
































-  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... 🔍

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

 Code Smell

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

 Code Smell

Arithmetic operations should not result in "NaN"

 Code Smell

"arguments" should not be accessed directly

 Code Smell

Comparison operators should not be used with strings

 Code Smell

Property getters and setters should come in pairs


 Code Smell

JavaScript parser failure

 Code Smell

Arguments to built-in functions should match documented types

Analyze your code

 Code Smell  Major ?

The types of the arguments to built-in functions are specified in the JavaScript language specifications. Calls to these functions should conform to the documented types, otherwise the result will most likely not be what was expected (e.g.: the call would always return `false`).

Noncompliant Code Example

```
const isTooSmall = Math.abs(x < 0.0042);
```





Compliant Solution

```
const isTooSmall = Math.abs(x) < 0.0042;
```

Available In:

 |  | 

© 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](#)

|   |
|---|
| <div>The ternary operator should not be used</div> <div> Code Smell</div>                  |
| <div>"===" and "!===" should be used instead of "==" and "!="</div> <div> Code Smell</div> |
| <div>Functions should not have too many lines of code</div> <div> Code Smell</div>         |
| <div>Track comments matching a regular expression</div> <div> Code Smell</div>             |
|   |