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...

Files should not have too many lines of code
Code Smell

Lines should not be too long
Code Smell

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

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 befo
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:

sonarlint sonarcloud sonarqube

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