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

TypeScript static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your TYPESCRIPT code

All rules 279 | Vulnerability 27 | Bug 51 | Security Hotspot 43 | Code Smell 158 | Quick Fix 50

Tags | Search by name...

Debugger statements 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

Primitive return types should be used
Code Smell

Default type parameters should be omitted
Code Smell

Type assertions should use "as"
Code Smell

Method overloads should be grouped together
Code Smell

Interfaces should not be empty
Code Smell

Trailing commas should be used
Code Smell

"import" should be used to include

Function parameters with default values should be last

Analyze your code

Code Smell Major es2015

The ability to define default values for function parameters can make a function easier to use. Default parameter values allow callers to specify as many or as few arguments as they want while getting the same functionality and minimizing boilerplate, wrapper code.

But all function parameters with default values should be declared after the function parameters without default values. Otherwise, it makes it impossible for callers to take advantage of defaults; they must re-specify the defaulted values or pass `undefined` in order to "get to" the non-default parameters.

Noncompliant Code Example

```
function multiply(a = 1, b) {  // Noncompliant
  return a*b;
}

var x = multiply(42);  // returns NaN as b is undefined
```

Compliant Solution

```
function multiply(b, a = 1) {
  return a*b;
}

var x = multiply(42);  // returns 42 as expected
```

Available In:
sonarlint sonarcloud sonarqube

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