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

JavaScript static code analysis

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

All rules 285 | Vulnerability 29 | Bug 62 | Security Hotspot 43 | Code Smell 151 | Quick Fix 41

Tags | Search by name...

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

Using publicly writable directories is security-sensitive
Security Hotspot

Using clear-text protocols is security-sensitive
Security Hotspot

Expanding archive files without controlling resource consumption is security-sensitive

Tests should include assertions

Analyze your code

Code Smell | Blocker | tests chai mocha

A test case without assertions ensures only that no exceptions are thrown. Beyond basic runnability, it ensures nothing about the behavior of the code under test.

This rule raises an exception when the assertion library `chai` is imported but no assertion is used in a test.

Noncompliant Code Example

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

describe("No assertion", function() {
    it("doesn't test anything", function() { // Noncompliant
        const str = "";
    });
});
```

Compliant Solution

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

describe("Has assertions", function() {
    it("tests a string", function() {
        const str = "";
        expect(str).to.be.a('string');
    });
});
```

Available In:

sonarlint | sonarcloud | sonarqube

Security Hotspot

Using weak hashing algorithms is security-sensitive

Security Hotspot

Disabling CSRF protections is security-sensitive

Security Hotspot

Using pseudorandom number generators (PRNGs) is security-sensitive

Security Hotspot

Dynamically executing code is