




 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... 🔍

added
<div>Code Smell</div>
Primitive types should be omitted from initialized or defaulted declarations
<div>Code Smell</div>
Non-null assertions should not be used
<div>Code Smell</div>
"undefined" should not be assigned
<div>Code Smell</div>
Trailing commas should not be used
<div>Code Smell</div>
Array constructors should not be used
<div>Code Smell</div>
Quotes for string literals should be used consistently
<div>Code Smell</div>
Statements should end with semicolons
<div>Code Smell</div>
Comments should not be located at the end of lines of code
<div>Code Smell</div>
Loops should not contain more than a single "break" or "continue" statement
<div>Code Smell</div>
Variable, property and parameter names should comply with a naming convention
<div>Code Smell</div>
Lines should not end with trailing whitespaces
<div>Code Smell</div>

Braces and parentheses should be used consistently with arrow functions

Analyze your code

Code Smell

Minor ?

convention es2015

Shared coding conventions allow teams to collaborate effectively. This rule raises an issue when the use of parentheses with an arrow function does not conform to the configured requirements.

Noncompliant Code Example

With the configured defaults forbidding parentheses

```
var foo = (a) => { /* ... */ }; // Noncompliant; remove parentheses
var bar = (a, b) => { return 0; }; // Noncompliant; remove parentheses
```





Compliant Solution

```
var foo = a => { /* ... */ };
var bar = (a, b) => 0;
```

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](#)

<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>