

Vesti Transpiler User Manual

Sungbae Jeong

October 29, 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

2 Structure of Vesti File

Vesti is similar as \LaTeX . Its structure consists with two parts: `preamble` and `main`. Preamble is the place where \LaTeX 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 \LaTeX 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 \LaTeX . However, Vesti does not have the analogous part of `\end{document}`, because almost every \LaTeX 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

```

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 defun [!] keyword (m) {{\tt\color{purple}\#1}}
27 defun useenv (s) {\IfBooleanTF{\#1}{\keyword{%-useenv-}}{\keyword{%-useenv-}} }
28 defun begenv (s) {\IfBooleanTF{\#1}{\keyword{%-begenv-}}{\keyword{%-begenv-}} }
29 defun endenv (s) {\IfBooleanTF{\#1}{\keyword{%-endenv-}}{\keyword{%-endenv-}} }
30
31 startdoc
32 \maketitle
33 \tableofcontents
34
35 \section{Introduction}
36
37 \section{Structure of Vesti File}
38 Vesti is similar as \LaTeX. Its structure consists with two parts: {\tt preamble} and
39 {\tt main}. Preamble is the place where \LaTeX\ documentclass, packages, and
40 several settings are located. Main body is where actual documentation is located.
41 Below figure is the simple Vesti documentation.
42
43 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
44 %#|+color|purple@docclass@ article (10pt)
45 %#|+color|purple@importpkg@ {
46 %#     geometry (a4paper, margin=2.2cm)
47 %#}
48 %#|+color|purple@startdoc@
49 %#Hello, Vesti!

```

```

50 }
51
52 We will see later, but the very difference with \LaTeX\ is that Vesti has its
53 own keywords (keywords are colored with purple). It makes the code readable and
54 it is easier and faster to write the document. The keyword startdoc splits
55 the preamble and the main part of the documentation similar with
56
57 % Don't ask why I chose Q for catcode 0.
58 %#{\tt\catcode`Q=0 Q\catcode`\\=12 \begin{document}{} in \LaTeX.
59 However, Vesti does not have the analogous part of
60 %#{\tt\catcode`Q=0 Q\catcode`\\=12 \end{document}{},
61 because almost every \LaTeX\ document (99.999\% I'm sure) does not have any code
62 below %#{\tt\catcode`Q=0 Q\catcode`\\=12 \end{document}{}.
63 For this reason, Vesti automatically ends document when EOF (End Of File) is
64 found.
65
66 \section{Keywords}
67 Followings are reserved as keywords. In this document, every Vesti keyword has
68 the form like \keyword{this}.
69 useenv table [ht] {
70     \centering
71     #lu:
72     local content = read_all("../src/lexer/Token.zig")
73
74     -- Lua's built-in patterns don't support lookahead.
75     -- We capture both the keyword and the TokenType, then filter out 'deprecated'.
76     -- Pattern breakdown:
77     --   %.%{           => matches ".{"
78     --   %s*"([""]+)" => a quoted string -> capture 1
79     --   %s*,%s*TokenType%.([%w_]++) => TokenType.<Name> -> capture 2
80     local pat = "%.%{%" s*"([""]+)"%" s*,%" s*TokenType%.([%w_]++)"
81
82     local keywords = {}
83     for name, tok in content:gmatch(pat) do
84         if tok ~= "deprecated" then
85             keywords[#keywords + 1] = name
86         end
87     end
88
89     table.sort(keywords)
90
91     vesti.print([[\begin{tabular}{cccc}]])
92
93     for i, kw in ipairs(keywords) do
94         local cell = string.format("\\" keyword{%" s"}, kw)
95         if (i % 4) == 0 then
96             vesti.print(cell .. [[\\]])
97         else
98             vesti.print(cell .. "&")
99         end
100    end
101
102    vesti.print([[\end{tabular}]])
103    :lu#[readAll]
104    \caption{Keywords in Vesti}
105 }
106
107 \subsection{\keyword{docclass} keyword}
108 Keyword \keyword{docclass} is an analogous of \verb|\documentclass| in \LaTeX.
109 If \keyword{docclass} is in the main paragraph, it acts just a normal word.

```

```

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

```

```

170     \path (2.3,0) node {or};
171     useenv scope [shift={(6,0)}]{
172         \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
173     }
174 }
175 }

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

182
183 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
184 %#|+color|purple@useenv@ figure [ht] {
185 %#   \centering
186 %#   |+color|purple@useenv@ tikzpicture {
187 %#     |+color|purple@useenv@ scope {
188 %#       \path (0,0) node {\vbox{
189 %#         |+color|blue@%#@\hbox{\tt\useenv center \{}}
190 %#         |+color|blue@%#@\hbox{\tt\obeyspaces Hello, World!}
191 %#         |+color|blue@%#@\hbox{\tt\obeyspaces\}}
192 %#       }};
193 %#     }
194 %#     \path (2.3,0) node {or};
195 %#     |+color|purple@useenv@ scope [shift={(6,0)}] {
196 %#       \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
197 %#     }
198 %#   }
199 %# }
200 }

201 Full syntax about \useenv is the following.
202 useenv center {
203   \useenv{kern0.5em <environment name>\kern0.5em <argument>*\kern0.5em
204     {\tt\{} <body> {\tt\}}
205   }
206
207 where '*' means that the number of <argument> is zero or at least one, and
208 $$

209 "<argument>" = useenv cases {
210   "<argument>" & "mandatory arguments" \\
211   "[<argument>]" & "optional arguments"
212 }

213 $$

214 For instance, below one is a valid Vesti code (environment {\tt foo} is
215 undefined in general). As one can see, spaces can exist in between <argument>s.
216 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
217 %#|+color|purple@useenv@ foo (asd)(fff)[\ames and \awdsa] (askws) [\rrsaa] {
218 %#   foobar
219 %# }
220 }

221 \subsection{\begenv keyword and \endenv keyword}

222 As the name implies, both keywords \begenv and \endenv are analogous of
223 \verb|\begin{...}| and \verb|\end{...}| pair in \LaTeX, respectively.
224 Thus below code

225 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
226 %#|+color|purple@begenv@ center
227 %#   asdsad
228 %#|+color|purple@endenv@ center

```

```

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

```

```
290     vesti.print(line)
291     end
292 :lu#[readAll]
293 }
294
```