

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

TypeScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your TYPESCRIPT code

All rules279

Vulnerability27

Bug51

Security Hotspot43

Code Smell158

Quick Fix50

Tags ▾

Search by name... 🔍

Code Smell

"strict" mode should be used with caution

Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply

"switch" statements should have "default" clauses

"if ... else if" constructs should end with "else" clauses

Control structures should use curly braces

String literals should not be duplicated

Expressions should not be too complex

Template literal placeholder syntax should not be used in regular strings

Built-in objects should not be overridden

"for...in" loops should filter properties before acting on them

Results of operations on strings should not be ignored

"in" should not be used on arrays

Analyze your code

Code Smell

Major

Quick Fix

pitfall

The in operator used on an array is valid but the code will likely not have the expected behavior. The in operator deals with the indexes of the array, not with the values.

If checking for an array slot is indeed desired, using hasOwnProperty makes the code intention clearer.

Noncompliant Code Example

```
function func1() {
  let arr = ["a", "b", "c"];

  let expectedValue = "b";
  if (expectedValue in arr) { // Noncompliant, will be always true
    return expectedValue + " found in the array";
  } else {
    return expectedValue + " not found";
  }
}

function func2() {
  let arr = ["a", "b", "c"];

  let expectedValue = "1"; // index #1 is corresponding to "b"
  if (expectedValue in arr) { // Noncompliant, will be always true
    return expectedValue + " found in the array";
  } else {
    return expectedValue + " not found";
  }
}

```

Compliant Solution

```
function func() {
  let arr = ["a", "b", "c"];

  let expectedValue = "b";
  if (arr.includes(expectedValue)) {
    return expectedValue + " was found in the array";
  } else {
    return expectedValue + " not found";
  }
}

```

Available In:

sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/typescript/RSPEC-4619

1/2

Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression

 Code Smell


Optional boolean parameters should have default value

 Code Smell

Union types should not have too many elements

 Code Smell

Dependencies should be explicit

 Code Smell

© 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](#)