




 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... 🔍


Property getters and setters should come in pairs




JavaScript parser failure




The ternary operator should not be used




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




Functions should not have too many lines of code




Track comments matching a regular expression




Statements should be on separate lines




Magic numbers should not be used




Collapsible "if" statements should be merged




Standard outputs should not be used directly to log anything



Files should not have too many lines of code





Lines should not be too long




Multiline blocks should be enclosed in curly braces

Analyze your code

 Code Smell

 Major

 cwe

Curly braces can be omitted from a one-line block, such as with an `if` statement or for loop, but doing so can be misleading and induce bugs.

This rule raises an issue when the whitespacing of the lines after a one line block indicates an intent to include those lines in the block, but the omission of curly braces means the lines will be unconditionally executed once.

Note that this rule considers tab characters to be equivalent to 1 space. If you mix spaces and tabs you will sometimes see issues in code which looks fine in your editor but is confusing when you change the size of tabs.

Noncompliant Code Example

```
if (condition)
  firstActionInBlock();
  secondAction(); // Noncompliant; executed unconditionally
  thirdAction();

if (condition) firstActionInBlock(); secondAction(); // Non

if (condition) firstActionInBlock(); // Noncompliant
  secondAction(); // Executed unconditionally

if (condition); secondAction(); // Noncompliant; secondActi

let str = undefined;
for (let i = 0; i < array.length; i++)
  str = array[i];
doTheThing(str); // Noncompliant; executed only on last a
```

Compliant Solution

```
if (condition) {
  firstActionInBlock();
  secondAction();
}
thirdAction();

let str = undefined;
for (let i = 0; i < array.length; i++) {
  str = array[i];
  doTheThing(str);
}
```








See

- [MITRE, CWE-483](#) - Incorrect Block Delimitation

Available In:

https://rules.sonarsource.com/typescript/RSPEC-2681

1/2

<div>Debugger statements should not be used</div> <div> Vulnerability</div>	<div>sonarlint    sonarcloud    sonarqube </div> <div>© 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. <a href="#">Privacy Policy</a></div>
<div>Regular expressions using Unicode character classes or property escapes should enable the unicode flag</div> <div> Bug</div>	
<div>The base should be provided to "parseInt"</div> <div> Bug</div>	
<div>Function declarations should not be made within blocks</div> <div> Bug</div>	