sonar RULES Products

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...

Security Hotspot

Using shell interpreter when executing OS commands is security-sensitive
Security Hotspot

Setting loose POSIX file permissions is security-sensitive
Security Hotspot

Formatting SQL queries is security-sensitive
Security Hotspot

Comma operator should not be used
Code Smell

Regular expressions should not contain empty groups
Code Smell

Regular expressions should not contain multiple spaces
Code Smell

Chai assertions should have only one reason to succeed
Code Smell

Single-character alternations in regular expressions should be replaced with character classes
Code Smell

Reluctant quantifiers in regular expressions should be followed by an expression that can't match the empty string
Code Smell

Tests should check which exception is thrown
Code Smell

Character classes in regular expressions should not contain the

"void" should not be used

Analyze your code

Code Smell Critical confusing

The `void` operator evaluates its argument and unconditionally returns `undefined`. It can be useful in pre-ECMAScript 5 environments, where `undefined` could be reassigned, but generally, its use makes code harder to understand.

Noncompliant Code Example

```
void doSomething();
```

Compliant Solution

```
doSomething();
```

Exceptions

No issue is raised when `void 0` is used in place of `undefined`.

```
if (parameter === void 0) {...}
```

No issue is raised when `void` is used before immediately invoked function expressions.

```
void (function() {
   ...
}());
```

No issue is raised when `void`'s argument is a promise.

```
const promise = new Promise((resolve, reject) => {
   setTimeout(() => {
      resolve('done');
   }, 3000);
});
void promise;
```

Available In:
sonarlint sonarcloud sonarqube

same character twice

Code Smell

Names of regular expressions named groups should be used

Code Smell

Regular expressions should not be too complicated

Code Smell

Shorthand promises should be used

Code Smell

Template literals should not be nested