




 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... 🔍


 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

Functions should not have too many parameters

Analyze your code

 Code Smell

 Major ?

 brain-overload

A long parameter list can indicate that a new structure should be created to wrap the numerous parameters or that the function is doing too many things.

Noncompliant Code Example




With a maximum number of 4 parameters:

```
function doSomething(param1, param2, param3, param4, param5)
...
}
```

Compliant Solution

```
function doSomething(param1, param2, param3, param4) {
...
}
```

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-107

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>