




 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... 🔍

Braces and parentheses should be used consistently with arrow functions

Code Smell

Destructuring syntax should be used for assignments

Code Smell

Template strings should be used instead of concatenation

Code Smell

Shorthand object properties should be grouped at the beginning or end of an object declaration

Code Smell

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

Functions should not be defined inside loops

Analyze your code

Code Smell

Major ?




suspicious

Defining a function inside of a loop can yield unexpected results. Such a function keeps references to the variables which are defined in outer scopes. All function instances created inside the loop therefore see the same values for these variables, which is probably not expected.

Noncompliant Code Example

```
var funs = [];  
for (var i = 0; i < 13; i++) {  
  funs[i] = function() { // Non-Compliant  
    return i;  
  };  
}  
console.log(funs[0]()); // 13 instead of 0  
console.log(funs[1]()); // 13 instead of 1  
console.log(funs[2]()); // 13 instead of 2  
console.log(funs[3]()); // 13 instead of 3  
...
```




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-1515

1/2

<div>Comments should not be located at the end of lines of code</div> <div> Code Smell</div>
<div>Loops should not contain more than a single "break" or "continue" statement</div> <div> Code Smell</div>
<div>Variable, property and parameter names should comply with a naming convention</div> <div> Code Smell</div>
<div>Lines should not end with trailing whitespaces</div>