




 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... 🔍


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

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

Unused assignments should be removed

Analyze your code

 Code Smell  Major  cwe unused

A dead store happens when a local variable is assigned a value that is not read by any subsequent instruction. Calculating or retrieving a value only to then overwrite it or throw it away, could indicate a serious error in the code. Even if it's not an error, it is at best a waste of resources. Therefore all calculated values should be used.

Noncompliant Code Example

```
i = a + b; // Noncompliant; calculation result not used before
i = compute();
```

Compliant Solution

```
i = a + b;
i += compute();
```

Exceptions

This rule ignores initializations to -1, 0, 1, null, undefined, [], {}, true, false and "". Variables that start with an underscore (e.g. `_unused`) are ignored.

This rule also ignores variables declared with object destructuring using rest syntax (used to exclude some properties from object):




```
let {a, b, ...rest} = obj; // 'a' and 'b' are ok
doSomething(rest);

let [x1, x2, x3] = arr;    // but 'x1' is noncompliant, as o
doSomething(x2, x3);
```

See

- [MITRE, CWE-563](#) - Assignment to Variable without Use ("Unused Variable")





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-1854

1/2

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