




 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... 🔍

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

The ternary operator should not be used

Code Smell

"===" and "!==" should be used instead of "==" and "!="

Code Smell

Functions should not have too many lines of code

Code Smell

Track comments matching a regular expression

Code Smell

Functions should not be called both with and without "new"

Analyze your code

Code SmellMajor ?

Constructor functions, which create new object instances, must only be called with new. Non-constructor functions must not. Mixing these two usages could lead to unexpected results at runtime.

Noncompliant Code Example

```
function getNum() {
    return 5;
}

function Num(numeric, alphabetic) {
    this.numeric = numeric;
    this.alphabetic = alphabetic;
}

var myFirstNum = getNum();
var my2ndNum = new getNum(); // Noncompliant. An empty object is created.

var myNumObj1 = new Num();
var myNumObj2 = Num(); // Noncompliant. undefined is returned.
```






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-3686

1/2

 Code Smell
Statements should be on separate lines  Code Smell
Magic numbers should not be used  Code Smell
Collapsible "if" statements should be merged  Code Smell
Standard outputs should not be used directly to log anything  Code Smell