



JavaScript CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
8/25/2016	1.0	Original Release	Matthew Swartz
Date	Version No.	Click here to Description.	Name

Table of Contents

No.	Contents	Page No.
1.0	Definitions	4
1.1	SLOC	4
1.2	Physical SLOC	4
1.3	Logical SLOC	4
1.4	Data declaration line	4
1.5	Blank line	4
1.6	Comment line	4
1.7	Executable line of code	4
2.0	Checklist for source statement counts	6
3.0	Examples of logical SLOC counting	7
3.1	Executable Lines	7
3.1.1	Selection Statements	7
3.1.2	Iteration Statements	8
3.1.3	Jump Statements	9
3.1.4	Expression Statements	9
3.1.4	Block Statements	10
3.2	Declaration Lines	10
4.0	Complexity	11

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** – A line that contains declaration of data and used by a compiler or assembler to interpret other elements of the program.

The following lists the JavaScript data keywords:

implements	instanceOf	interface	protected	abstract
boolean	private	double	public	static
class	float	short	byte	char
enum	long	void	int	

Table 1 - JavaScript Data Keywords

Data declarations in JavaScript are defined by

```
var <variable name> <= value>;
```

- 1.5. **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).
- 1.6. **Comment Line** – A comment is defined as a string of zero or more characters that follow language-specific comment delimiter. C/C++ comment delimiters are “//” and “/*”. A whole comment line may span one line and does not contain any compilable source code. An embedded comment can co-exist with compilable source code on the same physical line. Banners and empty comments are treated as types of comments.
- 1.7. **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. An instruction can be stated in a simple or compound form.

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

- Selection statements (if, ? operator, switch)
- Iteration statements (for, while, do-while)
- Empty statement (one or more “;”)

- Jump statements (return, goto, break, continue, exit function)
- Expression statements (function calls, assignment statements, operations, etc)
- Block statements

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

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

2. Checklist for source statement counts

<u>PHYSICAL SLOC COUNTING RULES</u>			
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
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

<u>LOGICAL SLOC COUNTING RULES</u>				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	"for", "while" or "if" statement	1	Count once	"while" is an independent statement
R02	do {...} while (...); statement	2	Count once	Braces {...} and semicolon ; used with this statement are not counted
R03	Statements ending by a semicolon	3	Count once per statement, including empty statement	Semicolons within "for" statement are not counted. Semicolons used with R01 and R02 are not counted
R04	Block delimiters, braces {...}	4	Count once per pair of braces {...}, except where a closing brace is followed by a semicolon, or an opening brace comes after a keyword "else"	Braces used with R01 and R02 are not counted. Function definition is counted once since it is followed by {...}

3. Examples

EXECUTABLE LINES

SELECTION Statement

ESS1 - if, else if, and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
if (<boolean expression> <statements>;	if (x != 0) System.out.print ("non-zero");	1 1
if (<boolean expression> <statements>; else <statements>;	if (x > 0) System.out.print ("positive"); else System.out.print ("negative");	1 1 0 1
if (<boolean expression> <statements>; else if (<boolean expression> <statements>; . . . else <statements>;	if (x == 0) System.out.print ("zero"); else if (x > 0) System.out.print ("positive"); else { System.out.print ("negative"); }	1 1 1 1 0 1 0
NOTE: complexity is not considered, i.e. multiple "&&" or " " as part of the expression.	if ((x != 0) && (x > 0)) System.out.print (x);	1 1

ESS2 – ? : operator

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
Exp1 ? Exp2 : Exp3	x > 0 ? System.out.print ("positive") : System.out.print ("negative");	1

ESS3 – switch and nested switch statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
switch (<expression> { case <constant 1> : <statements>; break; default <statements>;	switch (number) { case 1: foo1(); break; default System.out.print ("invalid	1 0 0 1 1 0 1

}	case”); }	0
ESS4 – try-catch block		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
try {} catch() {}	try { inputFileName=args[0]; } catch (IOException e) { System.err.println(e); System.exit(1); }	1 1 0 1 1 1 0
<u>ITERATION</u> Statement		
EIS1 – for loop		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
for (initialization; condition; increment) statement; NOTE: “for” statement counts as one, no matter how many optional expressions it contains, i.e. for (i = 0, j = 0; i < 5, j < 10; i++, ,j++)	for (i = 0; i < 10; i++) System.out.print (i);	1 1
EIS2 – empty statement		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
for (i = 0; i < SOME_VALUE; i++) ;	for (i = 0; i < 10; i++) ;	2
EIS3 – while loop		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
while (<boolean expression>) <statement>;	while (i < 10) { System.out.print (i); i++; }	1 0 1 1 0
EIS4 – do-while loop		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
do { <statements>; } while (<boolean expression>;	do { ch = getchar(); } while (ch != '\n');	0 0 1 1

EJS5 – for-each loop		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
for (String name: moreNames) System.out.println(name.charAt(0));	for (String n: Names) System.out.println(n.charAt(0));	1 1
<u>JUMP</u> Statement		
EJS1 – return statement		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
return expression	return i;	1
EJS2 – break statement		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
break;	if (l > 10) break;	2
EJS3 – exit function		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
void exit (int return_code);	if (x < 0) exit (1);	2
EJS4 – continue statement		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
continue;	while (!done) { ch = getchar(); if (char == '\n') { done = true; continue; } }	1 0 1 1 0 1 1 0 0
<u>EXPRESSION</u> Statement		
EES1 – function call		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<function_name> (<parameters>);	read_file (name);	1
EES2 – assignment statement		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<name> = <value>;	x = y; char name[6] = "file1";	1 1

	a = 1; b = 2; c = 3;	3
EES3 – empty statement		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
one or more “;” in succession	; ;;	1 3
<u>Block Statement</u>		
EBS1 – blocks		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
/* start of block */ { <definitions> <statement> }	/* start of block */ { i = 0; printf ("%d", i); }	0 0 1 1 0 0
/* end of block */	/* end of block */	0

DECLARATION OR DATA LINES

CDL1 – function prototype, variable declaration		
GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
var <name> = <value>; function <function>(<params>) { STATEMENTS }	var myVar = "global"; function checkscope() { var myVar = "local"; document.write(myVar); }	1 1 1 1 0

4. Complexity

Complexity measures the occurrences of different keywords in code baseline. Below table identifies the categories and their respective keywords that are counted as part of the complexity metrics.

Table 2 – Complexity Keywords List

Math Functions	Trig	Log	Calculations	Conditionals	Logic	Pre-processor	Assignment	Pointer
random	atan2	log	++	else if	===		=	
floor	acos		--	switch	!==			
round	asin		+	while	==			
ceil	atan		-	case	!=			
sqrt	cos		*	else	>=			
abs	sin		/	for	=<			
exp	tan		%	if	>			
max					<			
min								
pow								