




 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... 🔍


Weak SSL/TLS protocols should not be used

 Vulnerability


Origins should be verified during cross-origin communications

 Vulnerability


Regular expressions should not be vulnerable to Denial of Service attacks

 Vulnerability


File uploads should be restricted

 Vulnerability


Function calls should not pass extra arguments

 Bug


Regular expressions should be syntactically valid

 Bug


Getters and setters should access the expected fields

 Bug


"super()" should be invoked appropriately

 Bug


"Symbol" should not be used as a constructor

 Bug

Results of "in" and "instanceof" should be negated rather than operands

 Bug


"in" should not be used with primitive types


 Bug


A compare function should be provided when using "Array.prototype.sort()"

Function returns should not be invariant

Analyze your code

 Code Smell

 Blocker






When a function is designed to return an invariant value, it may be poor design, but it shouldn't adversely affect the outcome of your program. However, when it happens on all paths through the logic, it is likely a mistake.

This rule raises an issue when a function contains several `return` statements that all return the same value.

Noncompliant Code Example





```
function foo(a) { // Noncompliant
  let b = 12;
  if (a) {
    return b;
  }
  return b;
}
```

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](#)

https://rules.sonarsource.com/javascript/RSPEC-3516

1/2

 Bug
<div>Jump statements should not occur in "finally" blocks</div> <div> Bug</div>
<div>Using slow regular expressions is security-sensitive</div> <div> Security Hotspot</div>
<div>Using publicly writable directories is security-sensitive</div> <div> Security Hotspot</div>
<div>Using clear-text protocols is security-sensitive</div>