




 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... 🔍

Object literal shorthand syntax should be used

Code Smell

Strings and non-strings should not be added

Code Smell

Object literal syntax should be used

Code Smell

"undefined" should not be assigned

Code Smell

Trailing commas should not be used

Code Smell

Array constructors should not be used

Code Smell

Quotes for string literals should be used consistently

Code Smell

Statements should end with semicolons

Code Smell

Comments should not be located at the end of lines of code

Code Smell

Loops should not contain more than a single "break" or "continue" statement

Code Smell

Variable, property and parameter names should comply with a naming convention

Code Smell

Lines should not end with trailing whitespaces

Code Smell

"with" statements should not be used

Analyze your code

Bug Minor ?

The use of the with keyword produces an error in JavaScript strict mode code. However, that's not the worst that can be said against with.

Using with allows a short-hand access to an object's properties - assuming they're already set. But use with to access some property not already set in the object, and suddenly you're catapulted out of the object scope and into the global scope, creating or overwriting variables there. Since the effects of with are entirely dependent on the object passed to it, with can be dangerously unpredictable, and should never be used.

Noncompliant Code Example

```
var x = 'a';

var foo = {
  y: 1
}

with (foo) { // Noncompliant
  y = 4; // updates foo.y
  x = 3; // does NOT add a foo.x property; updates x var in
}
print(foo.x + " " + x); // shows: undefined 3
```

Compliant Solution

```
var x = 'a';

var foo = {
  y: 1
}

foo.y = 4;
foo.x = 3;

print(foo.x + " " + x); // shows: 3 a
```





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

https://rules.sonarsource.com/javascript/RSPEC-1321

1/2

<div>Files should contain an empty newline at the end</div> <div> Code Smell</div>
<div>An open curly brace should be located at the end of a line</div> <div> Code Smell</div>
<div>Tabulation characters should not be used</div> <div> Code Smell</div>
<div>Function and method names should comply with a naming convention</div> <div> Code Smell</div>