
































-  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... 

-  Bug
- Variables declared with "var" should be declared before they are used
-  Code Smell
-  Track lack of copyright and license headers
-  Reading the Standard Input is security-sensitive
-  Using command line arguments is security-sensitive
-  Using Sockets is security-sensitive
-  Executing XPath expressions is security-sensitive
-  Encrypting data is security-sensitive
-  Using regular expressions is security-sensitive
-  Class methods should be used instead of "prototype" assignments
-  Function constructors should not be used
-  Variables should be declared with "let" or "const"
- Unchanged variables should be

## Character classes in regular expressions should not contain the same character twice

[Analyze your code](#)

 Code Smell

 Major 

 regex

Character classes in regular expressions are a convenient way to match one of several possible characters by listing the allowed characters or ranges of characters. If the same character is listed twice in the same character class or if the character class contains overlapping ranges, this has no effect.

Thus duplicate characters in a character class are either a simple oversight or a sign that a range in the character class matches more than is intended or that the author misunderstood how character classes work and wanted to match more than one character. A common example of the latter mistake is trying to use a range like `[0-99]` to match numbers of up to two digits, when in fact it is equivalent to `[0-9]`. Another common cause is forgetting to escape the `-` character, creating an unintended range that overlaps with other characters in the character class.

### Noncompliant Code Example

```
/[0-99]/ // Noncompliant, this won't actually match strings
/[0-9.-_]/ // Noncompliant, .-_ is a range that already cont
```





### Compliant Solution

```
/[0-9]{1,2}/
/[0-9.\-_]/
```

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](#)

<div>marked "const"</div> <div> Code Smell</div>
<div>Wildcard imports should not be used</div> <div> Code Smell</div>
<div>"switch" statements should not be nested</div> <div> Code Smell</div>
<div>Cyclomatic Complexity of functions should not be too high</div> <div> Code Smell</div>
<div>"strict" mode should be used with caution</div>