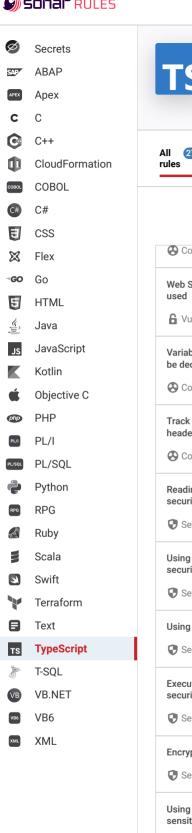
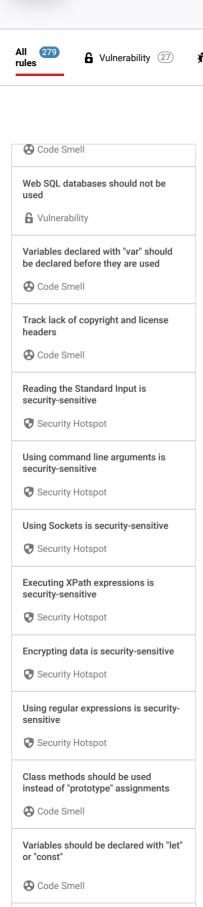
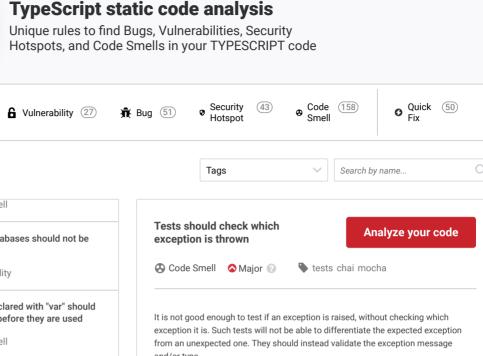


Products ✓





Unchanged variables should be



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, it 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');
```

TypeScript static code
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
                expect(data).to.match(/some expected string/
            } catch (e) {
                expect(e).to.be.an.instanceof(AssertionError
                done();
        });
    });
    it("does not 'expect' a specific exception", function()
        expect(funcThrows).to.throw(TypeError);
        expect(funcNoThrow).to.not.throw(Error, /My error me
    });
});
Available In:
```

© 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

sonarlint ⊕ | sonarcloud 🐼 | sonarqube