




 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


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


Function declarations should not be made within blocks

 Bug


Writing cookies is security-sensitive

 Security Hotspot


"continue" should not be used

 Code Smell


Trailing commas should be used

 Code Smell


"import" should be used to include external code

 Code Smell


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

Parameters should be passed in the correct order

Analyze your code

 Code Smell

 Major

When the names of arguments in a function call match the names of the function parameters, it contributes to clearer, more readable code. However, when the names match, but are passed in a different order than the function parameters, it indicates a mistake in the parameter order which will likely lead to unexpected results.

Noncompliant Code Example

```
function divide(divisor, dividend) {
  return divisor/dividend;
}

function doTheThing() {
  var divisor = 15;
  var dividend = 5;

  var result = divide(dividend, divisor); // Noncompliant;
  //...
}
```

Compliant Solution

```
function divide(divisor, dividend) {
  return divisor/dividend;
}

function doTheThing() {
  var divisor = 15;
  var dividend = 5;

  var result = divide(divisor, dividend);
  //...
}
```

Exceptions

Swapped arguments that are compared beforehand in an enclosing if-statement are ignored:

```
function divide(divisor, dividend) {
  return divisor/dividend;
}

function doTheThing() {
  var divisor = 15;
  var dividend = 5;
  if (divisor > dividend) {
    var result = divide(dividend, divisor);
    //...
  }
}
```

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

```
}  
}
```

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](#)