




 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 rules285

Vulnerability29

Bug62

Security Hotspot43

Code Smell151

Quick Fix41

Tags ▾

Search by name... 🔍

String literals should not be duplicated

Code Smell

Expressions should not be too complex

Code Smell

Local storage should not be used

Vulnerability

Template literal placeholder syntax should not be used in regular strings

Bug

Built-in objects should not be overridden

Bug

"for...in" loops should filter properties before acting on them

Bug

Results of operations on strings should not be ignored

Bug

Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression

Code Smell

"for in" should not be used with iterables

Code Smell

Functions should use "return" consistently

Code Smell

Variables and functions should not be declared in the global scope

Code Smell

Arithmetic operators should only have numbers as operands

Array-mutating methods should not be used misleadingly

Analyze your code

Code Smell

Major ?

Many of JavaScript's Array methods return an altered version of the array while leaving the source array intact. `reverse` and `sort` do not fall into this category. Instead, they alter the source array *in addition* to returning the altered version, which is likely not what was intended.

This rule raises an issue when the return values of these methods are assigned, which could lead maintainers to overlook the fact that the original value is altered.

Noncompliant Code Example

```
var b = a.reverse(); // Noncompliant
var d = c.sort(); // Noncompliant
```

Compliant Solution

```
var b = [...a].reverse(); // de-structure and create a new a.reverse();

c.sort(); // this sorts array in place
```

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-4043

1/2

 Code Smell
<b>Values not convertible to numbers should not be used in numeric comparisons</b>  Code Smell
<b>Arithmetic operations should not result in "NaN"</b>  Code Smell
<b>"arguments" should not be accessed directly</b>  Code Smell
<b>Comparison operators should not be used with strings</b>