shared_whiteboard_model
1.0

css:Sonar way
js:Sonar way
web:Sonar way
xml:Sonar way
2023-03-06

目录

1. shared_whiteboard_model Page 1
1.1. 概述 1
1.2. 问题分析 2
1.3. 问题详情 3
1.4. 质量配置 20

1. shared_whiteboard_model

报告提供了项目指标的概要，显示了与项目质量相关的最重要的指标。如果需要获取更详细的信息，请登陆网站进一步查询。

报告的项目为shared_whiteboard_model，生成时间为2023-03-06，使用的质量配置为 css:Sonar way js:Sonar way web:Sonar way xml:Sonar way，共计 146条规则。

1.1. 概述

编码问题

Bug	可靠性修复工作
5	40min
漏洞	安全修复工作
0	0min
坏味道	技术债务
133	1d6h20min

138 问题

开启问题	138
重开问题	0
确认问题	0
误判问题	0
不修复的问题	0
已解决的问题	0
已删除的问题	0
阻断	9
严重	4
主要	120
次要	5
提示	0

静态分析

项目规模

5323 代码行数	行数	7103
	方法	392
	类	3
	文件	44
	目录	N/A
	重复行(%)	1.7

复杂度

1047 复杂度	文件	29.1

注释(%)

8.2 注释(%)	注释行数	477

1.2. 问题分析

违反最多的规则TOP10	
Variables and functions should not be redeclared	75
Variables should not be shadowed	35
Variables should be declared explicitly	9
Using pseudorandom number generators (PRNGs) is security-sensitive	4
Unused local variables and functions should be removed	3
Function calls should not pass extra arguments	3
Dead stores should be removed	3
Using regular expressions is security-sensitive	2
Comma operator should not be used	2
Functions should not be defined inside loops	1

违规最多的文件TOP5	
path-data-polyfill.js	75
hand.js	29
server.js	5
jwtBoardnameAuth.js	4
intersect.js	3

复杂度最高的文件TOP5	
path-data-polyfill.js	208
board.js	157
hand.js	89
boardData.js	65
pencil.js	49

重复行最多的文件TOP5	
ellipse.js	30
path-data-polyfill.js	30
rect.js	30
hand.js	16
intersect.js	16

1.3. 问题详情

规则	Variables and functions should not be redeclared

规则描述	This rule checks that a declaration doesn't use a name that is already in use. Indeed, it is possible to use the same symbol multiple times as
	either a variable or a function, but doing so is likely to confuse maintainers. Further it's possible that such reassignments are made in error, with
	the developer not realizing that the value of the variable is overwritten by the new assignment.
	This rule also applies to function parameters.
	Noncompliant Code Example
	
	var a = 'foo';
	function a() {} // Noncompliant
	console.log(a); // prints "foo"
	
	function myFunc(arg) {
	  var arg = "event"; // Noncompliant, argument value is lost
	}
	
	fun(); // prints "bar"
	
	function fun() {
	  console.log("foo");
	}
	
	fun(); // prints "bar"
	
	function fun() { // Noncompliant
	  console.log("bar");
	}
	
	fun(); // prints "bar"
	
	Compliant Solution
	
	var a = 'foo';
	function otherName() {}
	console.log(a);
	
	function myFunc(arg) {
	  var newName = "event";
	}
	
	fun(); // prints "foo"
	
	function fun() {
	  print("foo");
	}
	
	fun(); // prints "foo"
	
	function printBar() {
	  print("bar");
	}
	
	printBar(); // prints "bar"

文件名称	违规行
server.js	129, 140, 163, 164

path-data-polyfill.js	488, 489, 532, 533, 545, 546, 555, 556, 569, 570, 579, 580, 581, 582, 583, 584, 593, 594, 595, 596, 605, 606, 607, 608, 617, 618, 630, 631, 643, 649, 655, 661, 667, 668, 669, 670, 679, 680, 681, 682, 691, 692, 701, 702, 756, 757, 769, 770, 779, 787, 795, 796, 797, 798, 821, 822, 824, 824, 835, 836, 850, 851, 852, 853, 855, 856, 857, 858, 875, 876, 955

规则	Variables should not be shadowed
规则描述	Overriding or shadowing a variable declared in an outer scope can strongly impact the readability, and therefore the maintainability, of a piece of code. Further, it could lead maintainers to introduce bugs because they think they're using one variable but are really using another. See CERT, DCL01-C. - Do not reuse variable names in subscopes CERT, DCL51-J. - Do not shadow or obscure identifiers in subscopes

文件名称	违规行
hand.js	295, 296, 117, 166, 174, 194, 200, 216, 222, 250, 260, 278, 293, 320, 345, 357, 382, 404, 127, 88, 249
intersect.js	117
createSVG.js	177, 85
canvascolor.js	68, 162
grid.js	55, 99
board.js	644
cursor.js	106
templating.js	31
zoom.js	39
path-data-polyfill.js	362, 362, 362

规则	Variables should be declared explicitly

规则描述	JavaScript variable scope can be particularly difficult to understand and get right. The situation gets even worse when you consider the accidental creation of global variables, which is what happens when you declare a variable inside a function or the for clause of a for-loop without using the let , const or var keywords. let and const were introduced in ECMAScript 2015, and are now the preferred keywords for variable declaration.
	Noncompliant Code Example
	
	function f(){
	i = 1; // Noncompliant; i is global
	
	for (j = 0; j < array.length; j++) { // Noncompliant; j is global now too
	// ...
	}
	}
	
	Compliant Solution
	
	function f(){
	var i = 1;
	
	for (let j = 0; j < array.length; j++) {
	// ...
	}
	}

文件名称	违规行
jwtBoardnameAuth.js	28
server.js	14
jwtauth.js	29
intersect.js	66
hand.js	74, 240
intersect.js	65
sockets.js	32
minitpl.js	27

规则	Using pseudorandom number generators (PRNGs) is security-sensitive

规则描述	Using pseudorandom number generators (PRNGs) is security-sensitive. For example, it has led in the past to the following vulnerabilities:
	
	CVE-2013-6386
	CVE-2006-3419
	CVE-2008-4102
	
	When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that
	will be generated, and use this guess to impersonate another user or access sensitive information.
	As the Math.random() function relies on a weak pseudorandom number generator, this function should not be used for security-critical
	applications or for protecting sensitive data. In such context, a cryptographically strong pseudorandom number generator (CSPRNG) should be used
	instead.
	Ask Yourself Whether
	
	the code using the generated value requires it to be unpredictable. It is the case for all encryption mechanisms or when a secret value, such
	as a password, is hashed.
	the function you use generates a value which can be predicted (pseudo-random).
	the generated value is used multiple times.
	an attacker can access the generated value.
	
	You are at risk if you answered yes to the first question and any of the following ones.
	Recommended Secure Coding Practices
	
	Use a cryptographically strong pseudorandom number generator (CSPRNG) like crypto.getRandomValues() .
	Use the generated random values only once.
	You should not expose the generated random value. If you have to store it, make sure that the database or file is secure.
	
	Questionable Code Example
	
	const val = Math.random(); // Questionable
	// Check if val is used in a security context.
	
	Compliant Solution
	
	// === Client side ===
	const crypto = window.crypto || window.msCrypto;
	var array = new Uint32Array(1);
	crypto.getRandomValues(array); // Compliant for security-sensitive use cases
	
	// === Server side ===
	const crypto = require('crypto');
	const buf = crypto.randomBytes(1); // Compliant for security-sensitive use cases
	
	See

	OWASP Top 10 2017 Category A3 - Sensitive Data Exposure MITRE, CWE-338 - Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG) MITRE, CWE-330 - Use of Insufficiently Random Values MITRE, CWE-326 - Inadequate Encryption Strength CERT, MSC02-J. - Generate strong random numbers CERT, MSC30-C. - Do not use the rand() function for generating pseudorandom numbers CERT, MSC50-CPP. - Do not use std::rand() for generating pseudorandom numbers Derived from FindSecBugs rule Predictable Pseudo Random Number Generator

文件名称	违规行
sockets.js	204
check_output_directory.js	20
board.js	631, 590

规则	Function calls should not pass extra arguments

规则描述	You can easily call a JavaScript function with more arguments than the function needs, but the extra arguments will be just ignored by function execution.
	Note that this rule requires Node.js to be available during analysis.
	Noncompliant Code Example
	
	function say(a, b) {
	print(a + " " + b);
	}
	
	say("hello", "world", "!"); // Noncompliant; last argument is not used
	
	Exceptions
	No issue is reported when arguments is used in the body of the function being called.
	
	function doSomething(a, b) {
	compute(arguments);
	}
	
	doSomething(1, 2, 3) // Compliant
	
	See
	
	MISRA C:2004, 16.6 - The number of arguments passed to a function shall match the number of parameters.
	MITRE, CWE-628 - Function Call with Incorrectly Specified Arguments
	CERT, DCL07-C. - Include the appropriate type information in function declarators
	CERT, EXP37-C. - Call functions with the correct number and type of arguments

文件名称	违规行
hand.js	461, 467, 472

规则	Dead stores should be removed

规则描述	A dead store happens when a local variable is assigned a value that is not read by any subsequent instruction. Calculating or retrieving a value
	only to then overwrite it or throw it away, could indicate a serious error in the code. Even if it's not an error, it is at best a waste of resources.
	Therefore all calculated values should be used.
	Noncompliant Code Example
	
	i = a + b; // Noncompliant; calculation result not used before value is overwritten
	i = compute();
	
	Compliant Solution
	
	i = a + b;
	i += compute();
	
	Exceptions
	This rule ignores initializations to -1, 0, 1, null , undefined , true , false , "" ,
	[] and {} .
	This rule also ignores variables declared with object destructuring using rest syntax (used to exclude some properties from object):
	
	let {a, b, ...rest} = obj; // 'a' and 'b' are ok
	doSomething(rest);
	
	let [x1, x2, x3] = arr; // but 'x1' is noncompliant, as omitting syntax can be used: "let [, x2, x3] = arr;"
	doSomething(x2, x3);
	
	See
	
	MITRE, CWE-563 - Assignment to Variable without Use ('Unused Variable')
	CERT, MSC13-C. - Detect and remove unused values
	CERT, MSC56-J. - Detect and remove superfluous code and values

文件名称	违规行
jwtBoardnameAuth.js	47
log.js	14
text.js	101

规则	Unused local variables and functions should be removed

规则描述	If a local variable or a local function is declared but not used, it is dead code and should be removed. Doing so will improve maintainability
	because developers will not wonder what the variable or function is used for.
	Noncompliant Code Example
	
	function numberOfMinutes(hours) {
	var seconds = 0; // seconds is never used
	return hours * 60;
	}
	
	Compliant Solution
	
	function numberOfMinutes(hours) {
	return hours * 60;
	}

文件名称	违规行
jwtBoardnameAuth.js	47
log.js	14
text.js	101

规则	Comma operator should not be used

规则描述	The comma operator takes two expressions, executes them from left to right and returns the result of the second one. Use of this operator is
	generally detrimental to the readability and reliability of code, and the same effect can be achieved by other means.
	Noncompliant Code Example
	
	i = a += 2, a + b; // What's the value of i ?
	
	Compliant Solution
	
	a += 2;
	i = a + b;
	
	Exceptions
	Use of comma operator is tolerated in initialization and increment expressions of for loops.
	
	for(i = 0, j = 5; i < 6; i++, j++) { ... }
	
	See
	
	MISRA C:2004, 12.10 - The comma operator shall not be used.
	MISRA C++:2008, 5-18-1 - The comma operator shall not be used.
	MISRA C:2012, 12.3 - The comma operator should not be used

文件名称	违规行
jwtBoardnameAuth.js	27
jwtauth.js	28

规则	Using regular expressions is security-sensitive

规则描述	Using regular expressions is security-sensitive. It has led in the past to the following vulnerabilities:
	
	CVE-2017-16021
	CVE-2018-13863
	
	Evaluating regular expressions against input strings is potentially an extremely CPU-intensive task. Specially crafted regular expressions such as
	(a+)+s will take several seconds to evaluate the input string aaaaaaaaaaaaaaaaaaaaaaaaaaaaaabs . The problem is that with every additional a character added to the input, the time required to evaluate the regex doubles. However, the equivalent regular expression, a+s (without grouping) is efficiently evaluated in milliseconds and scales linearly with the input size.
	Evaluating such regular expressions opens the door to a href="https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS">Regular expression Denial of Service (ReDoS) attacks. In the
	context of a web application, attackers can force the web server to spend all of its resources evaluating regular expressions thereby making the
	service inaccessible to genuine users.
	This rule flags any execution of a hardcoded regular expression which has at least 3 characters and at least two instances of any of the following
	characters: *+{ .
	Example: (a+)*
	Ask Yourself Whether
	
	the executed regular expression is sensitive and a user can provide a string which will be analyzed by this regular expression.
	your regular expression engine performance decrease with specially crafted inputs and regular expressions.
	
	You may be at risk if you answered yes to any of those questions.
	Recommended Secure Coding Practices
	Check whether your regular expression engine (the algorithm executing your regular expression) has any known vulnerabilities. Search for
	vulnerability reports mentioning the one engine you're are using.
	Use if possible a library which is not vulnerable to Redos Attacks such as Google Re2 .
	Remember also that a ReDos attack is possible if a user-provided regular expression is executed. This rule won't detect this kind of injection.
	Sensitive Code Example
	
	const regex = /(a+)+b/; // Sensitive
	const regex2 = new RegExp("(a+)+b"); // Sensitive
	
	str.search("(a+)+b"); // Sensitive
	str.match("(a+)+b"); // Sensitive
	str.split("(a+)+b"); // Sensitive
	
	Note: String.matchAll does not raise any issue as it is not supported by NodeJS.
	See
	
	OWASP Top 10 2017 Category A1 - Injection
	MITRE, CWE-624 - Executable Regular Expression Error

	OWASP Regular expression Denial of Service - ReDoS

文件名称	违规行
jwtBoardnameAuth.js	47
log.js	10

规则	Functions should not be defined inside loops

规则描述	Defining a function inside of a loop can yield unexpected results. Such a function keeps references to the variables which are defined in outer
	scopes. All function instances created inside the loop therefore see the same values for these variables, which is probably not expected.
	Noncompliant Code Example
	
	var funs = [];
	for (var i = 0; i < 13; i++) {
	funs[i] = function() { // Non-Compliant
	return i;
	};
	}
	console.log(funs[0]()); // 13 instead of 0
	console.log(funs[1]()); // 13 instead of 1
	console.log(funs[2]()); // 13 instead of 2
	console.log(funs[3]()); // 13 instead of 3
	...

文件名称	违规行
check_output_directory.js	39

规则	"delete" should not be used on arrays

规则描述	The delete operator can be used to remove a property from any object. Arrays are objects, so the delete operator can be used here too, but if it is, a hole will be left in the array because the indexes/keys won't be shifted to reflect the deletion.
	The proper method for removing an element at a certain index would be:
	
	Array.prototype.splice - add/remove elements from the array
	Array.prototype.pop - add/remove elements from the end of the array
	Array.prototype.shift - add/remove elements from the beginning of the array
	
	Noncompliant Code Example
	
	var myArray = ['a', 'b', 'c', 'd'];
	
	delete myArray[2]; // Noncompliant. myArray => ['a', 'b', undefined, 'd']
	console.log(myArray[2]); // expected value was 'd' but output is undefined
	
	Compliant Solution
	
	var myArray = ['a', 'b', 'c', 'd'];
	
	// removes 1 element from index 2
	removed = myArray.splice(2, 1); // myArray => ['a', 'b', 'd']
	console.log(myArray[2]); // outputs 'd'

文件名称	违规行
hand.js	75

规则	Assignments should not be redundant

规则描述	The transitive property says that if a == b and b == c , then a == c . In such cases, there's no point in assigning a to c or vice versa because they're already equivalent.
	This rule raises an issue when an assignment is useless because the assigned-to variable already holds the value on all execution paths.
	Noncompliant Code Example
	
	a = b;
	c = a;
	b = c; // Noncompliant: c and b are already the same
	
	Compliant Solution
	
	a = b;
	c = a;

文件名称	违规行
path-data-polyfill.js	57

规则	Using command line arguments is security-sensitive

规则描述	Using command line arguments is security-sensitive. It has led in the past to the following vulnerabilities:
	
	CVE-2018-7281
	CVE-2018-12326
	CVE-2011-3198
	
	Command line arguments can be dangerous just like any other user input. They should never be used without being first validated and sanitized.
	Remember also that any user can retrieve the list of processes running on a system, which makes the arguments provided to them visible. Thus
	passing sensitive information via command line arguments should be considered as insecure.
	This rule raises an issue when on every program entry points (main methods) when command line arguments are used. The goal is to guide
	security code reviews.
	Ask Yourself Whether
	
	any of the command line arguments are used without being sanitized first.
	your application accepts sensitive information via command line arguments.
	
	If you answered yes to any of these questions you are at risk.
	Recommended Secure Coding Practices
	Sanitize all command line arguments before using them.
	Any user or application can list running processes and see the command line arguments they were started with. There are safer ways of providing
	sensitive information to an application than exposing them in the command line. It is common to write them on the process' standard input, or give the
	path to a file containing the information.
	Questionable Code Example
	
	// The process object is a global that provides information about, and control over, the current Node.js process
	var param = process.argv[2]; // Questionable: check how the argument is used
	console.log('Param: ' + param);
	
	See
	
	OWASP Top 10 2017 Category A1 - Injection
	MITRE, CWE-88 - Argument Injection or Modification
	MITRE, CWE-214 - Information Exposure Through Process Environment
	SANS Top 25 - Insecure Interaction Between Components

文件名称	违规行
createSVG.js	219

规则	Boolean checks should not be inverted

规则描述	It is needlessly complex to invert the result of a boolean comparison. The opposite comparison should be made instead.
	Note that this rule requires Node.js to be available during analysis.
	Noncompliant Code Example
	
	if (!(a === 2)) { ... } // Noncompliant
	
	Compliant Solution
	
	if (a !== 2) { ... }

文件名称	违规行
hand.js	113

规则	The output of functions that don't return anything should not be used

规则描述	If a function does not return anything, it makes no sense to use its output. Specifically, passing it to another function, or assigning its "result" to a variable is probably a bug because such functions return undefined , which is probably not what was intended.
	Note that this rule requires Node.js to be available during analysis.
	Noncompliant Code Example
	
	function foo() {
	console.log("Hello, World!");
	}
	
	a = foo();
	
	Compliant Solution
	
	function foo() {
	console.log("Hello, World!");
	}
	
	foo();

文件名称	违规行
zoom.js	78

规则	A conditionally executed single line should be denoted by indentation

规则描述	In the absence of enclosing curly braces, the line immediately after a conditional is the one that is conditionally executed. By both convention
	and good practice, such lines are indented. In the absence of both curly braces and indentation the intent of the original programmer is entirely
	unclear and perhaps not actually what is executed. Additionally, such code is highly likely to be confusing to maintainers.
	Noncompliant Code Example
	
	if (condition) // Noncompliant
	doTheThing();
	
	doTheOtherThing();
	somethingElseEntirely();
	
	foo();
	
	Compliant Solution
	
	if (condition)
	  doTheThing();
	
	doTheOtherThing();
	somethingElseEntirely();
	
	foo();

文件名称	违规行
hand.js	430

规则	"switch" statements should have at least 3 "case" clauses

规则描述	switch statements are useful when there are many different cases depending on the value of the same expression.
	For just one or two cases however, the code will be more readable with if statements.
	Note that this rule requires Node.js to be available during analysis.
	Noncompliant Code Example
	
	switch (variable) {
	case 0:
	doSomething();
	break;
	default:
	doSomethingElse();
	break;
	}
	
	Compliant Solution
	
	if (variable == 0) {
	doSomething();
	} else {
	doSomethingElse();
	}
	
	See
	
	MISRA C:2012, 16.6 - Every switch statement shall have at least two switch-clauses

文件名称	违规行
eraser.js	68

规则	Property names should not be duplicated within a class or object literal

规则描述	JavaScript allows duplicate property names in classes and object literals, but only the last instance of a duplicated name determines the actual
	value that will be used for it. Therefore, changing values of other occurrences of a duplicated name will have no effect and may cause
	misunderstandings and bugs.
	Defining a class with a duplicated constructor will generate an error.
	Before ECMAScript 2015, using duplicate names will generate an error in JavaScript strict mode code.
	Noncompliant Code Example
	
	var data = {
	"key": "value",
	"1": "value",
	"key": "value", // Noncompliant - duplicate of "key"
	'key': "value", // Noncompliant - duplicate of "key"
	key: "value", // Noncompliant - duplicate of "key"
	\u006bey: "value", // Noncompliant - duplicate of "key"
	"\u006bey": "value", // Noncompliant - duplicate of "key"
	"\x6bey": "value", // Noncompliant - duplicate of "key"
	1: "value" // Noncompliant - duplicate of "1"
	}
	
	Compliant Solution
	
	var data = {
	"key": "value",
	"1": "value",
	"key2": "value",
	'key3': "value",
	key4: "value",
	\u006bey5: "value",
	"\u006bey6": "value",
	"\x6bey7": "value",
	1b: "value"
	}

文件名称	违规行
pencil.js	213

1.4. 质量配置

质量配置	css:Sonar way Bug:15 坏味道:8	
规则	类型	违规级别
CSS properties should be valid	Bug	阻断
Single line comment syntax should not be used	Bug	阻断
"calc" operands should be correctly spaced	Bug	阻断
Units should be valid	Bug	阻断
Color definitions should be valid	Bug	阻断
Shorthand properties that override related longhand properties should be avoided	Bug	严重
"linear-gradient" directions should be valid	Bug	严重

Selectors should be known	Bug	严重
"!important" should not be used on "keyframes"	Bug	主要
Properties should not be duplicated	Bug	主要
Media features should be valid	Bug	主要
"at-rules" should be valid	Bug	主要
Pseudo-class selectors should be valid	Bug	主要
Font declarations should contain at least one generic font family	Bug	主要
Pseudo-element selectors should be valid	Bug	主要
Selectors should not be duplicated	坏味道	主要
CSS files should not be empty	坏味道	主要
Strings should not contain new lines	坏味道	主要
Duplicated font names should be removed	坏味道	主要
Empty blocks should be removed	坏味道	主要
Multi-line comments should not be empty	坏味道	次要
Extra semicolons should be removed	坏味道	次要
Duplicate imports should be removed	坏味道	次要

质量配置	js:Sonar way Bug:41 漏洞:3 坏味道:44	
规则	类型	违规级别
Callbacks of array methods should have return statements	Bug	阻断
Loops should not be infinite	Bug	阻断
"yield" expressions should not be used outside generators	Bug	阻断
Jump statements should not occur in "finally" blocks	Bug	严重
"in" should not be used with primitive types	Bug	严重
Function calls should not pass extra arguments	Bug	严重
"Symbol" should not be used as a constructor	Bug	严重
Results of "in" and "instanceof" should be negated rather than operands	Bug	严重
"super()" should be invoked appropriately	Bug	严重
Destructuring patterns should not be empty	Bug	主要
Conditionally executed blocks should be reachable	Bug	主要
Property names should not be duplicated within a class or object literal	Bug	主要
Return values from functions without side effects should not be ignored	Bug	主要
"NaN" should not be used in comparisons	Bug	主要
Generators should "yield" something	Bug	主要
Function argument names should be unique	Bug	主要
Related "if/else if" statements and "cases" in a "switch" should not have the same condition	Bug	主要

All branches in a conditional structure should not have exactly the same implementation	Bug	主要
The output of functions that don't return anything should not be used	Bug	主要
Values should not be uselessly incremented	Bug	主要
Jump statements should not be followed by dead code	Bug	主要
Special identifiers should not be bound or assigned	Bug	主要
Properties of variables with "null" or "undefined" values should not be accessed	Bug	主要
A "for" loop update clause should move the counter in the right direction	Bug	主要
Variables should not be self-assigned	Bug	主要
Non-empty statements should change control flow or have at least one side-effect	Bug	主要
Calls should not be made to non-callable values	Bug	主要
Non-existent operators '=+', '=-' and '=!' should not be used	Bug	主要
"new" operators should be used with functions	Bug	主要
Identical expressions should not be used on both sides of a binary operator	Bug	主要
Array-mutating methods should not be used misleadingly	Bug	主要
Strict equality operators should not be used with dissimilar types	Bug	主要
Setters should not return values	Bug	主要
Comma and logical OR operators should not be used in switch cases	Bug	主要
Collection elements should not be replaced unconditionally	Bug	主要
Bitwise operators should not be used in boolean contexts	Bug	主要
Attempts should not be made to update "const" variables	Bug	主要
Errors should not be created without being thrown	Bug	主要
Collection sizes and array length comparisons should make sense	Bug	主要
"delete" should be used only with object properties	Bug	次要
"with" statements should not be used	Bug	次要
Cross-document messaging domains should be carefully restricted	漏洞	严重
Debugger statements should not be used	漏洞	次要
"alert(...)" should not be used	漏洞	次要
Octal values should not be used	坏味道	阻断
Variables should be declared explicitly	坏味道	阻断
"future reserved words" should not be used as identifiers	坏味道	阻断

"switch" statements should not contain non-case labels	坏味道	阻断
Function returns should not be invariant	坏味道	阻断
Switch cases should end with an unconditional "break" statement	坏味道	阻断
Conditionals should start on new lines	坏味道	严重
A conditionally executed single line should be denoted by indentation	坏味道	严重
Equality operators should not be used in "for" loop termination conditions	坏味道	严重
Boolean expressions should not be gratuitous	坏味道	主要
Redundant pairs of parentheses should be removed	坏味道	主要
Functions should not be called both with and without "new"	坏味道	主要
Comma operator should not be used	坏味道	主要
Multiline blocks should be enclosed in curly braces	坏味道	主要
Labels should not be used	坏味道	主要
Variables should not be shadowed	坏味道	主要
"switch" statements should not have too many "case" clauses	坏味道	主要
"indexOf" checks should not be for positive numbers	坏味道	主要
Arguments to built-in functions should match documented types	坏味道	主要
Nested blocks of code should not be left empty	坏味道	主要
Dead stores should be removed	坏味道	主要
Array indexes should be numeric	坏味道	主要
Variables and functions should not be redeclared	坏味道	主要
"delete" should not be used on arrays	坏味道	主要
Jump statements should not be used unconditionally	坏味道	主要
Function parameters with default values should be last	坏味道	主要
Two branches in a conditional structure should not have exactly the same implementation	坏味道	主要
Assignments should not be redundant	坏味道	主要
Functions should not be defined inside loops	坏味道	主要
Collection and array contents should be used	坏味道	主要
Boolean checks should not be inverted	坏味道	次要
Default export names and file names should match	坏味道	次要
A "while" loop should be used instead of a "for" loop	坏味道	次要
Function call arguments should not start on new lines	坏味道	次要
Extra semicolons should be removed	坏味道	次要
Return of boolean expressions should not be wrapped into an "if-then-else" statement	坏味道	次要

Unnecessary imports should be removed	坏味道	次要
Wrapper objects should not be used for primitive types	坏味道	次要
Unary operators "+" and "-" should not be used with objects	坏味道	次要
Multiline string literals should not be used	坏味道	次要
"switch" statements should have at least 3 "case" clauses	坏味道	次要
The global "this" object should not be used	坏味道	次要
"catch" clauses should do more than rethrow	坏味道	次要
Unused local variables and functions should be removed	坏味道	次要

质量配置	web:Sonar way Bug:11 坏味道:7	
规则	类型	违规级别
"<title>" should be present in all pages	Bug	主要
<script>...</script> elements should not be nested	Bug	主要
"<!DOCTYPE>" declarations should appear before "<html>" tags	Bug	主要
Elements deprecated in HTML5 should not be used	Bug	主要
"<frames>" should have a "title" attribute	Bug	次要
Server-side image maps ("ismap" attribute) should not be used	Bug	次要
"" and "" tags should be used	Bug	次要
Images tags and buttons should have an "alt" attribute	Bug	次要
Flash animations should be embedded using both "<object>" and "<embed>"	Bug	次要
" " and "<dt>" item tags should be in " ", " " or "<dl>" container tags	Bug	次要
"<fieldset>" tags should contain a "<legend>"	Bug	次要
Track uses of "FIXME" tags	坏味道	主要
Videos should have subtitles	坏味道	主要
Attributes deprecated in HTML5 should not be used	坏味道	主要
Sections of code should not be commented out	坏味道	主要
Meta tags should not be used to refresh or redirect	坏味道	主要
Links should not directly target images	坏味道	主要
Track uses of "TODO" tags	坏味道	提示

质量配置	xml:Sonar way Bug:1 坏味道:3	
规则	类型	违规级别

XML files containing a prolog header should start with "<?xml" characters	Bug	严重
Track uses of "FIXME" tags	坏味道	主要
Sections of code should not be commented out	坏味道	主要
Track uses of "TODO" tags	坏味道	提示