




 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... 🔍


 Vulnerability


Template literal placeholder syntax should not be used in regular strings


 Bug


 Built-in objects should not be overridden


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


 Results of operations on strings should not be ignored


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


 "for in" should not be used with iterables

 Functions should use "return" consistently

 Variables and functions should not be declared in the global scope


 Arithmetic operators should only have numbers as operands


 Values not convertible to numbers should not be used in numeric comparisons


 Arithmetic operations should not

Collection and array contents should be used

Analyze your code

 Code Smell

 Major ?

 unused suspicious

When a collection is populated but its contents are never used, then it is surely some kind of mistake. Either refactoring has rendered the collection moot, or an access is missing.

This rule raises an issue when no methods are called on a collection other than those that add or remove values.

Noncompliant Code Example

```
function getLength(a, b, c) {
  const strings = []; // Noncompliant
  strings.push(a);
  strings.push(b);
  strings.push(c);

  return a.length + b.length + c.length;
}
```

Compliant Solution

```
function getLength(a, b, c) {
  return a.length + b.length + c.length;
}
```

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-4030

1/2

<b>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>  Code Smell
<b>Property getters and setters should come in pairs</b>  Code Smell
<b>JavaScript parser failure</b>