




 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... 🔍

have default value

Code Smell

Union types should not have too many elements

Code Smell

Dependencies should be explicit

Code Smell

"this" should not be assigned to variables

Code Smell

The "any" type should not be used

Code Smell

"for in" should not be used with iterables

Code Smell

Functions should use "return" consistently

Code Smell

"arguments" should not be accessed directly

Code Smell

Comparison operators should not be used with strings

Code Smell

Private properties that are only assigned in the constructor or at declaration should be "readonly"

Code Smell

Property getters and setters should come in pairs

Code Smell

JavaScript parser failure

Code Smell

Assertion arguments should be passed in the correct order

Analyze your code

Code Smell

Major

Quick Fix

tests chai suspicious

Many assertion functions have specific parameters for the expected and actual values. Swap them, and your test will still have the same outcome (succeed/fail when it should) but the error messages will be confusing.

This rule raises an issue when the "expected" argument of an assertion function is a hard-coded value and the "actual" argument is not.

This rule currently supports Chai assertions.

Noncompliant Code Example

```
const assert = require('chai').assert;
const expect = require('chai').expect;
const should = require('chai').should();

it("inverts arguments", function() {
  assert.equal(42, aNumber); // Noncompliant
  expect(42).to.equal(aNumber); // Noncompliant
  should.fail(42, aNumber); // Noncompliant
});
```

Compliant Solution

```
const assert = require('chai').assert;
const expect = require('chai').expect;
const should = require('chai').should();

it("inverts arguments", function() {
  assert.equal(aNumber, 42);
  expect(aNumber).to.equal(42);
  should.fail(aNumber, 42);
});
```

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](#)

https://rules.sonarsource.com/typescript/RSPEC-3415

1/2

<div>The ternary operator should not be used</div> <div> Code Smell</div>
<div>"===" and "!===" should be used instead of "==" and "!="</div> <div> Code Smell</div>
<div>Functions should not have too many lines of code</div> <div> Code Smell</div>
<div>Track comments matching a regular expression</div> <div> Code Smell</div>