codeanatomy – Draw Code Anatomy*

Usage with listings

Hồng-Phúc Bùi†

Released 2023/01/24

Contents

1 General Usage with Package listings

1.1 Setup Package listings ...................................... 1
1.2 Typeset Code .............................................. 2
1.3 Mark Code ................................................ 2
1.4 Highlight some tokens ..................................... 3
1.5 Add Annotations to Listing .............................. 4

2 Some examples ............................................. 4

2.1 Anatomy of a Java Program [1, p. 5] ................. 4
2.2 Anatomy of an expression [1, p. 17] .................... 5
2.3 Using a primitive Data Type [1, p. 17] ............... 6
2.4 Anatomy of a method signature [1, p. 30] ......... 7
2.5 Using a library method [1, p. 30] ...................... 7
2.6 Anatomy of an if statement [1, p. 51] .............. 8
2.7 Anatomy of a while loop [1, p. 54] .................. 9
2.8 Anatomy of a for loop [1, p. 59] ..................... 10
2.9 Anatomy of a static method [1, p. 196] ........... 10

1 General Usage with Package listings

1.1 Setup Package listings

The most important setup for the package listings is the delimiter to escape \LaTeX commands in Listing. With this escape delimiter we can mark a piece of code as with \texttt{\Part}. In this example we use ! and ! as delimiter. Code between ! and ! is evaluated as \LaTeX-code.

∗This file describes v0.4-Beta, last revised 2023/01/24.
†E-mail: hong-phuc.bui (at) htwsaar dot de
Delimiter can also be reset in \texttt{document}-Environment, typical just before a new \texttt{begin{lstlisting}} environment so each anatomy can have different delimiter. The fact is, in this document I use + and + for the above listing, so that I can typeset \texttt{!} and \texttt{!} in this listing.

You may also want to set option \texttt{keepspaces} to \texttt{true}, so that your reader can easily copy past your example code.

1.2 Typeset Code

The command \texttt{codeBlock} does not work if the environment \texttt{lstlisting} is passed to its argument. So instead of \texttt{codeBlock} we must use the Ti\textit{k}Z command \texttt{node}: 

\begin{tikzpicture}[remember picture]
  \node[code] at (0,0) {
    \begin{lstlisting}
      function gcd(p,q) {
        if (q === 0) {
          return q;
        }else{
          let r = p \% q;
          return gcd(q, r);
        }
      }
    \end{lstlisting}
  };
\end{tikzpicture}

Figure \ref{figure-1} shows result of the above code.

1.3 Mark Code

The command \texttt{cPart} can be used to mark single-line code parts. For multiline code parts one can use \texttt{xxxPoint} family to mark the outer most points of code parts and \texttt{fitExtrem} to cover extern points of a code part. These commands must be put between escape delimiter, here \texttt{!} and \texttt{!}.
function gcd(p, q) {
    if (q === 0) {
        return q;
    } else {
        let r = p % q;
        return gcd(q, r);
    }
}

Figure 1: Code Listing is formatted

\begin{tikzpicture}[remember picture]
    \node(code) [anatomy] at (0,0) {
        \begin{lstlisting}
        function gcd(p, q) {
            if (q === 0) {
                return q;
            } else {
                let r = p % q;
                return gcd(q, r);
            }
        }
        \end{lstlisting}
    };
\fitExtrem{fnBody}{(mostLeft) (mostRight) (mostBottom)}
\end{tikzpicture}

Figure 2 shows the result of the above code.

\begin{tikzpicture}[remember picture]
    \tikzstyle{token} = [code part, fill=yellow]
    \node(code) [anatomy] at (0,0) {
        function gcd(p, q) {
            if (q === 0) {
                return q;
            } else {
                let r = p % q;
                return gcd(q, r);
            }
        }
    };
\fitExtrem{fnBody}{(mostLeft) (mostRight) (mostBottom)}
\end{tikzpicture}

Figure 2: Code Listing with mark of code parts

1.4 Highlight some tokens

\lstset{escapeinside=!{!},basicstyle=\linespread{1.8}}
\begin{tikzpicture}[remember picture]
\tikzstyle{token} = [code part, fill=yellow]
\end{tikzpicture}
function \gcd(p,q) {  
if \(q === 0\) {  
    return \gcd(q);  
} else {  
    let r = p \% q;  
    return \gcd(q, r);  
}  
}

1.5 Add Annotations to Listing

This step is the same as the description in the usage document of package \texttt{codeanatomy}. Readers can typeset annotations to the above listing like an exercise.

2 Some examples

Most of examples in this section are redrawn from the textbook [1].

2.1 Anatomy of a Java Program [1] p. 5

\begin{tikzpicture}[remember picture]
\node[code] at (0,0) {Public
\begin{lstlisting}
public class HelloWorld {
    public static void main(String[] argv) {
        \textbf{mainLeft}!
    
\end{lstlisting}}
\end{tikzpicture}
2.2 Anatomy of an expression [1, p. 17]
2.3 Using a primitive Data Type [1, p. 17]

\lstset{escapeinside=(!}{!})
\begin{tikzpicture}
\remember picture
\code annotation/.append style = { % customize style of annotation text
    font=\sffamily\footnotesize
}
\end{tikzpicture}

%%% Annotations
\codeAnnotation{declareText} ( 1,2.75 ) {declaration statement}
\codeAnnotation{literalText} ( 2.5,1.45) {literal}
\codeAnnotation{varText} (-1.5,1.75 ) {variable name}
\codeAnnotation{assignText} (-1.5,0.75 ) {assignment\statement}
\codeAnnotation{initText} (-1.5,-0.75) {inline initialization\statement}

%%% Arrows
\draw[->,annotation] (declareText) -- (d);
\draw[->,annotation] (literalText) -- (l);
\draw[->,annotation] (varText.south east) -- (v);
\draw[->,annotation] (assignText) -- (a);
\draw[->,annotation] (initText) -- (i.south west);
\end{tikzpicture}
2.4 Anatomy of a method signature [1, p. 30]

\begin{tikzpicture}[remember picture]
\node(code) [anatomy] at (0,0) {
\begin{lstlisting}
public class Math
....
static double sqrt(double a)
....
\end{lstlisting}
};
% Annotation
\codeAnnotation{lText} (3,2.5) {library name}
\codeAnnotation{sText} (-1,1) {signature}
\codeAnnotation{nText} (4.5,1.5) {method name}
\codeAnnotation{rText} (2.0,-0.51) {return type}
\codeAnnotation{aText} (4.5,-0.51) {argument type}
% Arrows
\draw[->, annotation] (lText) -- (l);
\draw[->, annotation] (nText) -- (n);
\draw[->, annotation] (rText) -- (r);
\draw[->, annotation] (aText) -- (a);
\end{tikzpicture}

2.5 Using a library method [1, p. 30]

\begin{tikzpicture}[remember picture]
2.6 Anatomy of an if statement [1] p. 51

```latex
\begin{lstlisting}
if (!e) {
    int t = x;
    x = y;
    y = t;
}
\end{lstlisting}
```

\begin{tikzpicture}
\node[code] at (0,0) {\begin{lstlisting}
if (!e) {
    int t = x;
    x = y;
    y = t;
}
\end{lstlisting}};
\end{tikzpicture}
2.7 Anatomy of a while loop [1, p. 54]
2.8 Anatomy of a for loop [1, p. 59]

```java
int power = 1;
for ( int i = 0; i <= n; i++ )
{
    System.out.println(i + " " + power);
    power *= 2;
}
```

2.9 Anatomy of a static method [1, p. 196]
public static double harmonic(int n)
{
    double sum = 0.0;
    for (int i = 0; i <= n; ++i)
    {
        sum += 1.0/i;
    }
    return sum;
}
public static double harmonic(int n) {
    double sum = 0.0;
    for (int i = 0; i <= n; ++i) {
        sum += 1.0/i;
    }
    return sum;
}

References