist = \WordSpaceAmount{1}
185   , number-title-sep    : tokenlist = 20pt
186 }
```

headformat hang (*templ.*) The hang template produces

```
187 \DeclareTemplateInterface{headformat}{hang}{6}
188 {
189   , heading-indent      : length = 0pt
190   , title-format        : function(1) = #1
191   , prefix-number-sep   : tokenlist = \WordSpaceAmount{1}
192   , number-title-sep    : tokenlist = 1em
193 }
```

headformat runin (*templ.*) The runin template produces

```
194 \DeclareTemplateInterface{headformat}{runin}{6}
195 {
196   , heading-indent      : length = 0pt
197   , title-format        : function(1) = #1
198   , prefix-number-sep   : tokenlist = \WordSpaceAmount{1}
199   , number-title-sep    : tokenlist = 1em
200 }
```

### 11.6.4 heading templates code

heading display (*templ.*)

```
201 \DeclareTemplateCode{heading}{display}{9}
202 {
203   , name                = \l__head_name_tl
204   , level                = \l__head_level_int
205   % next two not yet used
206   , parent-name         = \l__head_pname_tl
207   , reset-counter       = \l__head_reset_cnt_tl
```

The `placement` key determines whether the heading can appear anywhere (`normal`), automatically starts a new page or column (`top`), or is on a page of its own (`page`). It is implemented by setting `\l__head_start_code_tl` and `\l__head_final_code_tl`. These settings can be fine-tuned or overwritten with the keys `start-code` and `final-code`, if necessary.

```
208   , placement          = {
209     ,page               = \__head_debug_typeout:n{ A~ page~ heading }
210                         \tl_set:Nn \l__head_placement_tl { page }
211     ,top                = \__head_debug_typeout:n{ A~ top~ heading }
212                         \tl_set:Nn \l__head_placement_tl { top }
213     ,normal              = \__head_debug_typeout:n{ A~ normal~ heading }
```

```

214         \tl_set:Nn \l__head_placement_tl { normal }
215     }
216     , mark-cmd      = \__head_mark_cmd:n

```

For now we make use of the legacy coding for indentation of the following paragraph.

```

217     , para-indent   = {
218         ,true  = \@afterindenttrue
219         ,false = \@afterindentfalse
220     }
221     , before-vspace = \l__head_before_skip
222     , penalty       = \l__head_penalty_int
223     , after-penalty-vspace = \l__head_after_penalty_skip
224     , after-vspace  = \l__head_after_skip
225     , start-code    = \l__head_start_code_tl
226     , final-code    = \l__head_final_code_tl
227     , prefix        = \l__head_typeset_prefix_tl
228     , headformat-instance = \l__head_headformat_instance_tl
229     , heading-decls  = \l__head_heading_decls_tl
230     , prefix-decls   = \l__head_prefix_decls_tl
231     , number-decls   = \l__head_number_decls_tl
232     , title-decls    = \l__head_title_decls_tl
233     , subtitle-decls = \l__head_subtitle_decls_tl
234     , quote-decls    = \l__head_quote_decls_tl
235     , number-format  = \__head_number_format:n
236     , contents-extra = \l__head_contents_extra_tl
237 }
238 {
239     \tl_set_eq:Nc \theheading { the \l__head_name_tl }

```

We store the value of `secnumdepth` so that we can change it locally.

```

240     \tl_set:Nc \l__head_saved_secnumdepth_tl {\int_use:N \c@secnumdepth}
241     \__head_show_arguments:nnnnnnnnn
242     {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}

```

First evaluate any key setting done by the user in the optional first argument.

```

243     \tl_if_empty:oF {#1} { \SetTemplateKeys{heading}{display}{#1} }

```

Based on the placement key we set up `\l__head_start_code_tl` and `\l__head_final_code_tl`. These two variables can also be set with the keys `start-code` and `final-code` in which case we don't alter the definition.

```

244     \tl_if_empty:oT \l__head_start_code_tl
245     { \str_case:VnF \l__head_placement_tl
246         {
247             { page }
248             { \tl_set:Nn \l__head_start_code_tl

```

Straight from the placement definition of `\part`.

```

249         {
250             \if@openright

```

It would be nice if there is a variant of `\SetTemplateKey` in which the template type and name are implicit, e.g., `\SetCurrentTemplateKeys` because this is usually what I need.

Clearly not `\@tempwa` and the page styles should be adjustable.

```

251         \cleardoublepage
252     \else
253         \clearpage
254     \fi
255     \thispagestyle{plain}%
256     \if@twocolumn
257         \onecolumn
258         \@tempswatrue
259     \else
260         \@tempswafalse
261     \fi
262     \null\vfil
263 }
264 }
265 { top }
266 { \tl_set:Nn \l__head_start_code_tl
267 {
268     \if@openright\cleardoublepage\else\clearpage\fi
269     \thispagestyle{plain}%
270     \global\@topnum\z@
271 }
272 }
273 }
274 { \tl_clear:N \l__head_start_code_tl }
275 }
276 %
277 \tl_if_empty:oT \l__head_final_code_tl
278 { \str_case:VnF \l__head_placement_tl
279 {
280     { page }
281     { \tl_set:Nn \l__head_final_code_tl
282     {
283         \vfil\newpage
284         \if@twoside
285             \if@openright
286                 \null
287                 \thispagestyle{empty}%
288             \newpage
289         \fi
290     \fi
291     \if@tempswa
292         \twocolumn
293     \fi
294 }
295 }
296 }
297 { \tl_set:Nn \l__head_final_code_tl { \@afterheading } }
298 }

```

Then we set up the penalty to use (might be given as a key value).

```

299 \__head_determine_penalty:
300 \__head_determine_number_typesetting:N #2

```

Next comes the vertical spacing and penalty before the heading. This includes running

`\l__head_start_code_tl` if it contains any code, e.g., to start a new page.

```
301 \__head_vertical_before_spacing:
```

We are in vmode now and here is the point where we (with tagging) can close a previous Sect structure and open the new one.

```
302 \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
303 \UseTaggingSocket{sec/begin}
304   {\int_use:N\l__head_level_int}{tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}}
```

Up to this point everything is identical for both display and runin headings. But from now on they have their own code. We have dealt with argument #1 and #2 so now we can unbundle argument #9 so that it is easier to process downstream

```
305 \__head_debug_typeout:n{use~ 'headformat'~instance:~
306                      \l__head_headformat_instance_tl }
307 \use:e {
308   \UseInstance{headformat} { \l__head_headformat_instance_tl }
309   { \exp_not:o \UnusedTemplateKeys }
310   { \exp_not:o { \l__head_typeset_prefix_tl } }
311   { \exp_not:o { \l__head_typeset_number_tl } }
312   { \exp_not:n { #3 } }
313   { \exp_not:o { \use_i:nn #9 } }
314   { \exp_not:o { \use_ii:nn #9 } }
315 }
316 %
317 % --- handle marks, toc-entry, bookmark, nameref, and label
318 %
319 \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}
320 %
321 % --- post-heading handling (vertical)
```

Restore secnumdepth

```
322 \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
323 \par \nobreak
324 \skip_vertical:N \l__head_after_skip
325 % --- prepare next paragraph (defaults to \cs{@afterheading}
326 %
327 \l__head_final_code_tl
328 %
329 \ignorespaces
330 }
```

```
331 \AddToHook{begindocument}{
332   \ifcsname if@openright\endcsname
333   \else
```

Need to hide this a little if it is in fact already defined!

```
334   \expandafter\newif\csname if@openright\endcsname
335   \fi
336 }
```

*(End of definition for .)*

heading runin (*templ.*) The runin template is very similar to the display one.

```

337 \DeclareTemplateCode{heading}{runin}{9}
338 {
339   , name           = \l__head_name_tl
340   , level          = \l__head_level_int
341   % next two not yet used
342   , parent-name    = \l__head_pname_tl
343   , reset-counter  = \l__head_reset_cnt_tl

```

It wouldn't make sense to have page placement with a runin heading since there would be nothing to run into.

```

344   , placement      = {
345     page = \typeout{ ^^JA~ runin~ page~ placement~ heading~makes~no~sense
346              (top-used)}
347     \tl_set:Nn \l__head_placement_tl { top }
348     ,top  = \__head_debug_typeout:n{ A~ top~ heading }
349     \tl_set:Nn \l__head_placement_tl { top }
350     ,normal = \__head_debug_typeout:n{ A~ normal~ heading }
351     \tl_set:Nn \l__head_placement_tl { normal }
352   }
353   , mark-cmd       = \__head_mark_cmd:n

```

In a runin heading you can't set up paragraph indentation of the following paragraph (since that one is "run in". But we accept the key and just spit out a warning.

```

354   , para-indent    = {
355     ,true  = \typeout{para-indent~ setting~ ignored}
356     ,false = \typeout{para-indent~ setting~ ignored}
357   }
358   , before-vspace  = \l__head_before_skip
359   , penalty        = \l__head_penalty_int

```

UFi: this types out messages for all run-in headers! Therefore disabled for now

Next key makes no sense either in a runin heading and is not used.

```

360   , after-penalty-vspace = \l__head_after_penalty_skip
361   , after-vspace        = \l__head_after_skip
362   , start-code          = \l__head_start_code_tl
363   , final-code          = \l__head_final_code_tl
364   , prefix              = \l__head_typeset_prefix_tl
365   , headformat-instance = \l__head_headformat_instance_tl
366   , heading-decls       = \l__head_heading_decls_tl
367   , prefix-decls        = \l__head_prefix_decls_tl
368   , number-decls        = \l__head_number_decls_tl
369   , title-decls         = \l__head_title_decls_tl
370   , subtitle-decls      = \l__head_subtitle_decls_tl
371   , quote-decls         = \l__head_quote_decls_tl
372   , number-format       = \__head_number_format:n
373   , contents-extra      = \l__head_contents_extra_tl
374 }
375 {
376   \__head_show_arguments:nnnnnnnnn
377   {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}

```

Revise and separate the two templates better

store the value of `secnumdepth` so that we can change it locally.

```

378 \tl_set:Nc\l__head_saved_secnumdepth_tl{\int_use:N\c@secnumdepth}

define \theheading
379 \tl_set_eq:Nc \theheading { the \l__head_name_tl }
380 \tl_if_empty:oF {#1} { \SetTemplateKeys{heading}{runin}{#1} }

381 \tl_if_empty:oT \l__head_start_code_tl
382 {
383   \str_case:Vn \l__head_placement_tl
384   {
385     { top } { \tl_set:Nn \l__head_start_code_tl {\clearpage} } }
386   }

```

All other cases want an empty `\l__head_start_code_tl` so nothing to do.

```

387 % { \tl_clear:N \l__head_start_code_tl }
388 }
389 %

```

Nothing at all to do (for now) for `\l__head_final_code_tl`, it should by default be empty in all heading placement. But maybe we end up supporting further placements, so ...

```

390 % \tl_if_empty:oT \l__head_final_code_tl
391 % { \str_case:VnF \l__head_placement_tl
392 % {
393 % { top } { \tl_clear:N \l__head_final_code_tl }
394 % }
395 % { \tl_clear:N \l__head_final_code_tl }
396 % }

397 \__head_determine_penalty:
398 \__head_determine_number_typesetting:N #2
399 \__head_vertical_before_spacing:

```

We are in vmode now and here is the point where we (with tagging) can close a previous Sect structure and open the new one. We also have to initialize the change in the paratagging here as run-in titles typeset the heading in `\everypar`.

```

400 \UseTaggingSocket{sec/end}{\int_use:N\l__head_level_int}
401 \UseTaggingSocket{sec/begin}
402   {\int_use:N\l__head_level_int}{tag=\UseStructureName{sec/\int_use:N\l__head_level_int}}
403 \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}
404 \def \@svsechd {
405   \__head_debug_typeout:n{use~ 'headformat'~instance:~
406     \l__head_headformat_instance_tl }
407   \use:e {
408     \UseInstance{headformat} { \l__head_headformat_instance_tl }
409     { \exp_not:o \UnusedTemplateKeys }
410     { \exp_not:o { \l__head_typeset_prefix_tl } }
411     { \exp_not:o { \l__head_typeset_number_tl } }
412     { \exp_not:n { #3 } }
413     { \exp_not:o { \use_i:nn #9 } }
414     { \exp_not:o { \use_ii:nn #9 } }
415   }
416   \__head_handle_marks_etc:nnnnn {#4}{#5}{#6}{#7}{#8}

```

Restore secnumdepth

```

417     \int_gset:Nn\c@secnumdepth{\l__head_saved_secnumdepth_tl}
418   }
419   %
420   \@nobreakfalse
421   \global\@noskipsectrue
422   \everypar{%
423     \if@noskipsec
424       \global\@noskipsecfalse
425       {\setbox\z@\lastbox}
426       \clubpenalty\@M
427   %Fmi group is in headformat
428   %     \begingroup
429     \@svsechd
430   %     \endgroup
431     \unskip

```

This tagging socket starts the “paragraph” after the run-in heading

```

432     \UseTaggingSocket{sec/title/split}
433     \skip_horizontal:N \l__head_after_skip
434   \else
435     \clubpenalty \@clubpenalty
436     \everypar{}%
437   \fi
438 }

```

— prepare next paragraph (does nothing by default)

```

439 \l__head_final_code_tl
440 %
441 \ignorespaces
442 }

```

### 11.6.5 headformat templates code

`headformat display (templ.)` This template creates a headformat where the number is on a line on its own.

```

443 \DeclareTemplateCode{headformat}{display}{6} % args: keys,prefix,number,title,subtitle,quotat
444 {
445   , heading-indent      = \l__head_heading_indent_dim
446   , title-format        = \__head_title_format:n
447   , prefix-number-sep   = \l__head_prefix_number_sep_tl
448   , number-title-sep    = \l__head_number_title_sep_tl
449 }
450 {
451   \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}

```

First evaluate any key setting done by the user (normally supplied from the main heading instance.

```

452   \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{display}{#1} }

453   \group_begin:
454   \UseTaggingSocket{sec/title/begin}{\int_use:N\l__head_level_int}{#4}}

```

TODO: revisit if `\normalcolor` is needed, if yes, use as `\SaveLastSkip\normalcolor\RestoreLastSkip`

```
455 \normalfont
456 \interlinepenalty \@M
457 \l__head_heading_decls_tl{}
```

If there is a number and so a prefix, the link target should be before the number. We use for now the same place in the unnumbered case. TODO: If we put the target here we do not know the height of the line and the target is perhaps not high enough. And for unnumbered chapter it is perhaps too high. Check!

```
458 \bool_if:NTF \l__head_unnumbered_bool
459 {
460   \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
461   {
462     \skip_horizontal:N \l__head_heading_indent_dim
463     \MakeLinkTarget[\l__head_name_tl]{}
464   }
465   {
466     \MakeLinkTarget[\l__head_name_tl]{}\skip_horizontal:N \l__head_heading_indent_dim
467   }
468 }
469 {
470   \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
471   {
472     \skip_horizontal:N \l__head_heading_indent_dim
473     \MakeLinkTarget{\l__head_name_tl}
474   }
475   {
476     \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_heading_indent_dim
477   }
478   \IfNoValueF{#2}
479   {
480     { \l__head_prefix_decls_tl #2 }
481     \nobreak
482     \skip_horizontal:n { \l__head_prefix_number_sep_tl }
483   }
484   \l__head_number_decls_tl
485   #3
486   \par\nobreak
487   \skip_vertical:n { \l__head_number_title_sep_tl }
488 }
489 \l__head_title_decls_tl
490 \__head_title_format:n {#4}
491 \par
492 \UseTaggingSocket{sec/title/end}
493 \group_end:
494 }
```

probably better to use a `format:n` and drop these two code variables even though they are used by `titlesec`

`headformat hang (templ.)` This template creates a heading with a hanging number.

```
495 \DeclareTemplateCode{headformat}{hang}{6} % args: keys,prefix,number,title,subtitle,quotation
496 {
497   , heading-indent      = \l__head_heading_indent_dim
498   , title-format       = \__head_title_format:n
```



```

499 , prefix-number-sep = \l__head_prefix_number_sep_tl
500 , number-title-sep = \l__head_number_title_sep_tl
501 }
502 {
503   \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}

```

First evaluate any key setting done by the user (normally supplied from the main heading instance).

```

504   \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{hang}{#1} }

505   \group_begin:
506     \UseTaggingSocket{sec/title/init}{\int_use:N\l__head_level_int}

507     \normalfont
508     \interlinepenalty \@M
509     \l__head_heading_decls_tl{

```

The link target should be at the left text margin, or, if the section is moved into the margin, at the left of the number.

```

510     \bool_if:NTF \l__head_unnumbered_bool
511     {
512       \tl_set:Nn\l__head_tmpa_tl
513       {
514         \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
515         {
516           \skip_horizontal:N \l__head_heading_indent_dim \MakeLinkTarget[\l__head_name
517         }
518         {
519           \MakeLinkTarget[\l__head_name_tl]{
520             \skip_horizontal:N \l__head_heading_indent_dim
521           }
522         }
523       }
524     {
525       \tl_set:Nn \l__head_tmpa_tl
526       {
527         \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
528         {
529           \skip_horizontal:N \l__head_heading_indent_dim \MakeLinkTarget{\l__head_name
530         }
531         {
532           \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_heading_indent_c
533         }
534         \IfNoValueF{#2}
535         {
536           { \l__head_prefix_decls_tl #2 }
537           \nobreak
538           \skip_horizontal:n { \l__head_prefix_number_sep_tl }
539         }
540         { \l__head_number_decls_tl #3 }
541         \skip_horizontal:n { \l__head_number_title_sep_tl }
542       }
543     }
544     \UseTaggingSocket{sec/title/hang}

```

```

545     {{\int_use:N\l__head_level_int}\l__head_unnumbered_bool{\l__head_tmpa_tl}{#4}}
546     {
547         \@hangfrom
548         {
549             \l__head_tmpa_tl
550         }
551     }
552     \l__head_title_decls_tl
553     \__head_title_format:n {#4}
554     \par
555     \group_end:
556 }

```

probably better to use a `format:n` and drop these two code variables even though they are used by `titlesec`

`headformat runin` (*templ.*) This template creates a heading with a run-in title. Arguments are key-value, number, title, subtitle, quotation.

```

557 \DeclareTemplateCode{headformat}{runin}{6} % args: keys,prefix,number,title,subtitle,quotation
558 {
559     , heading-indent      = \l__head_heading_indent_dim
560     , title-format       = \__head_title_format:n
561     , prefix-number-sep  = \l__head_prefix_number_sep_tl
562     , number-title-sep   = \l__head_number_title_sep_tl
563 }
564 {
565     \__head_show_arguments:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6}

```

First evaluate any key setting done by the user (normally supplied from the main heading instance).

```

566     \tl_if_empty:oF {#1} { \SetTemplateKeys{headformat}{hang}{#1} }
567     \group_begin:

```

```

568     \normalfont
569     \interlinepenalty \@M
570     \l__head_heading_decls_tl{

```

Setting `\interlinepenalty` makes little sense I think (but that's the way it was in  $\text{\LaTeX 2}_{\epsilon}$ )

We must avoid that the sep between number and title is used in the unnumbered case. So we test with the boolean.

```

571     \bool_if:NTF \l__head_unnumbered_bool
572     {
573         \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
574         {
575             \skip_horizontal:N \l__head_heading_indent_dim
576             \MakeLinkTarget[\l__head_name_tl]{}
577         }
578         {
579             \MakeLinkTarget[\l__head_name_tl]{}\skip_horizontal:N \l__head_heading_indent_dim
580         }
581     }
582     {
583         \dim_compare:nNnTF \l__head_heading_indent_dim < \c_zero_skip
584         {

```

```

585         \skip_horizontal:N \l__head_heading_indent_dim \MakeLinkTarget{\l__head_name_tl}
586     }
587     {
588         \MakeLinkTarget{\l__head_name_tl}\skip_horizontal:N \l__head_heading_indent_dim
589     }
590     {
591         \UseTaggingSocket{sec/title/number}{\int_use:N\l__head_level_int}
592         {
593             \IfNoValueF{#2}
594             {
595                 { \l__head_prefix_decls_tl #2 }
596                 \nobreak
597                 \skip_horizontal:n { \l__head_prefix_number_sep_tl }
598             }
599             \l__head_number_decls_tl
600             #3
601         }
602     }
603     \skip_horizontal:n { \l__head_number_title_sep_tl }
604 }
605 \l__head_title_decls_tl
606 \l__head_title_format:n {#4}
607 \group_end:
608 }

```

### 11.6.6 Internal commands used by the template code

`\__head_determine_penalty:`

```

609 \cs_new:Npn \__head_determine_penalty: {
    If a penalty was specified use it, otherwise use \@secpenalty.
610     \int_compare:nNt \l__head_penalty_int = \c_max_int
611     { \int_set:Nn \l__head_penalty_int \@secpenalty }
612 }

```

*(End of definition for \\_\_head\_determine\_penalty:.)*

`\__head_determine_number_typesetting:N`

```

613 \cs_new:Npn \__head_determine_number_typesetting:N #1 {

```

Using or suppressing a heading number depends on the heading level compared to the document value of `\c@secnumdepth`. If that doesn't suppress the number then an explicit key or a star form might still have suppressed it.

```

614     \bool_set:Nn \l__head_unnumbered_bool
615     { \bool_lazy_or_p:nn
616       { \int_compare_p:nNn \l__head_level_int > \c@secnumdepth }
617       { \bool_if_p:N #1 }
618     }

```

If we aren't producing a heading with a number we set `\c@secnumdepth` to a low number (it is reset at the end of the template code)

```

619     \bool_if:NtF \l__head_unnumbered_bool
620     {
621         \int_gset:Nn\c@secnumdepth{-99}

```

and we set `\l__head_typeset_number_tl` to do nothing.

```
622 \tl_clear:N \l__head_typeset_number_tl
623 }
```

Otherwise the heading counter is incremented and a formatted version of the number plus any following (or preceding) space is stored in `\l__head_typeset_number_tl`. We use the kernel version of `\refstepcounter` as anchors are handled elsewhere.

```
624 {
625   \@kernel@refstepcounter{ \l__head_name_tl }
626   \protected@edef \l__head_typeset_number_tl
627     {
628       \__head_number_format:n { \l__head_name_tl }
629     }
630 }
631 }
```

*(End of definition for `\__head_determine_number_typesetting:N`.)*

`\__head_vertical_before_spacing:`

```
632 \cs_new:Npn \__head_vertical_before_spacing: {
633   \tl_if_blank:VTF \l__head_start_code_tl
634   {
635     \if@noskipsec \leavevmode \fi
636     \par
637     \if@nobreak
638       \everypar{}
639     \else
640       \addpenalty \l__head_penalty_int
641       \addvspace \l__head_before_skip
```

The `\vspace*` inserts a rule, we insert therefore the skip only if it is different to zero.

```
642     \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
643     { \vspace* \l__head_after_penalty_skip }
644   \fi
645 }
646 {
647   \l__head_start_code_tl
```

If `\l__head_start_code_tl` holds code, we assume that it handles pagination, e.g., a `\clearpage`, etc. We therefore only add the skip that would follow the penalty.

```
648   \dim_compare:nNnF{\l__head_after_penalty_skip}={0pt}
649   { \vspace* \l__head_after_penalty_skip }
650 }
651 }
```

*(End of definition for `\__head_vertical_before_spacing:`.)*

`\addcontentslinebookmark`  
`\addcontentslinebookmarkOff`  
`\addcontentslinebookmarkOn`

We want to be able to control bookmarks independently from the toc entries, and also want to disable a bookmark by setting it to empty. For this we need some command to handle the bookmark command.

```
652 \newcommand\addcontentslinebookmark[3]{}
653 \newcommand\addcontentslinebookmarkOff{}
654 \newcommand\addcontentslinebookmarkReset{}
655 \providecommand\texorpdfstring[2]{#1}
```

This can go once hyperref is updated (ufi,2026-04-21):

```

656 \AddToHook{package/hyperref/after}
657 {
658   \RenewCommandCopy\addcontentslinebookmark\Hy@addcontentsline@addbookmark
659   \renewcommand\addcontentslinebookmarkOff
660   {
661     \ifHy@bookmarks
662       \let\addcontentslinebookmarkReset\Hy@bookmarkstrue
663     \else
664       \let\addcontentslinebookmarkReset\relax
665     \fi
666     \Hy@bookmarksfalse
667   }
668 }

```

*(End of definition for \addcontentslinebookmark, \addcontentslinebookmarkOff, and \addcontentslinebookmarkOn. These functions are documented on page ??.)*

\\_head\\_handle\\_marks\\_etc:nnnnn Arg 1: toc entry, arg 2: mark, arg 3: bookmark, arg 4: nameref, arg 5: label code

```

669 \cs_new:Npn \_head\_handle\_marks\_etc:nnnnn #1#2#3#4#5 {
670 %
671   \tl_set:Nn \@currentlabelname{#4}
672   \IfBlankF {#2}
673     { \_head\_mark\_cmd:n { #2 } }
674   \IfBlankTF {#1}

```

If the toc entry is empty the bookmarks must be set without \addcontentsline

```

675   {
676     \IfBlankF{#3}
677     {
678       \addcontentslinebookmark{toc}{\l__head\_name\_tl}
679       {
680         \bool_if:NF \l__head\_unnumbered\_bool
681         {
682           \protect\numberline{ \use:c{ the \l__head\_name\_tl } }
683         }
684         #3
685       }
686     }
687   }
688   {

```

If the bookmark entry is empty we must disable the bookmarks locally before the \addcontentsline and reenale afterwards. We do not check if they are already disabled, perhaps later.

```

689   \IfBlankT{#3}{\addcontentslinebookmarkOff}
690   \addcontentsline{toc}{ \l__head\_name\_tl }
691   {
692     \bool_if:NF \l__head\_unnumbered\_bool
693     {
694       \protect\numberline{ \use:c{ the \l__head\_name\_tl } }
695     }

```

We use `\texorpdfstring` to separate toc and bookmark text.

```
696     \texorpdfstring{#1}{#3}
697   }
698   \addcontentsline{bookmark}{Reset
699 }
```

Some headings (like `\chapter`) also want to write stuff to other contents files like `.lot` or `.lof`.

perhaps this should have a hook for packages

```
700 \l__head_contents_extra_tl
```

We can always run the label code (it might be empty)

```
701 \__head_debug_typeout:n{--->~label(s):~ \exp_not:n{#5}}
702 #5
703 }
```

*(End of definition for \\_\_head\_handle\_marks\_etc:nnnnn.)*

## 11.7 Support for legacy classes and packages

If we define heading commands in the kernel (or in the tagging code) using

```
\DeclareDocumentCommand \section {s = {shorttitle} o m}
{ \ParseLaTeXeHeading {section} {#1} {#2} {#3} }
```

then this gets overwritten by every class right now.

If we redeclare them after the class was loaded then we overwrite the layout of legacy classes (done, for example, with `\@startsection`). New classes that use the above interface would be fine though, as long as we have declared the instances before the class is loaded.

So to make legacy classes work with their existing layout, we should not (ever) overwrite `\section` but let the class definition call `\@startsection` which would then set up suitable instances and only after that call `\ParseLaTeXeHeading`.

However, that would mean a “new” class like `ltx-article` would need to add the above definition and declare or edit the corresponding instances, instead of just declaring or adding the instances.

A perhaps better alternative could be to delay the definition of `\section` and friends until after the class got loaded and then take a peak at `\section` as defined by the class and if it contains a `\@startsection` call, leave it alone and otherwise overwrite it. And if the class hasn’t defined `\section` (because it is a new class) declare it. Of course that means `\section` would then be available with every class even if it was never meant to contain headings.

But while writing this up, I start to think it is best if a new class defines both the document interface (i.e., the above command) as well as the layout (declare the instances) and the kernel does neither. It has been this way before and it is consistent, so why change.

The situation with headings defined via `\secdef` is worse: we don’t have a nice handle as with `\@startsection` so there is no real way other than through static analysis to set up such a heading in the new template style. So all that is possible (I think) is that after the class has been loaded, we look if the usual candidates (`\part` and `\chapter`) have been defined and overwrite them with a standard layout. That would make the class tagging aware but, of course, would likely change the layout.

The current implementation

- provides instance for the heading commands of the standard classes;
- declares the heading commands for the standard classes with the new interfaces through class hooks;
- other classes call the adapted `\@startsection`; other headings command like `\part` or `\chapter` are not handled and must be setup individually.

### 11.7.1 Instances (sample/default definitions)

`\WordSpaceAmount`

Produce the amount of a (fractional) word space in the current font in a way that it can be used in a skip or dimen register assignment. The argument specifies the fraction, e.g., 2 gives two wordspaces and .5 would produce half a word space.

This general helper doesn't really belong here, so should be eventually moved.

```

704 \newcommand\WordSpaceAmount[1]{
705   \glueexpr
706     #1\fontdimen2\the\font
707   plus #1\fontdimen3\the\font
708   minus #1\fontdimen4\the\font
709   \relax
710 }
```

*(End of definition for \WordSpaceAmount. This function is documented on page ??.)*

`heading part` (*inst.*) An instance suitable for book and report. An instance suitable for article is above in the hook.

```

711 \DeclareInstance{heading}{part}{display}
712 {
713   , name           = part
714   , level          = -1
715   , placement      = page
716   , after-penalty-vspace = 0pt
717   , prefix         = \partname
718   , number-format  = \thepart
719   , heading-decls  = \centering\bfseries\huge
720   , title-decls    = \Huge
721   , headformat-instance = part
722   , mark-cmd       = \partmark {#1}
723 }
```

`heading chapter` (*inst.*)

```

724 \DeclareInstance{heading}{chapter}{display}
725 {
726   , name           = chapter
727   , level          = 0
728   , placement      = top
729   , after-penalty-vspace = 50pt
730   , after-vspace    = 40pt
731   , prefix         = \@chapapp
732   , number-format  = \thechapter
733   , heading-decls  = \raggedright \parindent0pt \bfseries \huge
734   , title-decls    = \Huge
```

```

735 , headformat-instance = chapter
736 , mark-cmd           = \chaptermark {#1}
737 , contents-extra     = \addtocontents{lof}{\protect\addvspace{10\p@}}
738                     \addtocontents{lot}{\protect\addvspace{10\p@}}
739 }

```

heading section (*inst.*)

```

740 \DeclareInstance{heading}{section}{display}
741 {
742   , name              = section
743   , level             = 1
744   , mark-cmd          = \sectionmark {#1}
745   , before-vspace     = 3.5ex plus 1ex minus .2ex
746   , after-vspace      = 2.3ex plus .2ex
747   , heading-decls     = \normalfont\Large\bfseries
748   , headformat-instance = section
749 }

```

heading subsection (*inst.*)

```

750 \DeclareInstance{heading}{subsection}{display}
751 {
752   , name              = subsection
753   , parent-name       = section
754   , level             = 2
755   , mark-cmd          = \subsectionmark {#1}
756   , before-vspace     = 3.25ex plus 1ex minus .2ex
757   , after-vspace      = 1.5ex plus .2ex
758   , heading-decls     = \normalfont\large\bfseries
759   , headformat-instance = subsection
760 }

```

heading subsubsection (*inst.*)

```

761 \DeclareInstance{heading}{subsubsection}{display}
762 {
763   , name              = subsubsection
764   , parent-name       = subsection
765   , level             = 3
766   , before-vspace     = 3.25ex plus 1ex minus .2ex
767   , after-vspace      = 1.5ex plus .2ex
768   , heading-decls     = \normalfont\normalsize\bfseries
769   , headformat-instance = subsubsection
770 }

```

heading paragraph (*inst.*)

```

771 \DeclareInstance{heading}{paragraph}{runin}
772 {
773   , name              = paragraph
774   , parent-name       = subsubsection
775   , level             = 4
776   , before-vspace     = 3.25ex \@plus1ex \@minus.2ex
777   , after-vspace      = 1em
778   , heading-decls     = \normalfont\normalsize\bfseries
779   , mark-cmd          = \paragraphmark {#1}

```



```

780     , headformat-instance = paragraph
781
782 }

heading subparagraph (inst.)

783 \DeclareInstance{heading}{subparagraph}{runin}
784 {
785     , name           = subparagraph
786     , parent-name    = paragraph
787     , level          = 5
788     , before-vspace  = 3.25ex \@plus1ex \@minus .2ex
789     , after-vspace   = 1em
790     , heading-decls  = \normalfont\normalsize\bfseries
791     , mark-cmd       = \subparagraphmark {#1}
792     , headformat-instance = subparagraph
793
794 }

headformat part (inst.) The headformat instances for part and chapter use the display template with identical
headformat chapter (inst.) settings by default, so one is made a copy of the other to speed up loading.

795 \DeclareInstance{headformat}{part}{display}
796 {
797     , heading-indent    = 0pt
798     , prefix-number-sep = \WordSpaceAmount{1}
799 }

800 \DeclareInstanceCopy{headformat}{chapter}{part}

headformat std (inst.) Similarly, by default the instances for section, subsection, and subsubsection are all using
headformat section (inst.) the hang template and the same key values as the std instance, so they are all made a
headformat subsection (inst.) copy of that one.
headformat subsubsection (inst.)

801 \DeclareInstance{headformat}{std}{hang}
802 {
803     , heading-indent = 0pt
804 }

805 \DeclareInstanceCopy{headformat}{section}{std}
806 \DeclareInstanceCopy{headformat}{subsection}{std}
807 \DeclareInstanceCopy{headformat}{subsubsection}{std}

headformat paragraph (inst.) In contrast, paragraph and subparagraph differ even though both use the runin template.
headformat subparagraph (inst.)

808 \DeclareInstance{headformat}{paragraph}{runin}
809 {
810     , heading-indent = 0pt
811 }

812 \DeclareInstance{headformat}{subparagraph}{runin}
813 {
814     , heading-indent = \parindent
815 }

adformat section-special (inst.)

```

A special headformat instance for testing. It can be used with `\section[headformat-instance=section-special]{bla}`

```
816 \DeclareInstance{headformat}{section-special}{hang}
817 {
818   , heading-indent = 4em
819   , title-format   = {#1!}
820 }
```

### 11.7.2 New `\@startsection`

`\@startsection` To support legacy classes that implement headings through a call to `\@startsection` we redefine this command to generate a heading instance from the arguments of `\@startsection` if it doesn't yet exist and then use this instance to typeset the heading. NOTE: To handle legacy definition like `{\normalfont\Large\bfseries\MakeUppercase}` in the last argument it copies this argument both to `heading-decls` and to `title-format`.

```
821 \DeclareDocumentCommand \@startsection {mmmmm s ={\shorttitle}o m}{
```

If there already exists an instance `#1-\@startsection` then arguments 2-6 are ignored and we simply call that instance. Otherwise we go through the process to set it up. This allows classes, packages and authors to create and overwrite definitions with `\@startsection`. But after a call to a command using the `\@startsection` the instances are created. It is therefore not possible to redefine the command after a first use or to create two commands using the same level, e.g. `\section` and `\specialsection`. Such setups should use the new interfaces to declare heading commands.

```
822   \IfInstanceExistsF{heading}{#1-\@startsection}{
823     \__head_debug_typeout:n{Info:~ setting~ up~ instances~ for~
824       legacy~ \string\@startsection}
```

In the  $\text{\LaTeX} 2_{\epsilon}$  logic a negative `#4` means we do not indent the following paragraph ... It is okay to change any `em` or `ex` specifications to real `pt` values here, because this code is executed in the same place where `\@startsection` is normally executed and inside the original definitions such assignments happen as well, before there is any font change happening for `#4` and `#5`.

```
825   \@tempskipa #4\relax
826   \@afterindenttrue
827   \ifdim \@tempskipa <\z@
828     \@tempskipa -\@tempskipa
829     \@afterindentfalse
830   \fi
```

... and a negative `#5` means we should produce a runin heading.

```
831   \@tempskipb #5\relax
832   \ifdim \@tempskipb >\z@
833     \use:e {
834       \DeclareInstance{heading}{#1-\@startsection}{display}{
835         , name           = #1
836         , level          = #2
837         , mark-cmd       = \exp_not:c {#1mark} {##1}
838         , before-vspace = \the\@tempskipa
839         , after-vspace  = \the\@tempskipb
```

```

840         , para-indent    = \if@afterindent true \else false\fi
841         , heading-decls = \exp_not:n {#6}

```

we also do the declarations in `title-format` to handle settings which ends, e.g., with `\MakeUppercase`,

```

842         , headformat-instance = #1-@startsection
843     }}

```

We can expect that the instance `#1-@startsection` is not set up by the user if `#1-@startsection` isn't, so no check.

```

844     \DeclareInstance{headformat}{#1-@startsection}{hang}{heading-indent = #3,title-
format={#6{##1}}}}
845     \else
846     \@tempskipb=-\@tempskipb
847     \use:e {
848     \DeclareInstance{heading}{#1-@startsection}{runin}{
849         , name          = #1
850         , level          = #2
851         , mark-cmd       = \exp_not:c {#1mark} {##1}
852         , before-vspace = \the\@tempskipa
853         , after-vspace  = \the\@tempskipb
854         , heading-decls = \exp_not:n{#6}
855         , headformat-instance = #1-@startsection
856     }}
857     \DeclareInstance{headformat}{#1-@startsection}{runin}{heading-indent = #3,title-
format={#6{##1}}}}
858     \fi
859 }
860 \ParseLaTeXeHeading {#1-@startsection}{#7}{#8}{#9}
861 }

```

(End of definition for `\@startsection`. This function is documented on page ??.)

Some classes redefine `\@startsection` and then our redefinition is lost. So we reinstate it

```

862 \NewCommandCopy\kernel@startsection\@startsection
863 \AddToHook{class/after}[head/@startsection]
864   {\RenewCommandCopy\@startsection\kernel@startsection}

```

### 11.7.3 New `\secdef`

The command maps standard `\secdef` definition of `\part` and `\chapter` to the new interface. Unknown heading commands warn (or error if tagging is active) and then call the old commands.

`\secdef`

```

865 \RenewDocumentCommand\secdef{mmsO{#5}m}
866 {
867   \str_case:enF {\cs_to_str:N#1}
868   {
869     {@part}
870     {
871       \IfNoValueTF{#4}
872       {\ParseLaTeXeHeading{part}{#3} {start-code=\relax} {#5}}

```

```

873     {
874       \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
875       \ExpandArgs{nne}\ParseLaTeXeHeading{part}{#3}{start-code=\relax,\exp_not:o{\l__head_
876     }
877     \@latex@warning{\noexpand\secdef-detected-in~\noexpand\part~command.\MessageBreak
878       \noexpand\part~will~be~redefined~to~use~templates.\MessageBreak
879       This~may~change~the~layout!}
880     \DeclareDocumentCommand \part {s ={\shorttitle}o m}
881     { \ParseLaTeXeHeading {part} {##1} {##2} {##3} }
882   }
883   {@chapter}
884   {
885     \IfNoValueTF{#4}
886     { \ParseLaTeXeHeading{chapter}{#3} {#4} {#5} }
887     {
888       \__head_process_shorttitle:nN{#4}\l__head_tmpa_tl
889       \ExpandArgs{nno}\ParseLaTeXeHeading{chapter}{#3}{\l__head_tmpa_tl} {#5}
890     }
891     \@latex@warning{\noexpand\secdef-detected-in~\noexpand\chapter~command.\MessageBreak
892       \noexpand\chapter~will~be~redefined~to~use~templates.\MessageBreak
893       This~may~change~the~layout!}
894     \DeclareDocumentCommand \chapter {s ={\shorttitle}o m}
895     { \ParseLaTeXeHeading {chapter} {##1} {##2} {##3} }
896   }
897 }
898 {
899   \tag_if_active:TF\@latex@error\@latex@warning
900   {\noexpand\secdef~with~unknown~argument~\noexpand#1 found.\MessageBreak
901     The~command~can~not~be~adapted~on~the~fly~to~support~tagging.\MessageBreak
902     The~heading~command~using~this~\noexpand\secdef~should~be~
903       reimplemented\MessageBreak with~templates}{ }
904   \IfBooleanTF{#3}{#2{#5}}{#1[#4]{#5}}
905 }
906 }

```

A helper command to reprocess the argument as key-val

```

907 \NewDocumentCommand\__head_process_shorttitle:nN{={\shorttitle}mm}
908 { \tl_set:Nn#2{#1} }

```

(End of definition for `\secdef`. This function is documented on page ??.)

#### 11.7.4 Core L<sup>A</sup>T<sub>E</sub>X

We define here as an example the heading commands with the new interfaces for the three standard classes.

```

909 \AddToHook{class/article/after}[head/example]
910 {
911   \DeclareInstance{heading}{part}{display}
912   {
913     , name          = part
914     , level         = -1
915     , before-vspace = 4ex
916     , after-vspace  = 3ex
917     , mark-cmd      = \partmark {#1}

```

```

918     , prefix          = \partname
919     , number-format = \thepart
920     , heading-decls = \raggedright\bfseries\Large
921     , title-decls    = \huge
922     , headformat-instance = part
923   }
924   \DeclareInstance{headformat}{part}{display}
925   {
926     , heading-indent    = Opt
927     , number-title-sep = Opt
928     , prefix-number-sep = \WordSpaceAmount{1}
929   }
930   \DeclareDocumentCommand \part {s ={\shorttitle}o m}
931   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
932   \__head_setup_default_heading:
933 }

934 \AddToHook{class/report/after}[head/example]
935 {
936   \DeclareDocumentCommand \part {s ={\shorttitle}o m}
937   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
938   \DeclareDocumentCommand \chapter {s ={\shorttitle}o m}
939   {
940     \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
941   }
942   \__head_setup_default_heading:
943 }
944 \AddToHook{class/book/after}[head/example]
945 {
946   \DeclareDocumentCommand \part {s ={\shorttitle}o m}
947   { \ParseLaTeXeHeading {part} {#1} {#2} {#3} }
948   \DeclareDocumentCommand \chapter {s ={\shorttitle}o m}
949   {
950     \if@mainmatter
951       \ParseLaTeXeHeading {chapter}{#1} {#2} {#3}
952     \else
953       \IfBooleanTF{#1}
954       {% starred:
955         \ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}
956       }
957       {\IfNoValueTF{#2}
958        {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {\shorttitle={#3}} {#3}}
959        {\ParseLaTeXeHeading {chapter}{\BooleanTrue} {#2} {#3}}}
960     }
961     \fi
962   }
963   \__head_setup_default_heading:
964 }
965 \cs_new_protected:Npn \__head_setup_default_heading:
966 {
\section
967   \DeclareDocumentCommand \section {s ={\shorttitle}o m}
968   { \ParseLaTeXeHeading {section} {##1} {##2} {##3} }

```

*(End of definition for \section. This function is documented on page ??.)*

`\subsection`

```
969 \DeclareDocumentCommand \subsection {s ={\shorttitle}o m}
970 { \ParseLaTeXeHeading {subsection} {##1} {##2} {##3} }
```

*(End of definition for \subsection. This function is documented on page ??.)*

`\subsubsection`

```
971 \DeclareDocumentCommand \subsubsection {s ={\shorttitle}o m}
972 { \ParseLaTeXeHeading {subsubsection} {##1} {##2} {##3} }
```

*(End of definition for \subsubsection. This function is documented on page ??.)*

`\paragraph`

```
973 \DeclareDocumentCommand \paragraph {s ={\shorttitle}o m}
974 { \ParseLaTeXeHeading {paragraph} {##1} {##2} {##3} }
```

*(End of definition for \paragraph. This function is documented on page ??.)*

`\subparagraph`

```
975 \DeclareDocumentCommand \subparagraph {s ={\shorttitle}o m}
976 { \ParseLaTeXeHeading {subparagraph} {##1} {##2} {##3} }
977 } %end of command
```

*(End of definition for \subparagraph. This function is documented on page ??.)*

```
978 \end{package}
```

## 12 Issues and problems noticed along the way

### 12.1 Local \DeclareInstance

`\DeclareInstance` does its declaration locally. That might be the right decision (or not) but we need to make sure that it in that case all of the declaration is local.

One potential problem with that is that it might allocate `TeX` registers.

The problem showed up in the redefinition of `\@startsection`, because in the current document some headings are local to an environment, so things get redeclared over and over again.

### 12.2 \SetCurrentTemplateKeys

Furthermore, I think I would like to have some `\SetCurrentTemplateKeys` where one does not have to specify the type nor the template name, because that is how it is always used (by me).

## 12.3 O-expansion of \UseInstance arguments

Whenever a template code calls a sub-instance and that sub-instance takes arguments, then the values for these arguments are typically inside tokenlists or registers so that for efficiency (and sometimes as a total must) one has to o-expand all arguments. While that can be coded somehow, e.g.,

```
\use:e {
  \UseInstance{headformat} { \l_@@_headformat_instance_tl }
    { \exp_not:o \UnusedTemplateKeys }
    { \exp_not:o { \l_@@_typeset_number_tl } }
    { \exp_not:n { #3 } }
    { \exp_not:o { \use_i:nn #9 } }
    { \exp_not:o { \use_ii:nn #9 } }
}
```

it would be better to have a version of \UseInstance that does this automatically. No suggestion for a name.

## 12.4 Switch \openright is not defined by core

I think this should change and it might be enough to just define it in the kernel (I think \newif no longer complains if it acts on an existing \if...

## 12.5 Indexheading at least for l3doc is wrong now

Needs checking.

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
\@startsection .....	<i>16</i>
<b>A</b>	
\addcontents .....	<i>12</i>
\addcontentsline .....	<i>17, 37, 690</i>
\addcontentslinebookmark .....	<i>652, 678</i>
\addcontentslinebookmarkOff ..	<i>652, 689</i>
\addcontentslinebookmarkOn .....	<i>652</i>
\addcontentslinebookmarkReset .....	<i>654, 662, 664, 698</i>
\addpenalty .....	<i>640</i>
\addtocontents .....	<i>737, 738</i>
\AddToHook .....	<i>67, 331, 656, 863, 909, 934, 944</i>
\advspace .....	<i>641, 737, 738</i>
<b>B</b>	
\begingroup .....	<i>428</i>
<b>C</b>	
\centering .....	<i>719</i>
\bfseries .....	<i>719, 733, 747, 758, 768, 778, 790, 920</i>
bool commands:	
\bool_gset_false:N .....	<i>18</i>
\bool_gset_true:N .....	<i>13</i>
\bool_if:NTF .....	<i>24,</i>
<i>26, 97, 458, 510, 571, 619, 680, 692</i>	
\bool_if_p:N .....	<i>617</i>
\bool_lazy_or_p:nn .....	<i>615</i>
\bool_new:N .....	<i>8, 59</i>
\bool_set:Nn .....	<i>614</i>
\bool_set_false:N .....	<i>85</i>
\bool_set_true:N .....	<i>79</i>
\BooleanFalse .....	<i>9, 97</i>
\BooleanTrue .....	<i>9, 97, 955, 958, 959</i>

<code>\chapter</code> .....	4, 8, 15, 16, 38, 39, 43, 891, 892, 894, 938, 948
<code>\chaptermark</code> .....	11, 736
<code>\cleardoublepage</code> .....	251, 268
<code>\clearpage</code> .....	10, 36, 253, 268, 385
<code>\clubpenalty</code> .....	426, 435
<code>\cs</code> .....	325
cs commands:	
<code>\cs_gset_protected:Npe</code> .....	23, 25
<code>\cs_new:Npn</code> .....	
..	31, 40, 125, 140, 609, 613, 632, 669
<code>\cs_new_eq:NN</code> .....	9, 10
<code>\cs_new_protected:Npn</code> .....	
.....	11, 16, 21, 28, 29, 68, 965
<code>\cs_to_str:N</code> .....	867
<code>\csname</code> .....	334
<b>D</b>	
<code>\DebugHeadingsOff</code> .....	17, 28
<code>\DebugHeadingsOn</code> .....	17, 28
<code>\DeclareDocumentCommand</code> .....	
.....	821, 880, 894, 930, 936, 938, 946, 948, 967, 969, 971, 973, 975
<code>\DeclareInstance</code> ...	46, 711, 724, 740, 750, 761, 771, 783, 795, 801, 808, 812, 816, 834, 844, 848, 857, 911, 924
<code>\DeclareInstanceCopy</code> .	800, 805, 806, 807
<code>\DeclareTemplateCode</code> .....	
.....	24, 201, 337, 443, 495, 557
<code>\DeclareTemplateCopy</code> .....	24, 179
<code>\DeclareTemplateInterface</code> .....	
.....	151, 180, 187, 194
<code>\def</code> .....	404
dim commands:	
<code>\dim_compare:nNnTF</code> .....	
.....	460, 470, 514, 527, 573, 583, 642, 648
<code>\DocumentMetadata</code> .....	2
<b>E</b>	
<code>\EditTemplateDefaults</code> .....	24
<code>\else</code> .....	252, 259, 268, 333, 434, 639, 663, 840, 845, 952
<code>\endcsname</code> .....	332, 334
<code>\endgroup</code> .....	430
<code>\everypar</code> .....	30, 422, 436, 638
exp commands:	
<code>\exp_not:N</code> .....	95, 837, 851
<code>\exp_not:n</code> .....	33, 34, 35, 36, 37, 38, 42, 43, 44, 45, 46, 47, 48, 49, 50, 71, 72, 96, 98, 99, 100, 101, 102, 103, 104, 105, 138, 309, 310, 311, 312, 313, 314, 409, 410, 411, 412, 413, 414, 701, 841, 854, 875
<code>\expandafter</code> .....	334
<code>\ExpandArgs</code> .....	875, 889
<b>F</b>	
<code>\fbox</code> .....	4
<code>\fi</code> ..	254, 261, 268, 289, 290, 293, 335, 437, 635, 644, 665, 830, 840, 858, 961
<code>\font</code> .....	706, 707, 708
<code>\fontdimen</code> .....	706, 707, 708
<b>G</b>	
<code>\global</code> .....	270, 421, 424
<code>\glueexpr</code> .....	705
group commands:	
<code>\group_begin:</code> .....	453, 505, 567
<code>\group_end:</code> .....	493, 555, 607
<b>H</b>	
head commands:	
<code>\head_debug_off:</code> .....	17, 11, 16, 29
<code>\head_debug_on:</code> .....	17, 11, 11, 28
head internal commands:	
<code>\l__head_after_penalty_skip</code> ....	
.....	223, 360, 642, 643, 648, 649
<code>\l__head_after_skip</code> ..	224, 324, 361, 433
<code>\l__head_before_skip</code> ...	221, 358, 641
<code>\l__head_bookmark_tl</code> .....	
.....	53, 78, 84, 101, 111
<code>\l__head_contents_extra_tl</code> ....	
.....	236, 373, 700
<code>\__head_debug:n</code> .....	9, 9, 23
<code>\g__head_debug_bool</code> ..	8, 13, 18, 24, 26
<code>\__head_debug_gset:</code> ....	11, 14, 19, 21
<code>\__head_debug_typeof:n</code> .....	9, 10, 25, 32, 33, 34, 35, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 69, 70, 93, 128, 133, 138, 144, 209, 211, 213, 305, 348, 350, 405, 701, 823
<code>\__head_determine_number_-</code> <code>typesetting:N</code> ...	300, 398, 613, 613
<code>\__head_determine_penalty:</code> ....	
.....	299, 397, 609, 609
<code>\l__head_final_code_tl</code> .....	
.....	25, 26, 30, 226, 277, 281, 297, 327, 363, 390, 393, 395, 439
<code>\__head_find_label:w</code> ....	73, 125, 125
<code>\__head_find_label_aux:w</code> .....	
.....	22, 136, 140, 146
<code>\__head_handle_marks_etc:nnnnn</code> .	
.....	319, 416, 669, 669
<code>\l__head_headformat_instance_tl</code> .....	228, 306, 308, 365, 406, 408
<code>\l__head_heading_decls_tl</code> .....	
.....	229, 366, 457, 509, 570
<code>\l__head_heading_indent_dim</code> ....	
.....	445, 460, 462, 466, 470, 472,



476, 497, 514, 516, 520, 527, 529, 532, 559, 573, 575, 579, 583, 585, 588	\l__head_instance_keys_tl . . . . .	\l__head_tmpa_tl . . . . . 52, 512, 525, 545, 549, 874, 875, 888, 889
21, 60, 91, 92, 96	..... 21, 60, 91, 92, 96	\l__head_toc_tl . . . 55, 76, 82, 99, 113
\l__head_label_tl . . . . .	..... 36, 60, 311, 411, 622, 626	\l__head_typeset_number_tl . . . . .
20, 22, 60, 103, 123, 129, 134, 145	..... 227, 310, 364, 410	\l__head_typeset_prefix_tl . . . . .
\l__head_level_int . . . . .	..... 204, 302, 304, 340, 400, 402, 403, 454, 506, 545, 591, 616	\l__head_unnumbered_bool . . . . .
216, 353, 673	..... 79, 85, 97, 116, 118, 458, 510, 545, 571, 614, 619, 680, 692	\l__head_vertical_before_spacing:
\l__head_name_tl . . . . .	..... 301, 399, 632, 632	headformat (type) . . . . . 150
203, 239, 339, 379, 463, 466, 473, 476, 516, 519, 529, 532, 576, 579, 585, 588, 625, 628, 678, 682, 690, 694	..... 231, 368, 484, 540, 599	headformat chapter (instance) . . . . . 795
\l__head_nameref_tl . . 56, 87, 102, 114	..... 235, 372, 628	headformat display (template) . . 180, 443
\l__head_number_decls_tl . . . . .	..... 448, 487, 500, 541, 562, 603	headformat hang (template) . . . 187, 495
231, 368, 484, 540, 599	..... 222, 359, 610, 611, 640	headformat paragraph (instance) . . . . 808
\l__head_number_format:n 235, 372, 628	..... 60, 210, 212, 214, 245, 278, 347, 349, 351, 383, 391	headformat part (instance) . . . . . 795
\l__head_number_title_sep_tl . . . . .	..... 206, 342	headformat runin (template) . . . 194, 557
448, 487, 500, 541, 562, 603	..... 230, 367, 480, 536, 595	headformat section (instance) . . . . . 801
\l__head_penalty_int . . . . .	..... 447, 482, 499, 538, 561, 597	headformat section-special (instance) 816
222, 359, 610, 611, 640	..... 874, 888, 907	headformat std (instance) . . . . . 801
\l__head_placement_tl 60, 210, 212, 214, 245, 278, 347, 349, 351, 383, 391	..... 234, 371	headformat subparagraph (instance) . . 808
\l__head_pname_tl . . . . . 206, 342	..... 58, 89, 105, 122	headformat subsection (instance) . . . 801
\l__head_prefix_decls_tl . . . . .	..... 207, 343	headformat subsubsection (instance) . 801
230, 367, 480, 536, 595	..... 54, 77, 83, 100, 112	heading (type) . . . . . 149
\l__head_prefix_number_sep_tl . . . . .	..... 60, 240, 322, 378, 417	heading chapter (instance) . . . . . 724
447, 482, 499, 538, 561, 597	..... 932, 942, 963, 965	heading display (template) . . . 151, 201
\l__head_process_shorttitle:nN . . . . .	..... 31, 31, 451, 503, 565	heading paragraph (instance) . . . . . 771
874, 888, 907	..... 40, 40, 241, 376	heading part (instance) . . . . . 711
\l__head_quote_decls_tl . . . 234, 371	..... 25, 26, 28, 30, 36, 225, 244, 248, 266, 274, 362, 381, 385, 387, 633, 647	heading runin (template) . . . . . 179, 337
\l__head_quote_tl . . . 58, 89, 105, 122	..... 233, 370	heading section (instance) . . . . . 740
\l__head_reset_cnt_tl . . . . . 207, 343	..... 57, 88, 104, 121	heading subparagraph (instance) . . . . 783
\l__head_running_tl 54, 77, 83, 100, 112	..... 232, 369, 489, 552, 605	heading subsection (instance) . . . . . 750
\l__head_saved_secnumdepth_tl . . . . .	..... 446, 490, 498, 553, 560, 606	heading subsubsection (instance) . . . . 761
60, 240, 322, 378, 417	..... 20, 22, 60, 82, 83, 84, 87, 98, 130, 135, 138, 141	\Huge . . . . . 720, 734
\l__head_setup_default_heading: . . . . .		\huge . . . . . 719, 733, 921
932, 942, 963, 965		
\l__head_show_arguments:nnnnnn . . . . .		
31, 31, 451, 503, 565		
\l__head_show_arguments:nnnnnnnnnn . . . . .		
40, 40, 241, 376		
\l__head_start_code_tl . . . . .		
25, 26, 28, 30, 36, 225, 244, 248, 266, 274, 362, 381, 385, 387, 633, 647		
\l__head_subtitle_decls_tl 233, 370		
\l__head_subtitle_tl . 57, 88, 104, 121		
\l__head_title_decls_tl . . . . .		
232, 369, 489, 552, 605		
\l__head_title_format:n . . . . .		
446, 490, 498, 553, 560, 606		
\l__head_title_tl . . . . . 20, 22, 60, 82, 83, 84, 87, 98, 130, 135, 138, 141		

## I

\if... . . . . . 47
\IfBlankF . . . . . 672, 676
\IfBlankT . . . . . 689
\IfBlankTF . . . . . 674
\IfBooleanT . . . . . 70
\IfBooleanTF . . . . . 74, 904, 953
\ifcsname . . . . . 332
\ifdim . . . . . 827, 832
\IfInstanceExistsF . . . . . 822
\IfNoValueF . . . . . 478, 534, 593
\IfNoValueTF . . . . . 90, 871, 885, 957
\IfValueT . . . . . 71
\ignorespaces . . . . . 329, 441
instances:
headformat chapter . . . . . 795





<code>\tl_new:N</code> .....	52, 53, 54, 55, 56, 57, 58, 60, 61, 62, 63, 64, 65, 66	<code>\use:n</code> .....	94, 307, 407, 833, 847
<code>\tl_put_right:Nn</code> .....	123, 141, 145	<code>\use_i:nn</code> .....	313, 413
<code>\tl_set:Nn</code> .....	130, 134, 135, 210, 212, 214, 240, 248, 266, 281, 297, 347, 349, 351, 378, 385, 512, 525, 671, 908	<code>\use_ii:nn</code> .....	314, 414
<code>\tl_set_eq:NN</code> .....	82, 83, 84, 87, 88, 89, 239, 379	<code>\use_none:n</code> .....	9, 10
<code>\twocolumn</code> .....	292	<code>\UseInstance</code> .....	21, 47, 95, 308, 408
<code>\typeout</code> .....	26, 345, 355, 356	<code>\UseStructureName</code> .....	304, 402
<b>U</b>		<code>\UseTaggingSocket</code> .....	302, 303, 400, 401, 403, 432, 454, 492, 506, 544, 591
		<b>V</b>	
<code>\unskip</code> .....	431	<code>\vfil</code> .....	262, 283
<code>\UnusedTemplateKeys</code> .....	309, 409	<code>\vspace</code> .....	643, 649
use commands:		<code>\vspace*</code> .....	36
<code>\use:N</code> .....	682, 694	<b>W</b>	
		<code>\WordSpaceAmount</code>	184, 191, 198, <u>704</u> , 798, 928