Vesti Transpiler User Manual

Sungbae Jeong

October 22, 2025

1 Introduction

2 Language Reference

2.1 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.

```
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 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.

2.2 Keywords

Followings are reserved as keywords.

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

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 $\LaTeX$. 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.

```
docclass article (10pt)
importpkg {
    geometry (a4paper, margin = 2.2cm),
    xcolor,
    tikz,
    fancyvrb,
}

\title{Vesti Transpiler User Manual}
\author{Sungbae Jeong}

importves (font.ves)

#lu:
local function read_all(path)
  local f, err = io.open(path, "rb")
  assert(f, ("cannot open %s: %s"):format(path, err))
  local data = f:read("*a")
  f:close()
  return data
end
:lu#<readAll>

startdoc
\maketitle

\section{Introduction}

\section{Language Reference}
\subsection{Structure of Vesti File}
Vesti is similar as \LaTeX. Its structure consists with two parts: {\tt preamble} and
{\tt 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.

useenv figure [ht] {
    \centering
    useenv tikzpicture {
        \path (0,0) node[draw, inner sep=5pt] {\vbox{
        %##\hbox{\tt\obeyspaces {\color{purple}docclass} article (10pt)}
        %##\hbox{\tt\obeyspaces {\color{purple}importves} \{}
        %##\hbox{\tt\obeyspaces     geometry (a4paper, margin=2.2cm)}
        %##\hbox{\tt\obeyspaces \}}
        %##\hbox{\tt\obeyspaces {\color{purple}startdoc}}
        %##\hbox{\tt\obeyspaces Hello, Vesti!}
        }};
    }
    \caption{Almost very simple Vesti documentation}
}
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
%
% Don't ask why I chose Q for catcode 0.
%##{\tt\catcode`Q=0 Qcatcode`\\=12 \beginQ{documentQ}} in \LaTeX.
However, Vesti does not have the analogous part of
%##{\tt\catcode`Q=0 Qcatcode`\\=12 \endQ{documentQ}},
because almost every \LaTeX\ document (99.999\% I'm sure) does not have any code
below %##{\tt\catcode`Q=0 Qcatcode`\\=12 \endQ{documentQ}}.
For this reason, Vesti automatically ends document when EOF (End Of File) is
found.

\subsection{Keywords}
Followings are reserved as keywords.
useenv table [ht] {
    \centering
    #lu:
    local content = read_all("../src/lexer/Token.zig")

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

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

    table.sort(keywords)

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

    for i, kw in ipairs(keywords) do
        local cell = string.format("{\\ttfamily %s}", kw)
        if (i % 4) == 0 then
            vesti.print(cell .. [[\\]])
        else
            vesti.print(cell .. "&")
        end
    end

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

```

```
\subsection{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 {\tt\#label}, which will be explained later. Since Vesti
is a typewriting-oriented language, the word \lq\lq label\rq\rq\ is often used in its
ordinary sense rather than in its special semantic meaning within the language.

Followings are reserved as builtin functions.

useenv table [ht] {
    \centering
    #lu:
    local content = read_all("../src/lexer/Token.zig")

    -- match .{ "here" }
    local pat = "%.%{%s*\"([^\"]+)\"%s*%}"

    local builtins = {}
    for name, tok in content:gmatch(pat) do
        builtins[#builtins + 1] = name
    end
    table.sort(builtins)

    vesti.print([[\begin{tabular}{ccccc}]])

    for i, kw in ipairs(builtins) do
        local cell = string.format("\\#\\verb@%s@", kw)
        if (i % 5) == 0 then
            vesti.print(cell .. [[\\]])
        else
            vesti.print(cell .. "&")
        end
    end

    vesti.print([[\end{tabular}]])
    :lu#[readAll]
    \caption{Builtins in Vesti}
}

\subsection{{\ttfamily docclass} keywords}
Keyword {\tt docclass} is an analogous of \verb|\documentclass| in \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.

\section{Source Code of This Document}
Below code was generated by inline lua.
useenv Verbatim [numbers=left, numbersep=5pt, frame=single] {
#lu:
    local content = read_all("vesti_man.ves")
    for line in content:gmatch("([^\r\n]*)\r?\n?") do
        vesti.print(line)
    end
:lu#[readAll]
}
```

164