




 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




Non-null assertions should not be used




"undefined" should not be assigned




Trailing commas should not be used




Array constructors should not be used




Quotes for string literals should be used consistently




Statements should end with semicolons




Comments should not be located at the end of lines of code




Loops should not contain more than a single "break" or "continue" statement





Variable, property and parameter names should comply with a naming convention



Lines should not end with trailing whitespaces



Debugger statements should not be used

 Vulnerability Minor ⓘ

The debugger statement can be placed anywhere in procedures to suspend execution. Using the debugger statement is similar to setting a breakpoint in the code. By definition such statement must absolutely be removed from the source code to prevent any unexpected behavior or added vulnerability to attacks in production.

Noncompliant Code Example

```
for (i = 1; i<5; i++) {
  // Print i to the Output window.
  Debug.write("loop index is " + i);
  // Wait for user to resume.
  debugger;
}
```

Compliant Solution

```
for (i = 1; i<5; i++) {
  // Print i to the Output window.
  Debug.write("loop index is " + i);
}
```




See

- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- MITRE, CWE-489 - Active Debug Code

Deprecated

This rule is deprecated; use {rule:javascript:S4507} instead.

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-1525

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>