




 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... 


braces and parentheses should be used consistently with arrow functions
 Code Smell


Destructuring syntax should be used for assignments
 Code Smell


Template strings should be used instead of concatenation
 Code Smell


Shorthand object properties should be grouped at the beginning or end of an object declaration
 Code Smell


Object literal shorthand syntax should be used
 Code Smell


Strings and non-strings should not be added
 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

Assignments should not be made from within sub-expressions

Analyze your code

 Code Smell  Major  cwe suspicious

Assignments within sub-expressions are hard to spot and therefore make the code less readable. Ideally, sub-expressions should not have side-effects.

Moreover, using chained assignment in declarations is also dangerous because one may accidentally create global variables, e.g. in `let x = y = 1;`, if `y` is not declared, it will be hoisted as global.

Noncompliant Code Example

```
if (val = value() && check()) { // Noncompliant
  // ...
}
```

Compliant Solution

```
val = value();
if (val && check()) {
  // ...
}
```

Exceptions




The rule does not raise issues for the following patterns:

- chained assignments: `a = b = c = 0;`
- relational assignments: `(a = 0) != b`
- sequential assignments: `a = 0, b = 1, c = 2`
- assignments in lambda body: `() => a = 0`
- conditional assignment idiom: `a || (a = 0)`
- assignments in (do-)while conditions: `while (a = 0);`

See

- [MITRE, CWE-481](#) - Assigning instead of Comparing





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-1121

1/2

 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