




 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...

Object literal shorthand syntax should be used

Code Smell

Strings and non-strings should not be added

Code Smell

Object literal syntax should be used

Code Smell

"undefined" should not be assigned

Code Smell

Trailing commas should not be used

Code Smell

Array constructors should not be used

Code Smell

Quotes for string literals should be used consistently

Code Smell

Statements should end with semicolons

Code Smell

Comments should not be located at the end of lines of code

Code Smell

Loops should not contain more than a single "break" or "continue" statement

Code Smell

Variable, property and parameter names should comply with a naming convention

Code Smell

Lines should not end with trailing whitespaces

Code Smell

Arithmetic operators should only have numbers as operands

Analyze your code

Code Smell

Major

confusing

Expressions with arithmetic (`/`, `*`, `%`, `++`, `--`, `-`, `-=`, `*=`, `/=`, `%=`, `+=`, `+`), unary (`-`), or comparison operators (`>`, `<`, `>=`, `<=`) where one, or both, of the operands is a String, Boolean or Date value rely on implicit conversions. Both the maintainability and reliability levels of such a piece of code are questionable.

Noncompliant Code Example

```
str = "80";
quarter = str / 4; // Noncompliant

if (str < 10) { // Noncompliant
    // ...
}
```

Compliant Solution

```
str = "80";
parsedStr = parseInt(str);
quarter = parsedStr / 4;

if (parsedStr < 10) {
    // ...
}
```

Exceptions

- Expressions using the binary `+` operator with at least one String operand are ignored because the `+` operator will perform a concatenation in that case.
- Comparisons where both operands are strings are ignored because a lexicographical comparison is performed in that case.

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-3760

1/2

<div>Files should contain an empty newline at the end</div> <div> Code Smell</div>
<div>An open curly brace should be located at the end of a line</div> <div> Code Smell</div>
<div>Tabulation characters should not be used</div> <div> Code Smell</div>
<div>Function and method names should comply with a naming convention</div> <div> Code Smell</div>