

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

**TypeScript**

T-SQL

VB.NET

VB6

XML

TS

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...

File uploads should be restricted

Vulnerability

Regular expressions should be syntactically valid

Bug

Types without members, 'any' and 'never' should not be used in type intersections

Bug

Getters and setters should access the expected fields

Bug

"super()" should be invoked appropriately

Bug

Results of "in" and "instanceof" should be negated rather than operands

Bug

A compare function should be provided when using "Array.prototype.sort()"

Bug

Jump statements should not occur in "finally" blocks

Bug

Using slow regular expressions is security-sensitive

Security Hotspot

Using publicly writable directories is security-sensitive

Security Hotspot

Using clear-text protocols is security-sensitive

Security Hotspot

Expanding archive files without

Assertions should be complete

Analyze your code

Code Smell

Blocker

tests

It is very easy to write incomplete assertions when using some test frameworks. This rule enforces complete Chai assertions in the following cases:

when `assert.fail`, `expect.fail` or `should.fail` are present but not called.

when an `expect(...)` or `should` assertion is not followed by an assertion method, such as `equal`.

when an `expect` or `should` assertion ends with a **chainable getters**, such as `.that`, or a modifier, such as `.deep`.

when an `expect` or `should` assertion function, such as `.throw`, is not called.

In such cases, what is intended to be a test doesn't actually verify anything

Noncompliant Code Example

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

describe("incomplete assertions", function() {
  const value = 42;

  it("uses chai 'assert'", function() {
    assert.fail; // Noncompliant
  });

  it("uses chai 'expect'", function() {
    expect(1 == 1); // Noncompliant
    expect(value.toString()).to.throw; // Noncompliant
  });
});
```

Compliant Solution








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

describe("incomplete assertions", function() {
  const value = 42;

  it("uses chai 'assert'", function() {
    assert.fail();
  });

  it("uses chai 'expect'", function() {
    expect(1).to.equal(1);
    expect(value.toString()).throw(TypeError);
  });
});
```

https://rules.sonarsource.com/typescript/RSPEC-29701/2

<p><b>controlling resource consumption is security-sensitive</b></p> <p> Security Hotspot</p>	<p>Available In:</p> <p><b>sonarlint</b>    <b>sonarcloud</b>    <b>sonarqube</b> </p>
<p><b>Using weak hashing algorithms is security-sensitive</b></p> <p> Security Hotspot</p>	
<p><b>Disabling CSRF protections is security-sensitive</b></p> <p> Security Hotspot</p>	
<p><b>Using pseudorandom number generators (PRNGs) is security-sensitive</b></p> <p> Security Hotspot</p>	

© 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](#)