

# Vesti Transpiler User Manual

Sungbae Jeong

October 19, 2025

## 1 Introduction

## 2 Language Reference

### 2.1 Structure of Vesti File

Vesti is similar as  $\text{\LaTeX}$ . Its structure consists with two parts: preamble and main. Preamble is the place where  $\text{\LaTeX}$  documentclass, packages, and several settings are located. Main body is where actual documentation is located. Below figure is the simple Vesti documentation.

```
docclass article (10pt)
importves {
    geometry (a4paper, margin=2.2cm)
}
startdoc
Hello, Vesti!
```

Figure 1: Almost very simple Vesti documentation

We will see later, but the very difference with  $\text{\LaTeX}$  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  $\text{\LaTeX}$ . However, Vesti does not have the analogous part of `\end{document}`, because almost every  $\text{\LaTeX}$  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.

### 2.2 Keywords

Followings are reserved as keywords.

<code>begenv</code>	<code>compty</code>	<code>cpfile</code>	<code>defenv</code>
<code>defun</code>	<code>docclass</code>	<code>endenv</code>	<code>importmod</code>
<code>importpkg</code>	<code>importves</code>	<code>startdoc</code>	<code>useenv</code>

Table 1: Keywords in Vesti

### 2.3 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 `#label`, 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_import	#ltx3_off	#ltx3_on	#makeatletter	#makeatother
#mathchardef	#mathmode	#nonstopmode	#picture	#showfont
#textmode				

Table 2: Builtins in Vesti

## 2.4 docclass keywords

Keyword docclass is an analogous of `\documentclass` in  $\text{\LaTeX}$ . If docclass keyword is in the main paragraph, it acts just a normal word. In other words, docclass keyword actives only in the preamble.

## 3 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 #lu:
15 local function read_all(path)
16     local f, err = io.open(path, "rb")
17     assert(f, ("cannot open %s: %s"):format(path, err))
18     local data = f:read("*a")
19     f:close()
20     return data
21 end
22 :lu#<readAll>
23
24 startdoc
25 \maketitle
26
27 \section{Introduction}
28
29 \section{Language Reference}
30 \subsection{Structure of Vesti File}
31 Vesti is similar as  $\text{\LaTeX}$ . Its structure consists with two parts: {\tt preamble} and
32 {\tt main}. Preamble is the place where  $\text{\LaTeX}$  documentclass, packages, and
33 several settings are located. Main body is where actual documentation is located.
34 Below figure is the simple Vesti documentation.
35
36 useenv figure [ht] {
37     \centering
38     useenv tikzpicture {
39         \path (0,0) node[draw, inner sep=5pt] {\vbox{
40             ###\hbox{\tt\obeyspaces {\color{purple}docclass} article (10pt)}
41             ###\hbox{\tt\obeyspaces {\color{purple}importves} \{}}
42             ###\hbox{\tt\obeyspaces geometry (a4paper, margin=2.2cm)}
43             ###\hbox{\tt\obeyspaces \}}

```

```

44     \hbox{\tt\obeyspaces {\color{purple}startdoc}}
45     \hbox{\tt\obeyspaces Hello, Vesti!}
46 };
47 }
48 \caption{Almost very simple Vesti documentation}
49 }
50 We will see later, but the very difference with \LaTeX is that Vesti has its
51 own keywords (keywords are colored with purple). It makes the code readable and
52 it is easier and faster to write the document. The keyword startdoc splits
53 the preamble and the main part of the documentation similar with
54 %
55 % Don't ask why I chose Q for catcode 0.
56 %\tt\catcode`Q=0 Qcatcode`\\=12 \beginQ{documentQ}} in \LaTeX.
57 However, Vesti does not have the analogous part of
58 %\tt\catcode`Q=0 Qcatcode`\\=12 \endQ{documentQ}},
59 because almost every \LaTeX document (99.999% I'm sure) does not have any code
60 below %\tt\catcode`Q=0 Qcatcode`\\=12 \endQ{documentQ}}.
61 For this reason, Vesti automatically ends document when EOF (End Of File) is
62 found.
63
64 \subsection{Keywords}
65 Followings are reserved as keywords.
66 useenv table [ht] {
67     \centering
68     #lu:
69     local content = read_all("../src/lexer/Token.zig")
70
71     -- Lua's built-in patterns don't support lookahead.
72     -- We capture both the keyword and the TokenType, then filter out 'deprecated'.
73     -- Pattern breakdown:
74     --   %.%{           => matches ".{"
75     --   %s*"([~"]+)" => a quoted string -> capture 1
76     --   %s*,%s*TokenType%.([%w_]+) => TokenType.<Name> -> capture 2
77     local pat = "%.%{s*"([~"]+)"%s*,%s*TokenType%.([%w_]+)"
78
79     local keywords = {}
80     for name, tok in content:gmatch(pat) do
81         if tok ~= "deprecated" then
82             keywords[#keywords + 1] = name
83         end
84     end
85
86     table.sort(keywords)
87
88     vesti.print([[\\begin{tabular}{cccc}]]
89
90     for i, kw in ipairs(keywords) do
91         local cell = string.format("{\\ttfamily %s}", kw)
92         if (i % 4) == 0 then
93             vesti.print(cell .. [[\\]])
94         else
95             vesti.print(cell .. "&")
96         end
97     end
98
99     vesti.print([[\\end{tabular}]])
100     :lu#[readAll]
101     \caption{Keywords in Vesti}
102 }
103

```

```

104 \subsection{Builtins}
105 Vesti also has its own builtin functions, which are prefixed with \#.
106 One might wonder what distinguishes builtins from keywords. In fact, from the
107 compiler's internal perspective, there is no real difference. However, in actual
108 language usage, constantly typing the prefix can be somewhat tedious, especially
109 for functions that are commonly used.
110
111 From the perspective of language design --particularly in Vesti-- it is sometimes
112 desirable to use names that cannot serve as keywords. For example, Vesti
113 provides a built-in function {\tt\#label}, which will be explained later. Since Vesti
114 is a typewriting-oriented language, the word \lq\lq label\rq\rq\ is often used in its
115 ordinary sense rather than in its special semantic meaning within the language.
116
117 Followings are reserved as builtin functions.
118
119 useenv table [ht] {
120     \centering
121     #lu:
122     local content = read_all("../src/lexer/Token.zig")
123
124     -- match .{ "here" }
125     local pat = "%.%.{%s*\"([^\"]+)\">%s*%}"
126
127     local builtins = {}
128     for name, tok in content:gmatch(pat) do
129         builtins[#builtins + 1] = name
130     end
131     table.sort(builtins)
132
133     vesti.print([[ \begin{tabular}{ccccc} ]])
134
135     for i, kw in ipairs(builtins) do
136         local cell = string.format("\#\verb@%s@", kw)
137         if (i % 5) == 0 then
138             vesti.print(cell .. [[\]])
139         else
140             vesti.print(cell .. "&")
141         end
142     end
143
144     vesti.print([[ \end{tabular} ]])
145     :lu#[readAll]
146     \caption{Builtins in Vesti}
147 }
148
149 \subsection{{\ttfamily docclass} keywords}
150 Keyword {\tt docclass} is an analogous of \verb|\documentclass| in \LaTeX.
151 If docclass keyword is in the main paragraph, it acts just a normal word.
152 In other words, docclass keyword actives only in the preamble.
153
154 \section{Source Code of This Document}
155 Below code was generated by inline lua.
156 useenv Verbatim [numbers=left, numbersep=5pt, frame=single] {
157 #lu:
158     local content = read_all("vesti_man.ves")
159     for line in content:gmatch("(^[\r\n]*)\r?\n?") do
160         vesti.print(line)
161     end
162 :lu#[readAll]
163 }

```

--