




 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... 🔍

Template strings should be used instead of concatenation

Code Smell

Shorthand object properties should be grouped at the beginning or end of an object declaration

Code Smell

Object literal shorthand syntax should be used

Code Smell

Strings and non-strings should not be added

Code Smell

Primitive types should be omitted from initialized or defaulted declarations

Code Smell

Non-null assertions should not 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

Labels should not be used

Analyze your code

Code Smell

Major ?

confusing

Labels are not commonly used, and many developers do not understand how they work. Moreover, their usage makes the control flow harder to follow, which reduces the code's readability.

Noncompliant Code Example

```
myLabel: {
  let x = doSomething();
  if (x > 0) {
    break myLabel;
  }
  doSomethingElse();
}
```

Compliant Solution

```
let x = doSomething();
if (x <= 0) {
  doSomethingElse();
}
```





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-1119

1/2

 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
Files should contain an empty newline at the end