


































-  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... 🔍


Assertion arguments should be passed in the correct order  



Ternary operators should not be nested  



"delete" should not be used on arrays  



Variables and functions should not be redeclared  



"indexOf" checks should not be for positive numbers  



"arguments.caller" and "arguments.callee" should not be used  



Multiline blocks should be enclosed in curly braces  


Boolean expressions should not be gratuitous  


Variables should be used in the blocks where they are declared  




Parameters should be passed in the correct order  


Two branches in a conditional structure should not have exactly the same implementation  


Unused assignments should be removed  


## Promise rejections should not be caught by 'try' block

Analyze your code

 Bug  Major ?

An exception (including reject) thrown by a promise will not be caught by a nesting try block, due to the asynchronous nature of execution. Instead, use catch method of Promise or wrap it inside await expression.

This rule reports try-catch statements containing nothing else but call(s) to a function returning a Promise (thus it's less likely that catch is intended to catch something else than Promise rejection).

### Noncompliant Code Example




```
function runPromise() {
  return Promise.reject("rejection reason");
}

function foo() {
  try { // Noncompliant, the catch clause of the 'try' will
    runPromise();
  } catch (e) {
    console.log("Failed to run promise", e);
  }
}
```

### Compliant Solution

```
function foo() {
  runPromise().catch(e => console.log("Failed to run promise", e));
}





// or
async function foo() {
  try {
    await runPromise();
  } catch (e) {
    console.log("Failed to run promise", e);
  }
}
```

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-4822

1/2

<div>Unused assignments should be removed</div> <div> Code Smell</div>
<div>Function parameters with default values should be last</div> <div> Code Smell</div>
<div>Functions should not be defined inside loops</div> <div> Code Smell</div>
<div>"switch" statements should not have too many "case" clauses</div> <div> Code Smell</div>
<div>...</div>