




 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... 🔍


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


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

 Bug


The base should be provided to "parseInt"

 Bug


Function declarations should not be made within blocks

 Bug

Writing cookies is security-sensitive



 Security Hotspot

"continue" should not be used

 Code Smell

Primitive return types should be used

Parameters should be passed in the correct order

 Code Smell  Major ?

When the names of arguments in a function call match the names of the function parameters, it contributes to clearer, more readable code. However, when the names match, but are passed in a different order than the function parameters, it indicates a mistake in the parameter order which will likely lead to unexpected results.

Noncompliant Code Example

```
function divide(divisor, dividend) {
    return divisor/dividend;
}

function doTheThing() {
    var divisor = 15;
    var dividend = 5;

    var result = divide(dividend, divisor); // Noncompliant;
    //...
}
```

Compliant Solution

```
function divide(divisor, dividend) {
    return divisor/dividend;
}

function doTheThing() {
    var divisor = 15;
    var dividend = 5;

    var result = divide(divisor, dividend);
    //...
}
```

Exceptions


Swapped arguments that are compared beforehand in an enclosing if-statement are ignored:

```
function divide(divisor, dividend) {
    return divisor/dividend;
}


function doTheThing() {
    var divisor = 15;
    var dividend = 5;
    if (divisor > dividend) {
        var result = divide(dividend, divisor);
        //...
    }
}
```

https://rules.sonarsource.com/typescript/RSPEC-2234


1/2

 Code Smell


Default type parameters should be omitted

 Code Smell


Type assertions should use "as"

 Code Smell

Method overloads should be grouped together




 Code Smell

Interfaces should not be empty

 Code Smell

```
}  
}
```

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](#)