

































-  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


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




Unused assignments should be removed




Function parameters with default values should be last




Functions should not be defined inside loops




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




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




Sections of code should not be commented out




Unused function parameters should be removed




Track uses of "FIXME" tags



Assignments should not be made from within sub-expressions






Labels should not be used



Variables should not be shadowed

Collection sizes and array length comparisons should make sense

Analyze your code

 Bug  Major  Quick Fix

The size of a collection and the length of an array are always greater than or equal to zero. So testing that a size or length is greater than or equal to zero doesn't make sense, since the result is always `true`. Similarly testing that it is less than zero will always return `false`. Perhaps the intent was to check the non-emptiness of the collection or array instead.

Noncompliant Code Example

```
if (someSet.size >= 0) {...} // Noncompliant

if (someMap.size < 0) {...} // Noncompliant

const result = someArray.length >= 0; // Noncompliant
```




Compliant Solution

```
if (someSet.size > 0) {...}

if (someMap.size == 0) {...}

const result = someArray.length > 0;
```





Available In:

 |  | 

© 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-3981

1/2

<div>variables should not be shadowed</div> <div> Code Smell</div>
<div>Redundant pairs of parentheses should be removed</div> <div> Code Smell</div>
<div>Nested blocks of code should not be left empty</div> <div> Code Smell</div>
<div>Functions should not have too many parameters</div> <div> Code Smell</div>
<div>OS commands should not be vulnerable to argument injection</div>