Security

Hotspot

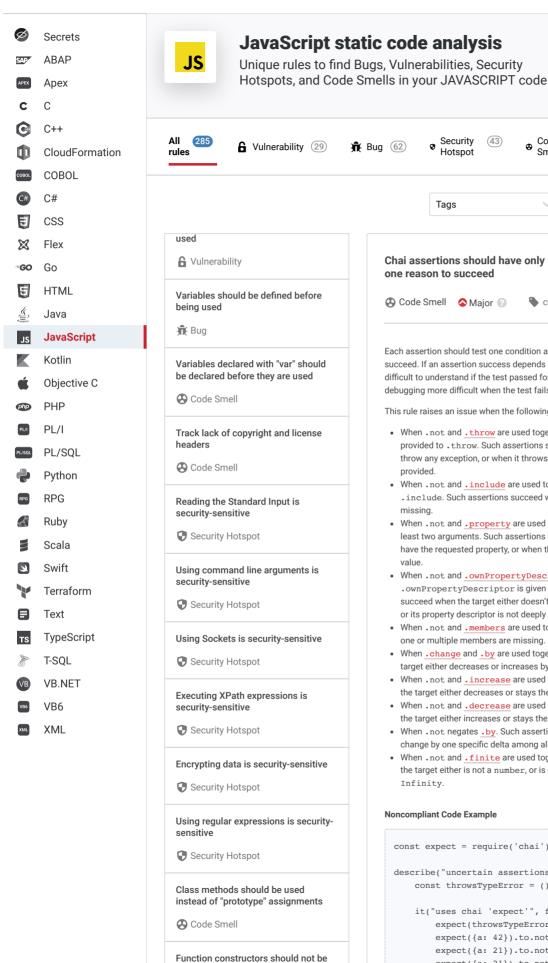
(43)

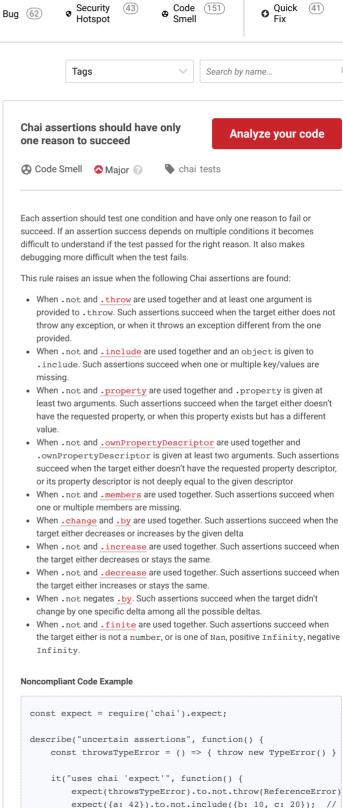


Products >

Quick 41 Fix

(151)





expect({a: 21}).to.not.have.property('b', 42); // No

expect({a: 21}).to.not.have.ownPropertyDescriptor('b

configurable: true, enumerable: true,

writable: true,

used

Code Smell

Variables should be declared with "let" or "const" Code Smell Unchanged variables should be marked "const" Code Smell Wildcard imports should not be used Code Smell "switch" statements should not be nested Code Smell

Cyclomatic Complexity of functions

```
value: 42,
       });
       expect([21, 42]).to.not.have.members([1, 2]); // Non
       var myObj = { value: 1 }
       const incThree = () => { myObj.value += 3; };
       const decThree = () => { myObj.value -= 3; };
       const doNothing = () => {};
       expect(incThree).to.change(myObj, 'value').by(3); //
       expect(decThree).to.change(myObj, 'value').by(3); //
       expect(decThree).to.not.increase(myObj, 'value'); //
       expect(incThree).to.not.decrease(myObj, 'value'); //
        expect(doNothing).to.not.increase(myObj, 'value'); /
       expect(doNothing).to.not.decrease(myObj, 'value'); /
       expect(incThree).to.increase(myObj, 'value').but.not
       let toCheck:
       expect(toCheck).to.not.be.finite; // Noncompliant
   });
});
```

## **Compliant Solution**

```
const expect = require('chai').expect:
describe("uncertain assertions", function() {
    const throwsTypeError = () => { throw new TypeError() }
    it("uses chai 'expect'", function() {
        expect(throwsTypeError).to.throw(TypeError)
        expect({a: 42}).to.not.have.any.keys('b', 'c');
        expect({a: 21}).to.not.have.property('b');
        expect({a: 21}).to.not.have.ownPropertyDescriptor('b
        expect([21, 42]).to.not.include(1).and.not.include(2
        var myObj = { value: 1 }
        const incThree = () => { myObj.value += 3; };
        const decThree = () => { myObj.value -= 3; };
        const doNothing = () => {};
        expect(incThree).to.increase(myObj, 'value').by(3);
        expect(decThree).to.decrease(myObj, 'value').by(3);
        expect(decThree).to.decrease(myObj, 'value').by(3);
        expect(incThree).to.increase(myObj, 'value').by(3);
        expect(doNothing).to.not.change(myObj, 'value');
        expect(incThree).to.increase(myObj, 'value').by(3);
        let toCheck;
        // Either of the following is valid
        expect(toCheck).to.be.a('string');
        expect(toCheck).to.be.NaN;
        expect(toCheck).to.equal(Infinity);
        expect(toCheck).to.equal(-Infinity);
   });
});
```

Available In:

sonarlint ⊕ | sonarcloud ♦ | sonarqube



© 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