




 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

Tags ▾

Search by name... 🔍

	Code Smell
	Web SQL databases should not be used
	Variables declared with "var" should be declared before they are used
	Track lack of copyright and license headers
	Reading the Standard Input is security-sensitive
	Using command line arguments is security-sensitive
	Using Sockets is security-sensitive
	Executing XPath expressions is security-sensitive
	Encrypting data is security-sensitive
	Using regular expressions is security-sensitive
	Class methods should be used instead of "prototype" assignments
	Variables should be declared with "let" or "const"
	Unchanged variables should be

Tests should check which exception is thrown

Analyze your code

 Code Smell

 Major ?

 tests chai mocha

It is not good enough to test if an exception is raised, without checking which exception it is. Such tests will not be able to differentiate the expected exception from an unexpected one. They should instead validate the exception message and/or type.

This rule raises an issue in the following cases:

- When an asynchronous Mocha test calls the `done()` callback, without parameters, in a `catch` block and there is no reference to the caught exception in this block. Either the error should be passed to `done()` or the exception should be checked further.
- When Chai assertions are used to test if a function throws any exception, or an exception of type `Error` without checking the message.
- When Chai assertions are used to test if a function does not throw an exception of type `Error` without checking the message.

Rule doesn't raise an issue when assertion is negated using `not`, in such case the exception doesn't need to be specific.

Noncompliant Code Example

```
const expect = require("chai").expect;
const fs = require("fs");





describe("exceptions are not tested properly", function() {
  const funcThrows = function () { throw new TypeError('Wh
  const funcNoThrow = function () { /*noop*/ };

  it("forgot to pass the error to 'done()", function(done) {
    fs.readFile("/etc/zshrc", 'utf8', function(err, data) {
      try {
        expect(data).to.match(/some expected string/);
      } catch (e) { // Noncompliant
        // Either the exception should be passed to done();
      }
    });
  });

  it("does not 'expect' a specific exception", function() {
    expect(funcThrows).to.throw(); // Noncompliant
    // Error is not precise enough
    expect(funcThrows).to.throw(Error); // Noncompliant
  });
});
```

Compliant Solution

```
const expect = require("chai").expect;
const { AssertionError } = require('chai');
```

marked const  Code Smell
Wildcard imports should not be used  Code Smell
"switch" statements should not be nested  Code Smell
Cyclomatic Complexity of functions should not be too high  Code Smell
"strict" mode should be used with caution

```
const fs = require("fs");

describe("exceptions are tested properly", function() {
  const funcThrows = function () { throw new TypeError('Wh
  const funcNoThrow = function () { /*noop*/ };

  it("forgot to pass the error to 'done()'", function(done) {
    fs.readFile("/etc/zshrc", 'utf8', function(err, data) {
      try {
        expect(data).toMatch(/some expected string/
      } catch (e) {
        expect(e).toBe.an.instanceof(AssertionError
        done();
      }
    });
  });

  it("does not 'expect' a specific exception", function() {
    expect(funcThrows).toThrow(TypeError);
    expect(funcNoThrow).to.not.throw(Error, /My error me
  });
});
```

Available In:
 |  | 