




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 Java


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



JavaScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVASCRIPT code

All rules 285

Vulnerability 29

Bug 62

Security Hotspot 43

Code Smell 151

Quick Fix 41

Tags ▾

Search by name... 🔍

Multiline blocks should be enclosed in curly braces

Code Smell

Boolean expressions should not be gratuitous

Code Smell

Variables should be used in the blocks where they are declared

Code Smell

Parameters should be passed in the correct order

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

Unused assignments should be removed

Code Smell

Function parameters with default values should be last

Code Smell

Functions should not be defined inside loops

Code Smell

"switch" statements should not have too many "case" clauses

Code Smell

Only "while", "do", "for" and "switch" statements should be labelled

Code Smell

Sections of code should not be commented out

Code Smell

Collection sizes and array length comparisons should make sense

Analyze your code

Bug

Major

Quick Fix

The size of a collection and the length of an array are always greater than or equal to zero. So testing that a size or length is greater than or equal to zero doesn't make sense, since the result is always `true`. Similarly testing that it is less than zero will always return `false`. Perhaps the intent was to check the non-emptiness of the collection or array instead.

Noncompliant Code Example

```
if (someSet.size >= 0) {...} // Noncompliant

if (someMap.size < 0) {...} // Noncompliant

const result = someArray.length >= 0; // Noncompliant
```

Compliant Solution

```
if (someSet.size > 0) {...}

if (someMap.size == 0) {...}

const result = someArray.length > 0;
```

Available In:

sonarlint

sonarcloud






sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

[Privacy Policy](#)

https://rules.sonarsource.com/javascript/RSPEC-3981

1/2

<div>Unused function parameters should be removed</div> <div> Code Smell</div>
<div>Track uses of "FIXME" tags</div> <div> Code Smell</div>
<div>Assignments should not be made from within sub-expressions</div> <div> Code Smell</div>
<div>Labels should not be used</div> <div> Code Smell</div>
<div>Variables should not be shadowed</div> <div></div>