




 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

Assignments should not be redundant
Code Smell
Functions should not have identical implementations
Code Smell
Sparse arrays should not be declared
Code Smell
Array-mutating methods should not be used misleadingly
Code Smell
Collection and array contents should be used
Code Smell
Literals should not be thrown
Code Smell
Array indexes should be numeric
Code Smell
Assertion arguments should be passed in the correct order
Code Smell
Ternary operators should not be nested
Code Smell
"delete" should not be used on arrays
Code Smell
Variables and functions should not be redeclared
Code Smell
"indexOf" checks should not be for positive numbers
Code Smell
"arguments.caller" and

Tags ▾

Search by name... 🔍

Assertions should not be given twice the same argument

Analyze your code

Bug

Major

tests

Many assertions compare two objects or properties of these objects. Passing twice the same argument is likely to be a bug due to developer's carelessness.

This rule raises an issue when a Chai assertion is given twice the same argument.

Noncompliant Code Example

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

describe("test the same object", function() {
  it("uses chai 'assert'", function() {
    const expected = '1'
    const actual = (1).toString()
    assert.equal(actual, actual); // Noncompliant
  });
});
```





Compliant Solution

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

describe("test the same object", function() {
  it("uses chai 'assert'", function() {
    const expected = '1'
    const actual = (1).toString()
    assert.equal(actual, expected);
  });
});
```

Available In:

sonarlint | sonarcloud | sonarqube

<div>arguments.callee and "arguments.callee" should not be used</div> <div> Code Smell</div>
<div>Multiline blocks should be enclosed in curly braces</div> <div> Code Smell</div>
<div>Boolean expressions should not be gratuitous</div> <div> Code Smell</div>
<div>Variables should be used in the blocks where they are declared</div> <div> Code Smell</div>
<div>Parameters should be passed in the</div>