

sonar


RULES


Products


▼


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

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

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

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

Analyze your code

Code Smell

Critical

brain-overload

Nested if, for, while, switch, and try statements are key ingredients for making what's known as "Spaghetti code".

Such code is hard to read, refactor and therefore maintain.

Noncompliant Code Example

With the default threshold of 3:

```
if (condition1) {                                // Compliant - depth =
/* ... */
  if (condition2) {                                // Compliant - depth =
/* ... */
    for(let i = 0; i < 10; i++) { // Compliant - depth =
/* ... */
      if (condition4) {                                // Non-Compliant - dept
        if (condition5) {                                // Depth = 5, exceeding
/* ... */
          }
        return;
      }
    }
  }
}
```

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-134

1/2

<div>Files should contain an empty newline at the end</div> <div> Code Smell</div>
<div>An open curly brace should be located at the end of a line</div> <div> Code Smell</div>
<div>Tabulation characters should not be used</div> <div> Code Smell</div>
<div>Function and method names should comply with a naming convention</div> <div> Code Smell</div>