

Secrets

ABAP

Apex

C

C++

CloudFormation

COBOL

C#

CSS

Flex

Go

HTML

Java

JS

JavaScript

Kotlin

Objective C

PHP

PL/I

PL/SQL

Python

RPG

Ruby

Scala

Swift

Terraform

Text

TypeScript

T-SQL

VB.NET

VB6

XML

JS

JavaScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVASCRIPT code

All rules285

Vulnerability29

Bug62

Security Hotspot43

Code Smell151

Quick Fix41

Tags

Search by name...

Vulnerability

File uploads should be restricted

Vulnerability

Function calls should not pass extra arguments

Bug

Regular expressions should be syntactically valid

Bug

Getters and setters should access the expected fields

Bug

"super()" should be invoked appropriately

Bug

"Symbol" should not be used as a constructor

Bug

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

Bug

"in" should not be used with primitive types

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

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()).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/javascript/RSPEC-2970

1/2

<p>Using publicly writable directories is security-sensitive</p> <p> Security Hotspot</p>	<p>Available In:</p> <p>  </p> <hr/> <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</p>
<p>Using clear-text protocols is security-sensitive</p> <p> Security Hotspot</p>	
<p>Expanding archive files without controlling resource consumption is security-sensitive</p> <p> Security Hotspot</p>	
<p>Using weak hashing algorithms is security-sensitive</p> <p> Security Hotspot</p>	