




 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

Bug
Destructuring patterns should not be empty
Bug
The output of functions that don't return anything should not be used
Bug
Comma and logical OR operators should not be used in switch cases
Bug
Generators should "yield" something
Bug
Attempts should not be made to update "const" variables
Bug
Strict equality operators should not be used with dissimilar types
Bug
"new" operators should be used with functions
Bug
Non-existent operators '+=', '=-' and '!=' should not be used
Bug
"NaN" should not be used in comparisons
Bug
Setters should not return values
Bug
Properties of variables with "null" or "undefined" values should not be accessed
Bug

A compare function should be provided when using "Array.prototype.sort()"

Analyze your code

- Bug
- Critical
- Quick Fix
- bad-practice

The default sort order is alphabetic, rather than numeric, regardless of the types in the array. Specifically, even if an array contains only numbers, all values in it will be converted to strings and sorted lexicographically, for an order like this: 1, 15, 2, 20, 5.

Fortunately the `sort` method allows you to pass an optional compare function to specify the sort order. When a compare function is supplied, the returned order depends on the return value of the compare function.

Noncompliant Code Example

```
var myarray = [80, 3, 9, 34, 23, 5, 1];

myarray.sort();
console.log(myarray); // outputs: [1, 23, 3, 34, 5, 80, 9]
```





Compliant Solution

```
var myarray = [80, 3, 9, 34, 23, 5, 1];

myarray.sort((a, b) => (a - b));
console.log(myarray); // outputs: [1, 3, 5, 9, 23, 34, 80]
```

Available In:

sonarlint | sonarcloud | sonarqube

<div>A "for" loop update clause should move the counter in the right direction</div> <div> Bug</div>
<div>Return values from functions without side effects should not be ignored</div> <div> Bug</div>
<div>Special identifiers should not be bound or assigned</div> <div> Bug</div>
<div>Values should not be uselessly incremented</div> <div> Bug</div>