

































-  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... 🔍

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

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"

Code Smell

Array indexes should be numeric

Code Smell

Assertion arguments should be passed in the correct order

Regular expressions should not contain control characters

Analyze your code

BugMajor?regex

Entries in the ASCII table below code 32 are known as control characters or non-printing characters. As they are not common in JavaScript strings, using these invisible characters in regular expressions is most likely a mistake.

Noncompliant Code Example

```
const pattern1 = /\x1a/;
const pattern2 = new RegExp('\x1a');
```

Compliant Solution

```
const pattern1 = /\x20/;
const pattern2 = new RegExp('\x20');
```





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-6324

1/2

passed in the correct order
 Code Smell
Ternary operators should not be nested
 Code Smell
"delete" should not be used on arrays
 Code Smell
Variables and functions should not be redeclared
 Code Smell
"indexOf" checks should not be for positive numbers