

Vesti Transpiler User Manual

Sungbae Jeong

November 5, 2025

Contents

1	Introduction	1
2	Structure of Vesti File	1
3	Keywords	1
3.1	<code>docclass</code> keyword	2
3.2	<code>importpkg</code> keywords	2
3.3	<code>startdoc</code> keyword	2
3.4	<code>useenv</code> keyword	2
3.5	<code>begenv</code> keyword and <code>endenv</code> keyword	3
4	Builtins	3
5	Source Code of This Document	4

1 Introduction

#get_file

2 Structure of Vesti File

Vesti is similar as L^AT_EX. Its structure consists with two parts: `preamble` and `main`. Preamble is the place where L^AT_EX documentclass, packages, and several settings are located. Main body is where actual documentation is located. Below figure is the simple Vesti documentation.

```
1 docclass article (10pt)
2 importpkg {
3     geometry (a4paper, margin=2.2cm)
4 }
5 startdoc
6 Hello, Vesti!
```

We will see later, but the very difference with L^AT_EX is that Vesti has its own keywords (keywords are colored with purple). It makes the code readable and it is easier and faster to write the document. The keyword `startdoc` splits the preamble and the main part of the documentation similar with

`\begin{document}` in L^AT_EX. However, Vesti does not have the analogous part of `\end{document}`, because almost every L^AT_EX document (99.999% I'm sure) does not have any code below `\end{document}`. For this reason, Vesti automatically ends document when EOF (End Of File) is found.

3 Keywords

Followings are reserved as keywords. In this document, every Vesti keyword has the form like `this`.

```

begenv      compty      cpfile      defenv
defun       docclass     endenv      importmod
importpkg   importves   startdoc   useenv

```

Table 1: Keywords in Vesti

3.1 `docclass` keyword

Keyword `docclass` is an analogous of `\documentclass` in L^AT_EX. If `docclass` is in the main paragraph, it acts just a normal word. In other words, `docclass` actives only in the preamble. The syntax of `docclass` is following:

```
docclass <class name> (<arguments>)
```

Here, arguments are separated by commas and embraced by `()`. Here are some examples.

- `docclass article`
- `docclass article (10pt)`
- `docclass article (10pt, twocols)`
- `docclass article (10pt,twocols)`

3.2 `importpkg` keywords

Keyword `importpkg` is an analogous of `\usepackage` in L^AT_EX. If `importpkg` is in the main paragraph, it acts just a normal word. In other words, `importpkg` actives only in the preamble.

`importpkg` has two different syntax. First one is same as `docclass`.

```
importpkg pkg-name (<arguments>)
```

Here, arguments are separated by commas and embraced by `()`. In the practical case, one should include several packages with options. `importpkg` also supports such case. We will look at an example instead of giving rigorous grammar.

```

1 importpkg {
2     amsmath, amssymb, amsthm,
3     geometry (a4paper, margin=2.2cm),
4 }
```

As one can see, inside of `{}`, several packages can be used together with their options.

3.3 `startdoc` keyword

Keyword `startdoc` tells to Vesti that the main document starts. In the main document, you can also write `startdoc` in the main document. In that case, `startdoc` does nothing.

3.4 `useenv` keyword

As the name implies, keyword `useenv` is an analogous of `\begin{...}` and `\end{...}` pair in L^AT_EX. The simplest `useenv` is like this.

```
useenv center {
    Hello, World!    or    useenv center { Hello, World! }
}
```

As you can see, `useenv center` is the part of `\begin{center}`, and the single `}` is the part of `\end{center}`. Since Vesti knows their pair, one can write a code with several environment, and each pair is properly matched. For instance, above example is written in Vesti like follows. Here, `\useenv` just prints `useenv` in that style.

```

1 useenv figure [ht] {
2   \centering
3   useenv tikzpicture {
4     useenv scope {
5       \path (0,0) node {\vbox{
6         %#\hbox{\tt\useenv center \{}}
7         %#\hbox{\tt\obeyspaces Hello, World!}
8         %#\hbox{\tt\obeyspaces\}}
9       }};
10    }
11    \path (2.3,0) node {or};
12    useenv scope [shift={(6,0)}] {
13      \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
14    }
15  }
16 }
```

Full syntax about `useenv` is the following.

`useenv <environment name> <argument>* { <body> }`

where '*' means that the number of `<argument>` is zero or at least one, and

$$\langle \text{argument} \rangle = \begin{cases} (\langle \text{argument} \rangle) & \text{mandatory arguments} \\ [\langle \text{argument} \rangle] & \text{optional arguments} \end{cases}$$

For instance, below one is a valid Vesti code (environment `foo` is undefined in general). As one can see, spaces can exist in between `<argument>`s.

```

1 useenv foo (asd)(fff)[\ames and \awdsa] (askws) [\rrsaa] {
2   foobar
3 }
```

3.5 `begenv` keyword and `endenv` keyword

As the name implies, both keywords `begenv` and `endenv` are analogous of `\begin{...}` and `\end{...}` pair in L^AT_EX, respectively. Thus below code

```

1 begenv center
2   asdsad
3 endenv center
```

is exactly same as

```

1 useenv center {
2   asdsad
3 }
```

Then why we need `begenv` and `endenv` if we already have `useenv`?

4 Builtins

Vesti also has its own builtin functions, which are prefixed with #. One might wonder what distinguishes builtins from keywords. In fact, from the compiler's internal perspective, there is no real difference. However, in actual language usage, constantly typing the prefix can be somewhat tedious, especially for functions that are commonly used.

From the perspective of language design –particularly in Vesti– it is sometimes desirable to use names that cannot serve as keywords. For example, Vesti provides a built-in function `#label1`, which will be explained later. Since Vesti is a typewriting-oriented language, the word “label” is often used in its ordinary sense rather than in its special semantic meaning within the language.

Followings are reserved as builtin functions.

```

#chardef      #enum         #eq          #get_filepath    #label
#ltx3_off    #ltx3_on     #makeatletter  #makeatother    #mathchardef
#mathmode     #noltx3     #nonstopmode   #picture       #showfont
#textmode    #xparsen

```

Table 2: Builtins in Vesti

5 Source Code of This Document

Below code was generated by inline lua.

```

1 docclass article (10pt)
2 importpkg {
3     geometry (a4paper, margin = 2.2cm),
4     xcolor,
5     tikz,
6     fancyvrb,
7 }
8
9 \title{Vesti Transpiler User Manual}
10 \author{Sungbae Jeong}
11
12 importves (font.ves)
13
14 % read file contents using lua
15 #lu:
16 local function read_all(path)
17     local f, err = io.open(path, "rb")
18     assert(f, ("cannot open %s: %s"):format(path, err))
19     local data = f:read("*a")
20     f:close()
21     return data
22 end
23 :lu#<readAll>
24
25 % definition of \keyword command
26 #xparse defun [!] keyword (m) {{\tt\color{purple}\#1}}
27 #xparse defun [!] builtin (v) {{\tt\color{yellow!70!black}\#\#1}}
28 #xparse defun useenv (s) {\IfBooleanTF{#1}{\keyword{\%-useenv-}}{\keyword{\%-useenv-}} }
29 #xparse defun begenv (s) {\IfBooleanTF{#1}{\keyword{\%-begenv-}}{\keyword{\%-begenv-}} }
30 #xparse defun endenv (s) {\IfBooleanTF{#1}{\keyword{\%-endenv-}}{\keyword{\%-endenv-}} }
31
32 defenv [p] foo (m) {begenv center #1 and #1}{endenv center}
33
34 startdoc
35 \maketitle
36 \tableofcontents
37
38 \section{Introduction}
39 \builtin|get_file|
40
41 \section{Structure of Vesti File}
42 Vesti is similar as \LaTeX. Its structure consists with two parts: {\tt preamble} and
43 {\tt main}. Preamble is the place where \LaTeX\ documentclass, packages, and
44 several settings are located. Main body is where actual documentation is located.
45 Below figure is the simple Vesti documentation.
46
47 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
48 %#|+color|purple@docclass@ article (10pt)
49 %#|+color|purple@importpkg@ {

```

```

50  %#      geometry (a4paper, margin=2.2cm)
51  %#}
52  %#|+color|purple@startdoc@
53  %#Hello, Vesti!
54  }

55
56 We will see later, but the very difference with \LaTeX\ is that Vesti has its
57 own keywords (keywords are colored with purple). It makes the code readable and
58 it is easier and faster to write the document. The keyword startdoc splits
59 the preamble and the main part of the documentation similar with
60
61 % Don't ask why I chose Q for catcode 0.
62 %#{\tt\catcode`Q=0 Q\catcode`\\=12 \begin{document}Q}} in \LaTeX.
63 However, Vesti does not have the analogous part of
64 %#{\tt\catcode`Q=0 Q\catcode`\\=12 \end{document}Q},
65 because almost every \LaTeX\ document (99.999\% I'm sure) does not have any code
66 below %#{\tt\catcode`Q=0 Q\catcode`\\=12 \end{document}Q}.
67 For this reason, Vesti automatically ends document when EOF (End Of File) is
68 found.

69
70 \section{Keywords}
71 Followings are reserved as keywords. In this document, every Vesti keyword has
72 the form like \keyword{this}.
73 useenv table [ht] {
74     \centering
75     #lu:
76     local content = read_all("../src/lexer/Token.zig")

77
78     -- Lua's built-in patterns don't support lookahead.
79     -- We capture both the keyword and the TokenType, then filter out 'deprecated'.
80     -- Pattern breakdown:
81     -- %.%{          => matches ".{"
82     -- %s*(["]+)    => a quoted string -> capture 1
83     -- %s*,%s*TokenType%.([%w_]++) => TokenType.<Name> -> capture 2
84     local pat = "%.%{%"s*"\\"(["]+)\\"%s*,%"s*TokenType%.([%w_]++)"

85
86     local keywords = {}
87     for name, tok in content:gmatch(pat) do
88         if tok ~= "deprecated" then
89             keywords[#keywords + 1] = name
90         end
91     end

92
93     table.sort(keywords)

94
95     vesti.print([[\begin{tabular}{cccc}]])

96
97     for i, kw in ipairs(keywords) do
98         local cell = string.format("\\"keyword{%s}", kw)
99         if (i % 4) == 0 then
100             vesti.print(cell .. [[\\]])
101         else
102             vesti.print(cell .. "&")
103         end
104     end

105
106     vesti.print([[\end{tabular}]])
107     :lu#[readAll]
108     \caption{Keywords in Vesti}
109 }

```

```

110 \subsection{\keyword{docclass} keyword}
111 Keyword \keyword{docclass} is an analogous of \verb|\documentclass| in \LaTeX.
112 If \keyword{docclass} is in the main paragraph, it acts just a normal word.
113 In other words, \keyword{docclass} actives only in the preamble.
114 The syntax of \keyword{docclass} is following:
115
116
117 \useenv center {
118     \keyword{docclass}\kern0.5em <class name>\kern0.5em {\tt{}<arguments>{\tt{}}
119 }
120 Here, arguments are separated by commas and embraced by {\tt ()}. Here are some
121 examples.
122
123 \goodbreak
124 \useenv itemize {
125     \item \keyword{docclass} {\tt article}
126     \item \keyword{docclass} {\tt article (10pt)}
127     \item \keyword{docclass} {\tt article (10pt, twocols)}
128     \item \keyword{docclass} {\tt article (10pt,twocols)}
129 }
130
131 \subsection{\keyword{importpkg} keywords}
132 Keyword \keyword{importpkg} is an analogous of \verb|\usepackage| in \LaTeX.
133 If \keyword{importpkg} is in the main paragraph, it acts just a normal word.
134 In other words, \keyword{importpkg} actives only in the preamble.
135
136 importpkg has two different syntax. First one is same as docclass.
137 \useenv center {
138     \keyword{importpkg}\kern1em pkg-name\kern1em {\tt{}<arguments>{\tt{}}
139 }
140 Here, arguments are separated by commas and embraced by {\tt ()}.
141 In the practical case, one should include several packages with options.
142 importpkg also supports such case. We will look at an example instead of
143 giving rigorous grammar.
144 \useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
145 %#|+color|purple@importpkg@ {
146 %#     amsmath, amssymb, amsthm,
147 %#     geometry (a4paper, margin=2.2cm),
148 %#}
149 }
150
151 \noindent As one can see, inside of \verb|{|}|, several packages can be used
152 together with thier options.
153
154 \subsection{\keyword{startdoc} keyword}
155 Keyword \keyword{startdoc} tells to Vesti that the main document starts. In the
156 main document, you can also write \keyword{startdoc} in the main document. In
157 that case, \keyword{startdoc} does nothing.
158
159 \subsection{\useenv keyword}
160 As the name implies, keyword \useenv is an analogous of \verb|\begin{...}| and
161 \verb|\end{...}| pair in \LaTeX.
162 The simplest \useenv is like this.
163
164 \useenv figure [ht] {
165     \centering
166     \useenv tikzpicture {
167         \useenv scope {
168             \path (0,0) node {\vbox{
169                 %#\hbox{\tt\useenv center \{}}

```

```

170      %#\hbox{\tt\obeyspaces Hello, World!}
171      %#\hbox{\tt\obeyspaces{}}
172  };
173 }
174 \path (2.3,0) node {or};
175 useenv scope [shift={(6,0)}]{
176     \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
177 }
178 }
179 }
180
181 As you can see, {\tt\useenv center} is the part of \verb|\begin{center}|, and
182 the single {\tt\}} is the part of \verb|\end{center}|. Since Vesti knows their
183 pair, one can write a code with several environment, and each pair is properly
184 matched. For instance, above example is written in Vesti like follows. Here,
185 \verb|\useenv| just prints \useenv in that style.
186
187 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
188 %#|+color|purple@useenv@ figure [ht] {
189 %#   \centering
190 %#   |+color|purple@useenv@ tikzpicture {
191 %#     |+color|purple@useenv@ scope {
192 %#       \path (0,0) node {\vbox{
193 %#         |+color|blue@%#@\hbox{\tt\useenv center \{}}
194 %#         |+color|blue@%#@\hbox{\tt\obeyspaces Hello, World!}
195 %#         |+color|blue@%#@\hbox{\tt\obeyspaces\}}
196 %#       }};
197 %#     }
198 %#     \path (2.3,0) node {or};
199 %#     |+color|purple@useenv@ scope [shift={(6,0)}] {
200 %#       \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
201 %#     }
202 %#   }
203 %# }
204 }
205
206 Full syntax about \useenv is the following.
207 useenv center {
208   \useenv\kern0.5em <environment name>\kern0.5em <argument>*\kern0.5em
209   {\tt\{} <body> {\tt\}}
210 }
211 where '*' means that the number of <argument> is zero or at least one, and
212 $$ 
213   "<argument>" = useenv cases {
214     "<argument>" & "mandatory arguments" \\
215     "[<argument>]" & "optional arguments"
216   }
217 $$ 
218 For instance, below one is a valid Vesti code (environment {\tt foo} is
219 undefined in general). As one can see, spaces can exist in between <argument>s.
220 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
221 %#|+color|purple@useenv@ foo (asd)(fff)[\ames and \awdsa] (askws) [\rrsaa] {
222 %#   foobar
223 %#}
224 }
225
226 \subsection{\begenv keyword and \endenv keyword}
227 As the name implies, both keywords \begenv and \endenv are analogous of
228 \verb|\begin{...}| and \verb|\end{...}| pair in \LaTeX, respectively.
229 Thus below code

```

```

230 |useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
231 |%#|+color|purple@begenv@ center
232 |%#    asdsad
233 |%#|+color|purple@endenv@ center
234 |}
235 |is exactly same as
236 |useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
237 |%#|+color|purple@useenv@ center {
238 |%#    asdsad
239 |%#}
240 |}
241 |Then why we need \begenv and \endenv if we already have \useenv*?
242 |
243 |\section{Builtins}
244 |Vesti also has its own builtin functions, which are prefixed with \#.
245 |One might wonder what distinguishes builtins from keywords. In fact, from the
246 |compiler's internal perspective, there is no real difference. However, in actual
247 |language usage, constantly typing the prefix can be somewhat tedious, especially
248 |for functions that are commonly used.
249 |
250 |From the perspective of language design --particularly in Vesti-- it is sometimes
251 |desirable to use names that cannot serve as keywords. For example, Vesti
252 |provides a built-in function {\tt\#label}, which will be explained later. Since Vesti
253 |is a typewriting-oriented language, the word \lq\lq label\rq\rq\ is often used in its
254 |ordinary sense rather than in its special semantic meaning within the language.
255 |
256 |Followings are reserved as builtin functions.
257 |
258 |useenv table [ht] {
259 |    \centering
260 |    #lu:
261 |    local content = read_all("../src/lexer/Token.zig")
262 |
263 |    -- match .{ "here" }
264 |    local pat = "%.%{%"s*\"([^\"]+)\\"%s*%}"
265 |
266 |    local builtins = {}
267 |    for name, tok in content:gmatch(pat) do
268 |        builtins[#builtins + 1] = name
269 |    end
270 |    table.sort(builtins)
271 |
272 |    vesti.print([[begin{tabular}{ccccc}]])
273 |
274 |    for i, kw in ipairs(builtins) do
275 |        local cell = string.format("\\#\\verb@%s@", kw)
276 |        if (i % 5) == 0 then
277 |            vesti.print(cell .. [[\\]])
278 |        else
279 |            vesti.print(cell .. "&")
280 |        end
281 |    end
282 |
283 |    vesti.print([[end{tabular}]])
284 |    :lu#[readAll]
285 |    \caption{Builtins in Vesti}
286 |}
287 |
288 |\section{Source Code of This Document}
289 |Below code was generated by inline lua.

```

```
290 |useenv Verbatim [numbers=left, numbersep=5pt, frame=single] {  
291 |#lu:  
292 |    local content = read_all("vesti_man.ves")  
293 |    for line in content:gmatch("(^\r\n*)\r?\n?") do  
294 |        vesti.print(line)  
295 |    end  
296 |:lu#[readAll]  
297 |}  
298 |
```