




 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 rules285

 Vulnerability29

 Bug62

 Security Hotspot43


 Code Smell151

 Quick Fix41


Tags ▾

Search by name... 🔍


Statements should be labeled

 Code Smell


Sections of code should not be commented out

 Code Smell


Unused function parameters should be removed

 Code Smell


Track uses of "FIXME" tags

 Code Smell


Assignments should not be made from within sub-expressions

 Code Smell


Labels should not be used

 Code Smell


Variables should not be shadowed

 Code Smell


Redundant pairs of parentheses should be removed

 Code Smell


Nested blocks of code should not be left empty

 Code Smell


Functions should not have too many parameters

 Code Smell

OS commands should not be vulnerable to argument injection attacks


 Vulnerability


Repeated patterns in regular expressions should not match the empty string


 Bug

Generators should "yield" something

Analyze your code

 Bug

 Major

 api-design

es2015

A generator without a `yield` statement is at best confusing, and at worst a bug in your code, since the iterator produced by your code will always be empty.




Noncompliant Code Example

```
function* myGen(a, b) { // Noncompliant
  let answer = 0;
  answer += a * b;
}
```

Compliant Solution

```
function* myGen(a, b) {
  let answer = 0;
  while (answer < 42) {
    answer += a * b;
    yield answer;
  }
}
```

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](#)

https://rules.sonarsource.com/javascript/RSPEC-3531

1/2

<div>Empty collections should not be accessed or iterated</div> <div> Bug</div>
<div>"delete" should be used only with object properties</div> <div> Bug</div>
<div>"with" statements should not be used</div> <div> Bug</div>
<div>Function parameters, caught exceptions and foreach variables' initial values should not be ignored</div> <div> Bug</div>