




 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... 🔍

Files should not have too many lines of code

Code Smell

Track comments matching a regular expression

Code Smell

Statements should be on separate lines

Code Smell

Magic numbers should not be used

Code Smell

Collapsible "if" statements should be merged

Code Smell

Standard outputs should not be used directly to log anything

Code Smell

Files should not have too many lines of code

Code Smell

Lines should not be too long

Code Smell

Debugger statements should not be used

Vulnerability

"alert(...)" should not be used

Vulnerability

Regular expressions using Unicode character classes or property escapes should enable the unicode flag

Bug

The base should be provided to "parseInt"

Bug

"indexOf" checks should not be for positive numbers

Analyze your code

Code Smell

Major

suspicious

Most checks against an `indexOf` call against an array compare it with `-1` because `0` is a valid index. Any checks which look for values `>0` ignore the first element, which is likely a bug. If you're merely checking the presence of the element, consider using `includes` instead. Before using `includes` method make sure that your browser version is supporting it.

Noncompliant Code Example

```
var color = "blue";
var name = "ishmael";

var arr = [color, name];

if (arr.indexOf("blue") > 0) { // Noncompliant
  // ...
}
```

Compliant Solution

```
var color = "blue";
var name = "ishmael";

var arr = [color, name];

if (arr.indexOf("blue") >= 0) {
  // ...
}
if (arr.includes("blue")) {
  // ...
}
```

See

[Array.prototype.includes\(\)](#) documentation at MDN

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](#)

https://rules.sonarsource.com/javascript/RSPEC-2692

1/2

<div>Function declarations should not be made within blocks</div> <div> Bug</div>
<div>Writing cookies is security-sensitive</div> <div> Security Hotspot</div>
<div>"continue" should not be used</div> <div> Code Smell</div>
<div>Trailing commas should be used</div> <div> Code Smell</div>
<div>"import" should be used to include external code</div>