




 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... 🔍

Comparison operators should not be used with strings

Code Smell

Private properties that are only assigned in the constructor or at declaration should be "readonly"

Code Smell

Property getters and setters should come in pairs

Code Smell

JavaScript parser failure

Code Smell

The ternary operator should not be used

Code Smell

"===" and "!==" should be used instead of "==" and "!="

Code Smell

Functions should not have too many lines of code

Code Smell

Track comments matching a regular expression

Code Smell

Statements should be on separate lines

Code Smell

Magic numbers should not be used

Code Smell

Collapsible "if" statements should be merged

Code Smell

Standard outputs should not be used directly to log anything

Code Smell

"arguments.caller" and "arguments.callee" should not be used

Analyze your code

Code Smell

Major

obsolete

Both arguments.caller and arguments.callee make quite a few optimizations impossible so they were deprecated in latest versions of JavaScript. In fact, EcmaScript 5 forbids the use of both in strict mode, according to the docs:

Arguments objects for strict mode functions define non-configurable accessor properties named "caller" and "callee" which throw a TypeError exception on access.

The same restriction applies to the function's caller and arguments properties in strict mode.

Noncompliant Code Example

```
function whoCalled() {
  if (arguments.caller == null) //Noncompliant
    console.log('I was called from the global scope.');
```

```
  else
    console.log(arguments.caller + ' called me!'); // Noncompliant

  console.log(whoCalled.caller); // Noncompliant
  console.log(whoCalled.arguments); // Noncompliant
}
```





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-2685

1/2

 Code Smell
Files should not have too many lines of code  Code Smell
Lines should not be too long  Code Smell
Debugger statements should not be used  Vulnerability
Regular expressions using Unicode character classes or property escapes should enable the unicode flag