




 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... 🔍

Code Smell

Tests should check which exception is thrown

Code Smell

Character classes in regular expressions should not contain the same character twice

Code Smell

Names of regular expressions named groups should be used

Code Smell

Regular expressions should not be too complicated

Code Smell

Shorthand promises should be used

Code Smell

Template literals should not be nested

Code Smell

"in" should not be used on arrays

Code Smell

Assignments should not be redundant

Code Smell

Functions should not have identical implementations

Code Smell

Sparse arrays should not be declared

Code Smell

Array-mutating methods should not be used misleadingly

Code Smell

Collection and array contents should be used

Non-empty statements should change control flow or have at least one side-effect

Analyze your code

Bug

Major

cwe unused

Any statement (other than a null statement, which means a statement containing only a semicolon ;) which has no side effect and does not result in a change of control flow will normally indicate a programming error, and therefore should be refactored.

Noncompliant Code Example

```
a == 1; // Noncompliant; was assignment intended?
var msg = "Hello, "
    "World!"; // Noncompliant; have we forgotten '+' operator
```

See

MITRE, CWE-482

 - Comparing instead of Assigning

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-905

1/2

-----  Code Smell
Functions should always return the same type  Code Smell
Arguments to built-in functions should match documented types  Code Smell
Literals should not be thrown  Code Smell
Functions should not be called both with and without "new"