Cascading Style Sheets Code Count™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
6/10/2016	1.0	Original Release	Derek Lengenfelder
Date	Version No.	Click here to Description.	Name

Table of Contents

1. Definitions

1.1. **SLOC –** Source Lines of Code is a unit used to measure the size of software program. SLOC counts the program source code based on a certain set of rules. SLOC is a key input for estimating project effort and is also used to calculate productivity and other measurements.

1.2. **Physical SLOC –** One physical SLOC is corresponding to one line starting with the first character and ending by a carriage return or an end-of-file marker of the same line, and which excludes the blank and comment line.

1.3. **Logical SLOC –** Lines of code intended to measure "statements", which normally terminate by a semicolon (C/C++, Java, C#) or a carriage return (VB, Assembly), etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

1.4. **Data declaration line or data line –** All executable lines are contained within a class. Each class begins and ends with a curly brace. The following example shows how a class can be defined.

```
<tag name> {
                    …..
                   }
```

Irrespective of how many executable lines are contained within the curly braces, the number of data declaration lines for the above example is 1. Therefore a data declaration line for CSS corresponds to those lines that define a class.

1.5. **Compiler directive –** A statement that tells the compiler how to compile a program, but not what to compile.

1.6. **Blank Line –** A physical line of code, which contains any number of white space characters (spaces, tabs, form feed, carriage return, line feed, or their derivatives).

1.7. **Comment Line –** A comment is defined as a string of zero or more characters that follow language-specific comment delimiter.

CSS comment delimiters are "/*" and end with "/*". A whole comment line may span one line and does not contain any compliable source code. An embedded comment can co-exist with compliable source code on the same physical line. Banners and empty comments are treated as types of comments.

1.8. **Executable Line of code –** A line that contains software instruction executed during runtime and on which a breakpoint can be set in a debugging tool. These include the lines contained within the definition of a CSS class such as p, td etc or even user defined classes. Every executable line needs to end with a semicolon barring the last line before the container class is closed (using a curly brace). An instruction can be stated in a simple or compound form.

An executable line of code may contain the following program control statements:

- XML does not have any executable lines of code.

An executable line of code may not contain the following statements:

- Data declaration (data) lines
- Whole line comments, including empty comments and banners
- Blank lines

2. Checklist for source statement counts

PHYSICAL SLOC COUNTING RULES			
MEASUREMENT UNIT	ORDER OF PRECEDENCE	PHYSICAL SLOC	COMMENTS
Executable lines	1	One per line	Defined in 1.8
Non-executable lines			
Declaration (Data) lines	2	One per line	Defined in 1.4
Compiler directives	3	One per line	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not included (NI)	
Embedded	5	NI	
Banners	6	NI	
Empty comments	7	NI	
Blank lines	8	NI	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	<tagname> { … }	1	Count once.	Each tag is counted once. It can be an HTML tag or a user defined class name.
R02	…{<keyword>: <value>; … }	2	Count number of semicolon separated statements.	Braces {…} and semicolon ; used with this statement are not counted. The last statement need not end with a semicolon.

3. Examples

DECLARATION OR DATA LINES

DDL1 – block declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<header name> {	para1 {	1
<keyword>: <value>;	text-align: center;	1
<keyword>: <value>;	color: red;	1
}	}	0

DDL2 – empty block

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<header name> {	para1 {	1
}	}	0

DDL3 – multiple semicolons, last line semicolon not present

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<header name> {	para1 {	1
<keyword>: <value>;;;;;	text-align: center;;;;;;	1
<keyword>: <value>	color: red	1
}	}	0

4. Complexity

Complexity measures the occurrences of different keywords in code baseline. CSS does not have complexity. Therefore, its complexity is always listed as zero (0).

5. Cyclomatic Complexity

Cyclomatic complexity measures the number of linearly independent paths through a program. It is measured for each function, procedure, or method according to each specific program language. This metric indicates the risk of program complexity and also determines the number of independent test required to verify program coverage.

The cyclomatic complexity is computed by counting the number of decisions plus one for the linear path. Decisions are determined by the number of conditional statements in a function. A function without any decisions would have a cyclomatic complexity of one. Each decision such as an if condition or a for loop adds one to the cyclomatic complexity.

The cyclomatic complexity metric v (G) was defined by Thomas McCabe. Several variations are commonly used but are not included in the UCC. The modified cyclomatic complexity counts select blocks as a single decision rather than counting each case. The strict or extended cyclomatic complexity includes boolean operators within conditional statements as additional decisions.

CSS does not have cyclomatic complexity. Therefore, its cyclomatic complexity is always listed as zero (0).