

































-  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... 🔍

Functions should not be given the same argument

Bug

Alternatives in regular expressions should be grouped when used with anchors

Bug

Promise rejections should not be caught by 'try' block

Bug

Collection elements should not be replaced unconditionally

Bug

Errors should not be created without being thrown

Bug

Collection sizes and array length comparisons should make sense

Bug

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

Bug

Destructuring patterns should not be empty

Bug

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

Bug

Comma and logical OR operators should not be used in switch cases

Bug

Generators should "yield" something

Bug

Attempts should not be made to update "const" variables

Bug

"Symbol" should not be used as a constructor

Analyze your code

BugCritical?es2015

Symbol is a primitive type introduced in ECMAScript2015. Its instances are mainly used as unique property keys.

An instance can only be created by using Symbol as a function. Using Symbol with the new operator will raise a TypeError.

Noncompliant Code Example

```
const sym = new Symbol("foo"); // Noncompliant
```

Compliant Solution

```
const sym = Symbol("foo");
```





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-3834

1/2

 Bug
Strict equality operators should not be used with dissimilar types  Bug
"new" operators should be used with functions  Bug
Non-existent operators '+=', '-=' and '!=' should not be used  Bug
"NaN" should not be used in comparisons