




 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 rules 279

Vulnerability 27

Bug 51

Security Hotspot 43

Code Smell 158

Quick Fix 50

Tags ▾

Search by name... 🔍

Multiline blocks should be enclosed in curly braces

Code Smell

Boolean expressions should not be gratuitous

Code Smell

Variables should be used in the blocks where they are declared

Code Smell

Parameters should be passed in the correct order

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

Unused assignments should be removed

Code Smell

Function parameters with default values should be last

Code Smell

Functions should not be defined inside loops

Code Smell

"switch" statements should not have too many "case" clauses

Code Smell

Only "while", "do", "for" and "switch" statements should be labelled

Code Smell

Sections of code should not be commented out

Code Smell

Collection elements should not be replaced unconditionally

Analyze your code

Bug Major ? suspicious

It is highly suspicious when a value is saved for a key or index and then unconditionally overwritten. Such replacements are likely in error.

Noncompliant Code Example

```
fruits[1] = "banana";
fruits[1] = "apple"; // Noncompliant - value on index 1 is

myMap.set("key", 1);
myMap.set("key", 2); // Noncompliant - value for key "key"

mySet.add(1);
mySet.add(1); // Noncompliant - element is already in the s
```





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/typescript/RSPEC-4143

1/2

<div>Unused function parameters should be removed</div> <div> Code Smell</div>
<div>Track uses of "FIXME" tags</div> <div> Code Smell</div>
<div>Assignments should not be made from within sub-expressions</div> <div> Code Smell</div>
<div>Labels should not be used</div> <div> Code Smell</div>
<div>Variables should not be shadowed</div>