The latex-lab-context package
Providing context for template instances and code that needs to know where and when it is executed

LaTeX Project*

v0.6a 2026-06-26

Abstract

Contents

1 Introduction **2**
1.1 Definition of a “context” . . . 2
1.2 The “primary context” . . . 2
1.3 The “secondary context” . . . 2
1.4 The “tertiary (font size) context” . . . 3

2 Setting context **3**

3 Context usage in template instances **4**

4 Notes **5**

5 Currently mainly internal interfaces **5**

6 Debugging **6**

7 Changes to internals of LaTeX **6**

8 Implementation **6**
8.0.1 Debugging . . . 6
8.1 Supporting font size changes as a tertiary context . . . 9
8.2 Supporting primary context . . . 9
8.2.1 Float context . . . 9
8.2.2 Caption context . . . 10
8.2.3 Footnote context . . . 10
8.2.4 Header context . . . 10
8.2.5 Footer context . . . 10
8.2.6 Marginal context . . . 10
8.3 Changes to lttemplates.dtx code . . . 10

*Initial implementation done by Frank Mittelbach

1 Introduction

In this module we implement the concept of "contexts" within the document and depending on the current context formatting behavior might change. The main application for this is in instances of templates (where things can be easily automated to react to context), but it is probably also applicable to other situations (but these might need more thought).

1.1 Definition of a "context"

The "context" is an attribute of every point of the formatted document, i.e., at each point during the formatting one can determine in which "context" the formatting happens and based on this adjust the formatting method.

The "context" is not an entirely flat structure: we distinguish between the "primary context", the "secondary context", and "tertiary context", all of which can be changed individually based on a number of rules as discussed below.

Any context is denoted by a name (`[a-z]*` letters only). The empty name is allowed to ease specification of the common case (i.e., the default "primary context" is the main galley and by default no "secondary context" and "tertiary context" are specified).

1.2 The "primary context"

The "primary context" is described with a fixed (but extensible?) set of names:

- ⟨*empty*⟩ denotes that we are in "galley" context producing text for the page;
- `caption` denotes that we are typesetting the caption text of a `\caption` command — this is also used for captions outside of floats,
- `footnote` denotes that we are typesetting the footnote text;
- `float` denotes that we are typesetting a float;
- `marginal` denotes that we are typesetting a marginal;
- `header` denotes that we are typesetting the running header;
- `footer` denotes that we are typesetting the running footer.

When the "primary context" is set it replaces the current "primary context" and resets the "secondary context" and "tertiary context" to ⟨*empty*⟩. The setting is local, i.e., it obeys grouping.

We may want to require that a context name is declared before its first use, e.g., via `\DeclarePrimaryContext` (right now this is not done).

1.3 The "secondary context"

The "secondary context" can be used to further sub-structure the primary context, e.g., with `float` as a primary, `figure` and `table` (e.g., the float types could form a secondary context).

1.4 The "tertiary (font size) context"

The "tertiary context" concerns itself only with font size changes, because that is most commonly a cause for layout changes. It is described through a fixed (but extensible) set of names:

- ⟨*empty*⟩ denotes that there is no special tertiary context, i.e., that we are producing material in `\normalsize`;
- `tiny` denotes that we are typesetting in `\tiny` font size;
- and similarly for `scriptsize`, `footnotesize`, `small`, `large`, `Large`, `LARGE`, `huge`, and `Huge`.

The above list of tertiary contexts are automatically set by the standard LaTeX font size commands (as part of `\@setfontsize`). Classes that use additional font size commands but do not use the LaTeX 2_ε command for this need to explicitly set the tertiary context with `\SetTertiaryContext` if they want their size be recognized as a context.

The "tertiary context" serves a double role: on the one hand it substructures the primary and secondary contexts further, e.g., ⟨*primary*⟩`:`⟨*secondary*⟩`:`⟨*tiertiary*⟩ allows you to specify different settings compared to ⟨*primary*⟩`:`⟨*secondary*⟩ but if there are only instance declarations for ⟨*primary*⟩ or ⟨*primary*⟩`:`⟨*secondary*⟩ and for `::`⟨*tertiary*⟩ then the latter is chosen over the the former.

2 Setting context

`\SetPrimaryContext` `\SetPrimaryContext {`⟨*new context*⟩`}`

The ⟨*new context*⟩ should be a name consisting only of lowercase letters a–z (and to make sense only a name that is actually used—typos will go undetected).

This declaration checks if there is a rule for the combination of the current context and the requested ⟨*new context*⟩ (see `\DeclarePrimaryContextRule`) and if so applies it to determine to which context the "primary context" should be set. If there is no such rule then ⟨*new context*⟩ will be used unconditionally.

As a second action the command resets the "secondary context" and "teriary context" to ⟨*empty*⟩, i.e., it clears the lower-level contexts. This is only done for the primary context. The reason is that when a primary context changes any sub-division (secondary and tertiary context) is no longer applicable and should therefore be reset.

The effects are local, i.e., both context changes revert to the previous value at the end of the current group.

`\DeclarePrimaryContextRule` `\DeclarePrimaryContextRule {`⟨*current*⟩`}{`⟨*new*⟩`}{`⟨*result*⟩`}`

This declaration defines the rule that when the current primary context is ⟨*current*⟩ and ⟨*new*⟩ is requested as the new context then instead ⟨*result*⟩ is made the new "primary context".

If `*` is used in the first argument then this indicates any current context and thus ⟨*new*⟩ is always replaced by ⟨*result*⟩. This can be more concisely written as

`\DeclarePrimaryContextRule* {`⟨*new*⟩`}{`⟨*result*⟩`}`

if preferred.

Declaring such rules is a global operation.

\SetSecondaryContext `\SetSecondaryContext {⟨new context⟩}`

The ⟨*new context*⟩ should be a name consisting only of lowercase letters a–z (and to make sense only a name that is actually used—typos will go undetected).

This declaration checks if there is a rule for the combination of the current secondary context and the requested ⟨*new context*⟩ (see `\DeclareSecondaryContextRule`) and if so applies it to determine to which context the "secondary context" should be set. If there is no such rule then ⟨*new context*⟩ will be used unconditionally.

This does not reset the "tertiary context".

The effect is local, i.e., the context change reverts to the previous value at the end of the current group.

\DeclareSecondaryContextRule `\DeclareSecondaryContextRule {⟨current⟩}{⟨new⟩}{⟨result⟩}`

This declaration defines the rule that when the current secondary context is ⟨*current*⟩ and ⟨*new*⟩ is requested as the new context then instead ⟨*result*⟩ is made the new "secondary context".

If `*` is used in the first argument then this indicates any current context and thus ⟨*new*⟩ is always replaced by ⟨*result*⟩.

Declaring such rules is a global operation.

TODO: we should perhaps provide

> `\DeclareSecondaryContextRule {`⟨*current primary*⟩`}{`⟨*current secondary*⟩`}{`⟨*new secondary*⟩`}{`⟨*result*⟩`}`

instead of the above. This would be a breaking change, but as this is currently only a trial implementation that should be ok.

3 Context usage in template instances

If a template instance is used via `\UseInstance{`⟨*type*⟩`}{`⟨*inst-name*⟩`}` then this normally results in calling up an instance of type ⟨*type*⟩ with the name ⟨*inst-name*⟩.[1]

However, when the "primary context" and/or the "secondary context" or "tertiary context" is non-empty then `\UseInstance` searches for an instance that is especially tailored to the current context. This works as follows:

- The string `:`⟨*primary context*⟩`:`⟨*secondary context*⟩`:`⟨*tertiary context*⟩ is appended to ⟨*inst-name*⟩ and if that instance exist it is used.[2]
- If not then ⟨*inst-name*⟩`:`⟨*primary context*⟩`:`⟨*secondary context*⟩ is tried next.
- If not then ⟨*inst-name*⟩`:`⟨*primary context*⟩ is tried next.
- If that doesn't exist either than the "tertiary context" is considered once more on its own by checking for ⟨*inst-name*⟩`:::`⟨*tertiary context*⟩ and using that if it exist.
- Otherwise ⟨*inst-name*⟩ is used as usual.

[1] Such instances are defined with `\DeclareInstance` or `\DeclareInstanceCopy`, see the documentation in `lttemplates-doc.pdf`.

[2] Note that this means that if the ⟨*primary context*⟩ and "secondary context" are empty we effectively append `:::`⟨*tertiary context*⟩.

This means it becomes trivial to alter the behavior of instances if they appear in a special typesetting context. For example, in LaTeX 2_ε display blocks, e.g., lists, center, etc., all changed their vertical spacing setup if the surrounding text was in `\small` or in `\footnotesize` (did nothing if you typeset in, say, `\Large` which had the strange effect that lists in `\Large` or `\huge` got whatever was set by a size command that changed list parameters).

With the new context model all you have to do is to provide additional instances, e.g., if `itemize-1` is the instance name for itemized lists on the first level then `itemize-1:footnote` would be the instance to be used if an itemize environment is used within a footnote.

In addition (or alternatively) you could setup `itemize-1:::Large` which would be selected if itemize is used anywhere and the fontsize is `\Large` (and there are no explicit specifications for the context otherwise).

4 Notes

With special classes, e.g., `ltx-talk`, additional primary and secondary contexts could be added (e.g., using the modes there).Could do with some experimentation what works best.

If the design of a class requires identical behavior in different contexts, e.g., the same behavior in `header` and `footer`, it is possible to simplify the setup by only specifying the design for the `header` context and then declaring:

```
\DeclarePrimaryContextRule{*}{footer}{header}
```

One can also add new primary or secondary contexts simply by making use of `\Set...Context` declarations with a new name and then use these named contexts when setting up special instance versions.

Bottom line, what is here is nothing more than a first, or rather by now, second prototype and likely to change to some extent.

5 Currently mainly internal interfaces

`\l_context_primary_str` This variable holds the current primary context name (or is empty if we are in the context of the main galley).

`\l_context_secondary_str` A sub-structure of the primary context, e.g., for the `float` context the secondary could be the float type.

`\l_context_tertiary_str` This variable holds the current tertiary (font size) context name (or is empty if we are typesetting in `\normalsize`). It is by default automatically set by all fontsize commands in core LaTeX.

6 Debugging

\DebugContextsOn
\DebugContextsOff
\context_debug_on:
\context_debug_off:

These commands enable/disable debugging messages for context related processing.

7 Changes to internals of LaTeX

\@setfontsize

This command has be augmented to execute `\SetSecondaryContext` with the name of the current fontsize command as the context argument.

\@float

This command has been augmented to execute `\SetPrimaryContext{float}`. This results in the body of all real floats being typeset in the primary `float` context. Not covered are packages or methods that make floats not using `\@float` or making pseudo floats, e.g., something not floating but having a caption.

8 Implementation

```
⟨*package⟩
⟨@@=tag⟩

\ProvidesExplPackage {latex-lab-testphase-context}
                     {\ltlabcontextdate}
                     {\ltlabcontextversion}
  {Providing context for instance, etc.}

⟨*package⟩
⟨@@=context⟩
```

8.0.1 Debugging

\g__context_debug_bool

```
\bool_new:N \g__context_debug_bool
```

__context_debug:n
__context_debug_typeout:n

```
\cs_new_eq:NN \__context_debug:n \use_none:n
\cs_new_eq:NN \__context_debug_typeout:n \use_none:n
```

(*End of definition for* `\__context_debug:n` *and* `\__context_debug_typeout:n`.)

\context_debug_on:
\context_debug_off:
__context_debug_gset:

```
\cs_new_protected:Npn \context_debug_on:
  {
    \bool_gset_true:N \g__context_debug_bool
    \__context_debug_gset:
  }

\cs_new_protected:Npn \context_debug_off:
  {
    \bool_gset_false:N \g__context_debug_bool
    \__context_debug_gset:
  }

\cs_new_protected:Npn \__context_debug_gset:
  {
    \cs_gset_protected:Npe \__context_debug:n ##1
      { \bool_if:NT \g__context_debug_bool {##1} }
    \cs_gset_protected:Npe \__context_debug_typeout:n ##1
      { \bool_if:NT \g__context_debug_bool { \typeout{[Context]~ ==>~ ##1} } }
  }
```

(*End of definition for* `\context_debug_on:` , `\context_debug_off:` , *and* `\__context_debug_gset:`. *These functions are documented on page* 6.)

\DebugContextsOn
\DebugContextsOff

```
\cs_new_protected:Npn \DebugContextsOn  { \context_debug_on:  }
\cs_new_protected:Npn \DebugContextsOff { \context_debug_off: }

\DebugContextsOff
```

(*End of definition for* `\DebugContextsOn` *and* `\DebugContextsOff`. *These functions are documented on page* 6.)

\l_context_primary_str

Variable to hold the name of the current primary context.

```
\str_new:N \l_context_primary_str
```

(*End of definition for* `\l_context_primary_str`. *This function is documented on page* 5.)

\SetPrimaryContext

Change the context: if the target context is empty, just do it. Otherwise if we have a rule for the current context and new context use that; otherwise if there is a star rule apply that; otherwise set the new value as requested,

```
\cs_new_protected:Npn \SetPrimaryContext #1 {
  \str_set:Ne \l_context_primary_str
     { \tl_if_empty:nF { #1 }
       { \cs_if_exist_use:cF
         { g__context_primary_ \l_context_primary_str _ #1 _tl }
         { \cs_if_exist_use:cF
           { g__context_primary_ * _ #1 _tl }
           { #1 }
         }
       }
     }
  \__context_debug_typeout:n{set~primary~ <-~ \l_context_primary_str }
```

Also reset the secondary and tertiary context.

```
  \SetSecondaryContext {}
  \SetTertiaryContext {}
}
```

(*End of definition for* `\SetPrimaryContext`. *This function is documented on page* 3.)

`\DeclarePrimaryContextRule` Rules are simply stored in macro names:

```
\cs_new_protected:Npn \DeclarePrimaryContextRule #1#2#3 {
  \tl_gclear_new:c { g__context_primary_ #1 _ #2 _tl }
  \tl_gset:cn      { g__context_primary_ #1 _ #2 _tl } {#3}
}
```

(*End of definition for* `\DeclarePrimaryContextRule`. *This function is documented on page* 3.)

`\l_context_secondary_str` Variable to hold the name of the current secondary context.

```
\str_new:N \l_context_secondary_str
```

(*End of definition for* `\l_context_secondary_str`. *This function is documented on page* 5.)

`\SetSecondaryContext` Similar to `\SetPrimaryContext` but without resetting the tertiary context.

```
\cs_new_protected:Npn \SetSecondaryContext #1 {
  \str_set:Ne \l_context_secondary_str
     { \tl_if_empty:nF { #1 }
       { \cs_if_exist_use:cF
         { g__context_secondary_ \l_context_secondary_str _ #1 _tl }
         { \cs_if_exist_use:cF
           { g__context_secondary_ * _ #1 _tl }
           { #1 }
         }
       }
     }
  \__context_debug_typeout:n{set~ secondary~ <-~ \l_context_secondary_str }
}
```

(*End of definition for* `\SetSecondaryContext`. *This function is documented on page* 4.)

`\DeclareSecondaryContextRule`

```
\cs_new_protected:Npn \DeclareSecondaryContextRule #1#2#3 {
  \tl_gclear_new:c { g__context_secondary_ #1 _ #2 _tl }
  \tl_gset:cn      { g__context_secondary_ #1 _ #2 _tl } {#3}
}
```

(*End of definition for* `\DeclareSecondaryContextRule`. *This function is documented on page* 4.)

`\SetTertiaryContext`
`\DeclareTertiaryContextRule`
`\l_context_tertiary_str`

e-type

```
\cs_new_protected:Npn \SetTertiaryContext #1 {
  \str_set:Ne \l_context_tertiary_str
     { \tl_if_empty:nF { #1 }
       { \cs_if_exist_use:cF
         { g__context_tertiary_ \l_context_tertiary_str _ #1 _tl }
         { \cs_if_exist_use:cF
           { g__context_tertiary_ * _ #1 _tl }
           { #1 }
```

```
          }
        }
      }
  \__context_debug_typeout:n{set~ tertiary~ <-- \l_context_tertiary_str }
}

\cs_new_protected:Npn \DeclareTertiaryContextRule #1#2#3 {
  \tl_gclear_new:c { g__context_tertiary_ #1 _ #2 _tl }
  \tl_gset:cn      { g__context_tertiary_ #1 _ #2 _tl } {#3}
}

\str_new:N \l_context_tertiary_str

```

(*End of definition for* `\SetTertiaryContext`, `\DeclareTertiaryContextRule`, *and* `\l_context_tertiary_str`. *These functions are documented on page* **??**.)

8.1 Supporting font size changes as a tertiary context

`\@setfontsize` We try to be careful when setting the name as we don't know if `\string` returns a backslash or not.

```
\def\@setfontsize#1#2#3{\@nomath#1%
  \ifx\protect\@typeset@protect
  \let\@currsize#1%
  \begingroup
    \escapechar\m@ne
    \expandafter
  \endgroup
    \expandafter
  \SetTertiaryContext
    \expandafter {\string#1}%
  \fi
  \fontsize{#2}{#3}\selectfont
}
```

If we are typesetting in `\normalsize` we don't want that as the context, so here is a first application of a rule.

```
\DeclareTertiaryContextRule{*}{normalsize}{}
```

(*End of definition for* `\@setfontsize`. *This function is documented on page 6.*)

8.2 Supporting primary context

8.2.1 Float context

`\@float` This should work with most float pages. There are of course some classes or packages that produce pseudo-floats without calling `\@float`. These need to be handled separately by adding a `\SetPrimaryContext` in their code.

```
\AddToHook{cmd/@float/before}{\SetPrimaryContext{float}}
```

(*End of definition for* `\@float`. *This function is documented on page 6.*)

8.2.2 Caption context

[**TODO:** in latex-lab-float] `\@makecaption` is redefined there and the context should be set in that command (or its replacement one day).

[**TODO:** longtable and anything else that runs around producing `\caption`s]

[**TODO:** Again, the secondary context should be the current float type]

8.2.3 Footnote context

[**TODO:** in latex-lab-footnote]

8.2.4 Header context

[**TODO:**]

8.2.5 Footer context

[**TODO:**]

8.2.6 Marginal context

[**TODO:**]

8.3 Changes to lttemplates.dtx code

```
⟨@@=template⟩
```

__template_use_instance_auxii:nn
__template_use_instance_finally:nn

The `\__template_use_instance_auxii:nn` is the command that is used by `\UseInstance` to select and execute a requested instance. So we now have to have a little more logic here.

In the normal case (both primary and secondary context are empty) we now have 2 tests. We have up to 6 tests if one of the contexts is non-empty.

```
\cs_new_eq:NN \__template_use_instance_finally:nn \__template_use_instance_auxii:nn


\cs_set_protected:Npn \__template_use_instance_auxii:nn #1#2 {
  \exp_args:Ne\str_if_empty:nTF
     { \l_context_primary_str \l_context_secondary_str \l_context_tertiary_str }
     {
       \__context_debug_typeout:n{no~ special~ context}
       \__template_if_instance_exist:nnTF {#1} {#2}
          {
            \__context_debug_typeout:n{basic~ instance~ exists}
            \__template_use_instance_finally:nn {#1} {#2} }
          {
            \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
     }
     {
       \__template_debug:n {
         \str_if_empty:NF \l_context_primary_str
             { \__template_debug_typeout:n {primary~ context~ is~
                 '\l_context_primary_str' } }
```

```
    \str_if_empty:NF \l_context_secondary_str
        { \__template_debug_typeout:n {secondary~ context~ is~
            '\l_context_secondary_str' } }
    \str_if_empty:NF \l_context_tertiary_str
        { \__template_debug_typeout:n {tertiary~ context~ is~
            '\l_context_tertiary_str' } }
  }
  \__template_if_instance_exist:nnTF {#1}
     { #2 : \l_context_primary_str :
       \l_context_secondary_str :
       \l_context_tertiary_str  }
     {
       \__context_debug_typeout:n{:primary:secondary:tertiary~
         exists~ (primary~ or~ secondary~ might~ be~ empty)}
       \__template_use_instance_finally:nn {#1}
          { #2 : \l_context_primary_str :
            \l_context_secondary_str :
            \l_context_tertiary_str  }
     }
     {
       \__context_debug_typeout:n{:primary:secondary:tertiary~ does~ not~ exist}
       \__template_if_instance_exist:nnTF {#1}
          { #2 : \l_context_primary_str : \l_context_secondary_str  }
          {
            \__context_debug_typeout:n{:primary:secondary~
              or~ ::secondary~ exists}
            \__template_use_instance_finally:nn {#1}
               { #2 : \l_context_primary_str : \l_context_secondary_str }
          }
          {
            \__context_debug_typeout:n{:primary:secondary~ and~
              ::secondary~ do~ not~ exist}
            \__template_if_instance_exist:nnTF {#1}
               { #2 : \l_context_primary_str }
               {
                 \__context_debug_typeout:n{:primary~ exists}
                 \__template_use_instance_finally:nn {#1}
                    { #2 : \l_context_primary_str }
               }
               {
                 \__context_debug_typeout:n{:primary~ does~ not~ exist}
                 \__template_if_instance_exist:nnTF {#1}
                   { #2 : : : \l_context_tertiary_str  }
                    {
                      \__context_debug_typeout:n{:::tertiary~ exists}
                      \__template_use_instance_finally:nn {#1}
                         { #2 : : : \l_context_tertiary_str  }
                    }
                    {
                      \__template_if_instance_exist:nnTF {#1} {#2}
                         {
                           \__context_debug_typeout:n{basic~ instance~ exists}
                           \__template_use_instance_finally:nn {#1} {#2} }
                         {
```

```
                                    \msg_error:nnnn { template } { unknown-instance } {#1} {#2} }
                                }
                            }
                        }
                    }
                }
            }
}

```

(*End of definition for* `\__template_use_instance_auxii:nn` *and* `\__template_use_instance_finally:nn`.)

⟨/package⟩