




 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... 🔍


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

 Code Smell


"switch" statements should have "default" clauses

 Code Smell


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

 Code Smell


Control structures should use curly braces

 Code Smell


String literals should not be duplicated

 Code Smell


Expressions should not be too complex

 Code Smell


Template literal placeholder syntax should not be used in regular strings

 Bug


Built-in objects should not be overridden

 Bug


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

 Bug

Results of operations on strings should not be ignored

 Bug


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

Assignments should not be redundant

Analyze your code

 Code Smell

 Major ?

 redundant

The transitive property says that if `a == b` and `b == c`, then `a == c`. In such cases, there's no point in assigning `a` to `c` or vice versa because they're already equivalent.

This rule raises an issue when an assignment is useless because the assigned-to variable already holds the value on all execution paths.




Noncompliant Code Example

```
a = b;
c = a;
b = c; // Noncompliant: c and b are already the same
```

Compliant Solution

```
a = b;
c = a;
```





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-4165

1/2

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
"this" should not be assigned to variables  Code Smell
The "any" type should not be used