




 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... 🔍

Security Hotspot

Disabling resource integrity features is security-sensitive

Security Hotspot

Disclosing fingerprints from web application technologies is security-sensitive

Security Hotspot

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

Security Hotspot

Delivering code in production with debug features activated is security-sensitive

Security Hotspot

Creating cookies without the "HttpOnly" flag is security-sensitive

Security Hotspot

Creating cookies without the "secure" flag is security-sensitive

Security Hotspot

Using hardcoded IP addresses is security-sensitive

Security Hotspot

Regular expression quantifiers and character classes should be used concisely

Code Smell

Regular expression literals should be used when possible

Code Smell

"await" should not be used redundantly

Code Smell

"for of" should be used with Iterables

All code should be reachable

Analyze your code

Bug

Major

Quick Fix

cwe unused

Jump statements (return, break and continue) and throw expressions move control flow out of the current code block. So any statements that come after a jump are dead code.

Noncompliant Code Example

```
function fun(a) {
  var i = 10;
  return i + a;
  i++;           // Noncompliant; this is never executed
}
```

Compliant Solution

```
function fun(int a) {
  var i = 10;
  return i + a;
}
```

See

MITRE, CWE-561

 - Dead Code

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-1763

1/2

 Code Smell
Imports from the same modules should be merged  Code Smell
Jump statements should not be redundant  Code Smell
Default export names and file names should match  Code Smell
The global "this" object should not be used