
codeanatomy – Draw Code Anatomy

Usage with listings

Hồng-Phúc Bùi †

Released 2019/07/12

Contents

1 General Usage in Conjunction with Package listings 1
  1.1 Setup Package listings .......................... 1
  1.2 Typeset Code ................................. 2
  1.3 Mark Code .................................. 3
  1.4 Add Annotations to Listing .................. 3

2 Some examples 3
  2.1 Anatomy of a Java Program [1, p. 5] .............. 3
  2.2 Anatomy of an expression [1, p. 17] .............. 4
  2.3 Using a primitive Data Type [1, p. 17] ............ 5
  2.4 Anatomy of a method signature [1, p. 30] ........ 6
  2.5 Using a library method [1, p. 30] .................. 6
  2.6 Anatomy of an if statement [1, p. 51] ............. 7
  2.7 Anatomy of a while loop [1, p. 54] ............... 7
  2.8 Anatomy of a for loop [1, p. 59] .................. 8
  2.9 Anatomy of a static method [1, p. 196] .......... 9

1 General Usage in Conjunction 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 \cPart. In this example we use | and | as delimiter. Code between | and | is evaluated as \LaTeX-code.

---

*This file describes v0.4-Alpha, last revised 2019/07/12.
†E-mail: hong-phuc.bui (at) htwasaar dot de
Setup

Delimiter can also be reset in document-Environment, typical just before a new \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 + in this listing.

1.2 Typeset Code

The command \codeBlock does not work if the environment \lstlisting is passed to its argument. So instead of \codeBlock we must use the Ti\kZ command \node:

\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}
};
\end{tikzpicture}

Figure 1 shows result of the above code.

\begin{lstlisting}
function gcd(p,q) {
  if (q === 0) {
    return q;
  }else{
    let r = p \% q;
    return gcd(q, r);
  }
}\end{lstlisting}

Figure 1: Code Listing is formatted
1.3 Mark Code

The command \texttt{\cPart} can be used to mark single-line code parts. For multiple-line code parts once can use \texttt{\extremPoint} 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 in delimiter, here ! and !.

\begin{tikzpicture}
\node[code] at (0,0) {
\begin{lstlisting}
!\cPart{fnHead}{function \cPart{fnName}{gcd}\cPart{paramList}{(p,q)}}!
  if (q === 0) {
    return q;
  }else{
    let r = p % q;
    return gcd(q, r);\extremPoint{mostRight}!
  }\mbPoint{mostBottom}!
}\end{lstlisting}
};\fitExtrem{fnBody}{(mostLeft) (mostRight) (mostBottom)}\end{tikzpicture}

Figure 2 shows the result of the above code.

1.4 Add Annotations to Listing

This step is the same as the description in the main 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]
public class HelloWorld
{
    public static void main(String[] argv)
    {
        System.out.print("Hello World");
    }
}

2.2 Anatomy of an expression [1, p. 17]
2.3 Using a primitive Data Type [1, p. 17]
2.4 Anatomy of a method signature [1, p. 30]

```
\begin{tikzpicture}
\node[code] at (0,0) {
\begin{lstlisting}
public class !\iPart{l}{Math}!
....
!\cPart{s}{\bgcode{static} \iPart{r}{double} \iPart{n}{sqrt}(%\iPart{a}{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] (sText) -- (s);
\draw[->, annotation] (rText) -- (r);
\draw[->, annotation] (aText) -- (a);
\end{tikzpicture}
```

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

```
\begin{tikzpicture}
\codeBlock{
\iPart{r}{double} d = \iPart{l}{Math}.\iPart{m}{sqrt}(\iPart{a}{b*b - 4.0*a*c});
}
% Annotation
\codeAnnotation{lText} (2, 1.125) {library name}
\end{tikzpicture}
```
2.6 Anatomy of an if statement [1, p. 51]

```latex
\begin{tikzpicture}[remember picture]
\node[code] at (0,0) {
\begin{lstlisting}
if (!tPart(e)(x > y))
{
  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]
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;
}