LaTeX

Hajun Park

June 29, 2024

Contents

Chapter 1

Code

1.1 verbatim

1.1.1 Inline code

`\verb|<text>|` (“`|`” can be replaced by any character except “`*`”)

```
\verb|Hello, world!|
```

```
Hello, world!
```

1.1.2 Code block

`\begin{verbatim} ... \end{verbatim}`

```
\begin{verbatim}
def hello():
    print("Hello, world!")
\end{verbatim}
```

```
def hello():
    print("Hello, world!")
```

1.2 listings

`\usepackage{listings}`

1.2.1 Inline code

`\lstinline!<text>!` (“`|`” can be replaced by any character)

```
\lstinline|Hello, world!|
```

Hello, world!

1.2.2 Code block

`\begin{lstlisting} ... \end{lstlisting}`

```
\begin{lstlisting}
def hello():
    print("Hello, world!")
\end{lstlisting}
```

```
def hello():
    print("Hello, world!")
```

1.2.3 Input file

`\lstinputlisting{<file-path>}`

```
\lstinputlisting{hello.py}
```

```
def hello():
    print("Hello, world!")
```

1.3 minted

`\usepackage{listings}`

Minted uses Pygments for syntax highlighting.

Install Python and then Pygments.

```
$ pip install Pygments
```

To use Pygments on LaTeX, you need to pass `-shell-escape` flag to LaTeX.

```
$ lualatex -shell-escape <file>
```

If you want to compile LaTeXdocument containing minted with Visual Studio Code and LaTeX Workshop Plugin, add the following to `settings.json`.

```
{
  "latex-workshop.latex.tools": [
    {
      "name": "lualatex",
      "command": "lualatex",
      "args": [
        "-shell-escape",
        "-synctex=1",
        "-interaction=nonstopmode",
        "-file-line-error",
        "%DOC%"
      ],
      "env": {}
    },
    {
      "name": "bibtex",
      "command": "bibtex",
      "args": [
        "%DOCFILE%"
      ],
      "env": {}
    }
  ],
  "latex-workshop.latex.recipes": [
    {
      "name": "lualatex",
      "tools": [
        "lualatex"
      ]
    },
    {
      "name": "lualatex -> bibtex -> lualatex * 2",
      "tools": [
        "lualatex",
        "bibtex",
        "lualatex",
        "lualatex"
      ]
    }
  ]
}
```

1.3.1 Inline code

`\mintinline{<language>}{<text>}`

1.3.2 Code block

For single line: `\mint{<language>}{<text>}`

```
\mint{python}{
print("Hello, world!")
}
```

```
print("Hello, world!")
```

For multiple lines: `\begin{minted} ... \end{minted}`

```
\begin{minted}{python}
def hello():
    print("Hello, world!")
\end{minted}
```

```python
def hello():
    print("Hello, world!")
```

1.3.3 Input file

`\inputminted{<language>}{<file-path>}`

```
\inputminted{python}{hello.py}
```

```python
def hello():
    print("Hello, world!")
```

1.3.4 Captions and labels

Minted provides floating listing environment to use with caption and label.

```
\begin{listing}[H]
  \mint{python}|print("Hello,
↪   world!")|
  \caption{Code example}
  \label{lst:example}
\end{listing}
```

```python
print("Hello, world!")
```

Listing 1: Code example

1.3.5 Options

Setting global minted options

inline & code blocks

```
\setminted{<options>}
\setminted[<language>]{<options>}
```

inline

```
\setmintedinline{<options>}
\setmintedinline[<language]{<options>}
```

Defining shortcuts

minted environment

```
\newminted{<language>}{<options>} % default environment-name:
↪   <language>code
\newminted[<environment-name>]{<language>}{<options>}
```

```

\begin{<environment-name>}
\end{<environment-name>}
```

mint command

```
\newmint{<language>}{<options>} % default macro-name: <language>
\newmint[<macro-name>]{<language>}{<options>}

\<macro-name>/<text>/ % ``/'' can be replaces by any character
```

mintinline command

```
\newmintinline{<language>}{<options>} % default macro-name:
↪   <language>inline
\newmintinline[<macro-name>]{<language>}{<options>}

\<macro-name>/<text>/ % ``/'' can be replaces by any character
```

inputminted command

```
\newmintedfile{<language>}{<options>} % default macro-name:
↪   <language>file
\newmintedfile[<macro-name>]{<language>}{<options>}

\<macro-name>{<file-path>}
```

Available options

- autogobble (boolean): Remove gobble (leading whitespace)
- breaklines (boolean): Automatically break long lines
- frame (none | leftline | topline | bottomline | lines | single): Put lines around the code
- linenos (boolean): Linen numbers
- numbersep (dimension): Gap between numbers and start of line

```
\setminted{
  autogobble,
  breaklines,
  frame=single,
  linenos,
  numbersep=2mm,
}
```