

# Vesti Transpiler User Manual

Sungbae Jeong

November 6, 2025

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Structure of Vesti File</b>	<b>2</b>
<b>3</b>	<b>Keywords</b>	<b>2</b>
3.1	<code>docclass</code> keyword . . . . .	2
3.2	<code>importpkg</code> keywords . . . . .	3
3.3	<code>startdoc</code> keyword . . . . .	3
3.4	<code>useenv</code> keyword . . . . .	3
3.5	<code>begenv</code> keyword and <code>endenv</code> keyword . . . . .	4
<b>4</b>	<b>Builtins</b>	<b>4</b>
4.1	<code>#label</code> builtin . . . . .	5
4.2	<code>#eq</code> builtin . . . . .	5
4.3	<code>#mathchardef</code> builtin . . . . .	5
<b>5</b>	<b>Source Code of This Document</b>	<b>5</b>

## 1 Introduction

I used to create several documents using  $\text{\LaTeX}$  (or plain  $\text{\TeX}$  but  $\text{\TeX}$  is quite cumbersome to write—especially when working with very complex tables or inserting images). Its markdown-like syntax is also not comfortable to use. For example, here is a simple  $\text{\LaTeX}$  document:

```
1 % coprime is my custom class. See https://github.com/e0328eric/coprime.
2 \documentclass[tikz, geometry]{coprime}
3
4 \settitle{My First Document}{Sungbae Jeong}{}
5 \setgeometry{a4paper, margin = 2.5cm}
6
7 \begin{document}
8 \section{Foo}
9 Hello, World!
10 \begin{figure}[ht]
11   \centering
12   \begin{tikzpicture}
13     \draw (0,0) -- (1,1);
14   \end{tikzpicture}
15 \end{figure}
16
17 The code above is a figure using TikZ.
18
19 \end{document}
```

What annoys me most when using it is the ‘\begin’ and ‘\end’ blocks. Is there a way to write something much simpler? This question led me to start this project. Currently, the following code is generated into the L<sup>A</sup>T<sub>E</sub>X code above (except comments) using vesti:

```

1 % coprime is my custom class. See https://github.com/e0328eric/coprime.
2 docclass coprime (tikz, geometry)
3
4 \settitle{My First Document}{Sungbae Jeong}{}
5 \setgeometry{a4paper, margin = 2.5cm}
6
7 startdoc
8
9 \section{Foo}
10 Hello, World!
11 useenv figure [ht] {
12     \centering
13     useenv tikzpicture {
14         \draw (0,0) -- (1,1);
15     }
16 }
17
18 The code above is a figure using TikZ.

```

## 2 Structure of Vesti File

Vesti is similar as L<sup>A</sup>T<sub>E</sub>X. Its structure consists with two parts: preamble and main. Preamble is the place where L<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X. However, Vesti does not have the analogous part of \end{document}, because almost every L<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X. 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<sup>A</sup>T<sub>E</sub>X. 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 thier 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<sup>A</sup>T<sub>E</sub>X. 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 {}\br/>7                 %#\hbox{\tt\obeyspaces Hello, World!}\br/>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.

$$\text{useenv} \sqcup \langle\text{environment name}\rangle \sqcup \langle\text{argument}\rangle^* \sqcup \{ \langle\text{body}\rangle \}$$

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<sup>A</sup>T<sub>E</sub>X, 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 `#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.

<code>#chardef</code>	<code>#enum</code>	<code>#eq</code>	<code>#get_filepath</code>	<code>#label</code>
<code>#ltx3_off</code>	<code>#ltx3_on</code>	<code>#makeatletter</code>	<code>#makeatother</code>	<code>#mathchardef</code>
<code>#mathmode</code>	<code>#noltx3</code>	<code>#picture</code>	<code>#showfont</code>	<code>#textmode</code>
<code>#xparse</code>				

Table 2: Builtins in Vesti

Since `fancyvrb` does not allow verbatim-like commands inside of `Verbatim` environment, builtins are not colored in example codes.

## 4.1 #label builtin

## 4.2 #eq builtin

#eq is a abbreviation of ‘useenv equation’. Actually, below codes are same.

```
1 #label(eq:1)
2 useenv equation {
3     \sum_{n=1}^{\infty} {1/n^2} = {\pi^2/6}.
4 }
```

```
1 #eq(eq:1) {
2     \sum_{n=1}^{\infty} {1/n^2} = {\pi^2/6}.
3 }
```

Since Vesti is used in the mathematical documents mainly, small syntactic suger is needed, so Vesti introduce #eq builtin.

## 4.3 #mathchardef builtin

#mathchardef defines a math token using the unicode codepoint. Here is the grammar of #mathchardef builtin.

#mathchardef  $\langle kind \rangle \langle font-num \rangle \langle unicode-codepoint \rangle \langle function \rangle$

Before explaining each parameter, introduce some examples.

- #mathchardef .opening 0 29d8 \lfoo
- #mathchardef .ordinary 0 2202 \diff

# 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{blue!65!yellow}\#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 % space character used in the textbook.
33 #makeatletter
34 #chardef 2423 \@b
35 defun ob {\kern2pt{\tt\@b}\kern2pt}
36 #makeatother
37
38 startdoc
39 \maketitle
40 \tableofcontents
41
42 \section{Introduction}
43 I used to create several documents using \LaTeX (or plain \TeX but \TeX is
44 quite cumbersome to write—especially when working with very complex tables or
45 inserting images). Its markdown-like syntax is also not comfortable to use. For
46 example, here is a simple \LaTeX document:
47
48 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
49 %% coprime is my custom class. See https://github.com/e0328eric/coprime.
50 %%\documentclass[tikz, geometry]{coprime}
51 %%
52 %%\settitle{My First Document}{Sungbae Jeong}{}
53 %%\setgeometry{a4paper, margin = 2.5cm}
54 %%
55 %%\begin{document}
56 %%\section{Foo}
57 %%Hello, World!
58 %%\begin{figure}[ht]
59 %%    \centering
60 %%    \begin{tikzpicture}
61 %%        \draw (0,0) -- (1,1);
62 %%    \end{tikzpicture}
63 %%\end{figure}
64 %%
65 %%The code above is a figure using TikZ.
66 %%
67 %%\end{document}
68 }

69
70 What annoys me most when using it is the \lq\verb@\begin@\rq\ and
71 \lq\verb@\end@\rq\ blocks. Is there a way to write something much simpler? This
72 question led me to start this project. Currently, the following code is
73 generated into the \LaTeX code above (except comments) using vesti:
74
75 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
76 %% coprime is my custom class. See https://github.com/e0328eric/coprime.
77 %%+keyword|docclass@ coprime (tikz, geometry)
78 %%
79 %%\settitle{My First Document}{Sungbae Jeong}{}
80 %%\setgeometry{a4paper, margin = 2.5cm}
81 %%
82 %%+keyword|startdoc@
83 %%
84 %%\section{Foo}
85 %%Hello, World!
86 %%+keyword|useenv@ figure [ht] {

```

```

87  %#     \centering
88  %#     +keyword|useenv@ tikzpicture {
89  %#         \draw (0,0) -- (1,1);
90  %#     }
91  %#}
92  %#The code above is a figure using TikZ.
93 }
94
95
96 \section{Structure of Vesti File}
97 Vesti is similar as \LaTeX. Its structure consists with two parts: {\tt preamble} and
98 {\tt main}. Preamble is the place where \LaTeX\ documentclass, packages, and
99 several settings are located. Main body is where actual documentation is located.
100 Below figure is the simple Vesti documentation.
101
102 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
103 %#+keyword|docclass@ article (10pt)
104 %#+keyword|importpkg@ {
105 %#     geometry (a4paper, margin=2.2cm)
106 %#}
107 %#+keyword|startdoc@
108 %#Hello, Vesti!
109 }
110
111 We will see later, but the very difference with \LaTeX\ is that Vesti has its
112 own keywords (keywords are colored with purple). It makes the code readable and
113 it is easier and faster to write the document. The keyword startdoc splits
114 the preamble and the main part of the documentation similar with
115
116 % Don't ask why I chose Q for catcode 0.
117 %#{\tt\catcode`Q=0 Q\catcode`\\=12 \begin{documentQ}} in \LaTeX.
118 However, Vesti does not have the analogous part of
119 %#{\tt\catcode`Q=0 Q\catcode`\\=12 \end{documentQ}},
120 because almost every \LaTeX\ document (99.999\% I'm sure) does not have any code
121 below %#{\tt\catcode`Q=0 Q\catcode`\\=12 \end{documentQ}}.
122 For this reason, Vesti automatically ends document when EOF (End Of File) is
123 found.
124
125 \section{Keywords}
126 Followings are reserved as keywords. In this document, every Vesti keyword has
127 the form like \keyword{this}.
128 useenv table [ht] {
129     \centering
130     #lu:
131     local content = read_all("../src/lexer/Token.zig")
132
133     -- Lua's built-in patterns don't support lookahead.
134     -- We capture both the keyword and the TokenType, then filter out 'deprecated'.
135     -- Pattern breakdown:
136     -- %.%{           => matches ".{"
137     -- %s*"([""]+)" => a quoted string -> capture 1
138     -- %s*,%s*TokenType%.([%w_]++) => TokenType.<Name> -> capture 2
139     local pat = "%.%{%" s*"([""]+)"%" s*,%" s*TokenType%.([%w_]++)"
140
141     local keywords = {}
142     for name, tok in content:gmatch(pat) do
143         if tok ~= "deprecated" then
144             keywords[#keywords + 1] = name
145         end
146     end

```

```

147     table.sort(keywords)
148
149     vesti.print([[\begin{tabular}{cccc}]])
150
151     for i, kw in ipairs(keywords) do
152         local cell = string.format("\\\ keyword{\%s}", kw)
153         if (i % 4) == 0 then
154             vesti.print(cell .. [[\\]])
155         else
156             vesti.print(cell .. "&")
157         end
158     end
159
160     vesti.print([[\end{tabular}]])
161 :lu#[readAll]
162 \caption{Keywords in Vesti}
163 }
164
165 \subsection{\keyword{docclass} keyword}
166 Keyword \keyword{docclass} is an analogous of \verb|\documentclass| in \LaTeX.
167 If \keyword{docclass} is in the main paragraph, it acts just a normal word.
168 In other words, \keyword{docclass} actives only in the preamble.
169 The syntax of \keyword{docclass} is following:
170
171 useenv center {
172     \keyword{docclass}\ob<class name>\ob{\tt{}<arguments>{\tt{}}
173 }
174 Here, arguments are separated by commas and embraced by {\tt ()}. Here are some
175 examples.
176
177 \goodbreak
178 useenv itemize {
179     \item \keyword{docclass} {\tt article}
180     \item \keyword{docclass} {\tt article (10pt)}
181     \item \keyword{docclass} {\tt article (10pt, twocols)}
182     \item \keyword{docclass} {\tt article (10pt,twocols)}
183 }
184
185 \subsection{\keyword{importpkg} keywords}
186 Keyword \keyword{importpkg} is an analogous of \verb|\usepackage| in \LaTeX.
187 If \keyword{importpkg} is in the main paragraph, it acts just a normal word.
188 In other words, \keyword{importpkg} actives only in the preamble.
189
190 importpkg has two different syntax. First one is same as docclass.
191 useenv center {
192     \keyword{importpkg}\ob<pkg-name>\ob{\tt{}<arguments>{\tt{}}
193 }
194 Here, arguments are separated by commas and embraced by {\tt ()}.
195 In the practical case, one should include several packages with options.
196 importpkg also supports such case. We will look at an example instead of
197 giving rigorous grammar.
198 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
199 %#+keyword|importpkg@ {
200 %#     amsmath, amssymb, amsthm,
201 %#     geometry (a4paper, margin=2.2cm),
202 %#}
203 }
204
205 \noindent As one can see, inside of \verb|{|}|, several packages can be used

```

```

207 together with thier options.
208
209 \subsection{\keyword{startdoc} keyword}
210 Keyword \keyword{startdoc} tells to Vesti that the main document starts. In the
211 main document, you can also write \keyword{startdoc} in the main document. In
212 that case, \keyword{startdoc} does nothing.
213
214 \subsection{\useenv keyword}
215 As the name implies, keyword \useenv is an analogous of \verb|\begin{...}| and
216 \verb|\end{...}| pair in \LaTeX.
217 The simplest \useenv is like this.
218
219 \useenv figure [ht] {
220     \centering
221     \useenv tikzpicture {
222         \useenv scope {
223             \path (0,0) node {\vbox{
224                 %#\hbox{\tt\useenv center \{}%
225                 %#\hbox{\tt\obeyspaces Hello, World!}%
226                 %#\hbox{\tt\obeyspaces\}}
227             }};
228         }
229         \path (2.3,0) node {or};
230         \useenv scope [shift={(6,0)}]{
231             \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
232         }
233     }
234 }
235
236 As you can see, {\tt\useenv center} is the part of \verb|\begin{center}|, and
237 the single {\tt\{} is the part of \verb|\end{center}|. Since Vesti knows their
238 pair, one can write a code with several environment, and each pair is properly
239 matched. For instance, above example is written in Vesti like follows. Here,
240 \verb|\useenv| just prints \useenv in that style.
241
242 \useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
243 %#|+color|purple@useenv@ figure [ht] {
244 %#     \centering
245 %#     |+color|purple@useenv@ tikzpicture {
246 %#         |+color|purple@useenv@ scope {
247 %#             \path (0,0) node {\vbox{
248 %#                 |+color|blue@%#@\hbox{\tt\useenv center \{}%
249 %#                 |+color|blue@%#@\hbox{\tt\obeyspaces Hello, World!}%
250 %#                 |+color|blue@%#@\hbox{\tt\obeyspaces\}}
251 %#             }};
252 %#         }
253 %#         \path (2.3,0) node {or};
254 %#         |+color|purple@useenv@ scope [shift={(6,0)}] {
255 %#             \path (0,0) node {\tt\useenv center \{ Hello, World! \}};
256 %#         }
257 %#     }
258 %# }
259 }
260
261 Full syntax about \useenv is the following.
262 \useenv center {
263     \useenv\ob<environment name>\ob<argument>*\ob{\tt\{} <body> {\tt\}}
264 }
265 where '*' means that the number of <argument> is zero or at least one, and
266 $$
```

```

267  "<argument>" = useenv cases {
268      "(<argument>)" & "mandatory arguments" \\
269      "[<argument>]" & "optional arguments"
270  }
271  $$  

272  For instance, below one is a valid Vesti code (environment {\tt foo} is  

273  undefined in general). As one can see, spaces can exist in between <argument>s.  

274  useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {  

275      %#|+color|purple@useenv@ foo (asd)(fff)[\ames and \awdsa] (askws) [\rrsaa] {  

276          %#    foobar  

277      %#}  

278  }  

279  

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

281  As the name implies, both keywords \begenv and \endenv are analogous of  

282  \verb|\begin{...}| and \verb|\end{...}| pair in \LaTeX, respectively.  

283  Thus below code  

284  useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {  

285      %#|+color|purple@begenv@ center  

286      %#    asdsad  

287      %#|+color|purple@endenv@ center  

288  }  

289  is exactly same as  

290  useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {  

291      %#|+color|purple@useenv@ center {  

292          %#    asdsad  

293      %#}  

294  }  

295  Then why we need \begenv and \endenv if we already have \useenv*?  

296  

297  \section{Builtins}  

298  Vesti also has its own builtin functions, which are prefixed with \#.  

299  One might wonder what distinguishes builtins from keywords. In fact, from the  

300  compiler's internal perspective, there is no real difference. However, in actual  

301  language usage, constantly typing the prefix can be somewhat tedious, especially  

302  for functions that are commonly used.  

303  

304  From the perspective of language design --particularly in Vesti-- it is  

305  sometimes desirable to use names that cannot serve as keywords. For example,  

306  Vesti provides a built-in function {\tt\#label}, which will be explained later.  

307  Since Vesti is a typewriting-oriented language, the word \lq\lq label\rq\rq\ is  

308  often used in its ordinary sense rather than in its special semantic meaning  

309  within the language.  

310  

311  Followings are reserved as builtin functions.  

312  

313  useenv table [ht] {
314      \centering
315      #lu:
316      local content = read_all("../src/lexer/Token.zig")
317
318      -- match .{ "here" }
319      local pat = "%.%{%"s*\"([^\"]+)\\"%s*%}""
320
321      local builtins = {}
322      for name, tok in content:gmatch(pat) do
323          builtins[#builtins + 1] = name
324      end
325      table.sort(builtins)
326

```

```

327 vesti.print([[\begin{tabular}{ccccc}]])
328
329 for i, kw in ipairs(builtins) do
330     local cell = string.format("\\"builtin@%s@", kw)
331     if (i % 5) == 0 then
332         vesti.print(cell .. [[\\]])
333     else
334         vesti.print(cell .. "&")
335     end
336 end
337
338 vesti.print([[\end{tabular}]])
339 :lu#[readAll]
340 \caption{Builtins in Vesti}
341 }
342 Since {\tt fancyvrb} does not allow verbatim-like commands inside of
343 {\tt Verbatim} environment, builtins are not colored in example codes.
344
345 \subsection{\builtin|label| builtin}
346
347 \subsection{\builtin|eq| builtin}
348 \builtin|eq| is a abbreviation of \lq\keyword{-useenv-} equation\rq.
349 Actually, below codes are same.
350
351 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
352 %%label(eq:1)
353 %#+keyword|useenv@ equation {
354 %#   \sum_{n=1}^{\infty} {1/n^2} = {\pi^2/6}.
355 %#}
356 }
357 useenv Verbatim [numbers=left, numbersep=5pt, frame=single, commandchars=+|@] {
358 %%eq(eq:1) {
359 %#   \sum_{n=1}^{\infty} {1/n^2} = {\pi^2/6}.
360 %#}
361 }
362 Since Vesti is used in the mathematical documents mainly, small syntactic suger
363 is needed, so Vesti introduce \builtin|eq| builtin.
364
365 \subsection{\builtin|mathchardef| builtin}
366 \builtin|mathchardef| defines a math token using the unicode codepoint.
367 Here is the grammar of \builtin|mathchardef| builtin.
368 useenv center {
369     \builtin|mathchardef|\ob<kind>\ob<font-num>\ob<unicode-codepoint>\ob<function>
370 }
371 Before explaining each parameter, introduce some examples.
372 useenv itemize {
373     \item \builtin|mathchardef|\ob.opening\ob0\ob29d8\ob\verb|\lfoo|
374     \item \builtin|mathchardef|\ob.ordinary\ob0\ob2202\ob\verb|\diff|
375 }
376
377 \section{Source Code of This Document}
378 Below code was generated by inline lua.
379 useenv Verbatim [numbers=left, numbersep=5pt, frame=single] {
380 #lu:
381     local content = read_all("vesti_man.ves")
382     for line in content:gmatch("(^\r\n*)\r?\n?") do
383         vesti.print(line)
384     end
385 :lu#[readAll]
386 }

```

