**Bugs**

### 1) "=+" should not be used instead of "+="

### The operator pairs (=+, =-, =!) will compile and run but don’t give the expected output.

### They will also raise issue if there is no spacing between the operator pairs and if there is atleast one space character after the operators.

### 2) "Array.reverse" should not be used misleadingly

### Array.reverse() alters the source array  and also returns the altered version, which is likely not what was intended. Therefore, its usage must be taken care of it.

### 3) "delete" should be used only with object properties

### Delete operator gives unexpected results when used for other than object parameters. It may or may not work properly for the rest.

### 4) "eval" and "arguments" should not be bound or assigned

### Eval is used to evaluate a string as JavaScript code, and arguments is used to access function arguments through indexed properties. So, they shouldn’t be bound or assigned

### 5) "in" should not be used with primitive types

### The in operator is used to test whether the specified property is in the specified the object. So, if it is used with primitive data then it raises a typeError.

### 6) "NaN" should not be used in comparisons

### NaN is not equal to anything. It isn’t equal to itself. Therefore, there is no point of using comparisons with NaN.

### 7) "new" operators should be used with functions

### The ‘new’ operator must be used only with functions that are insisted to work as a constructor. If we use it with a function that isn’t a function then it raises typeError.

### 8) "yield" expressions should not be used outside generators

The yield expression is only used inside the generator to iterate against the results. It has no purpose outside the generator.

### 9) A "for" loop update clause should move the counter in the right direction

A for loop with a stop condition that can never be reached, such as one with a counter that moves in the wrong direction, will run infinitely. While there are occasions when an infinite loop is intended, the convention is to construct such loops as while loops. More typically, an infinite for loop is a bug.

### 10) All branches in a conditional structure should not have exactly the same implementation

### Having all branches in a switch or if chain with the same implementation is an error. Either a copy-paste error was made and something different should be executed, or there shouldn't be a switch/if chain at all.

**Vulnerability**

### 1) "alert(...)" should not be used

Alert() is useful in debugging environment only.

### 2) Code should not be dynamically injected and executed

Eval() is used to run code at runtime. But it isn’t safe to run code at run time as it might be a potential security threat and can even slow down the execution process.

### 3) Console logging should not be used

### Never should we include the debugging statements in production code which runs on the client side.

### 4) Cross-document messaging domains should be carefully restricted

We must be careful while sending data to documents hosted by unknown domains.

### 5) Debugger statements should not be used

Debugger statements must be removed from the source code before it is being implemented in the production environment.

### 6) Function constructors should not be used

They are slow and poses security risk.

### 7) Local storage should not be used

Local storage might enable us to store huge amounts of data on client side. But it also has a risk as it doesn’t have any encryption.

### 8) Untrusted content should not be included

It might compromise the whole site or even expose the users to attackers.

### 9) Web SQL databases should not be used

It is deprecated and is never implemented in a large scale. It also poses security threats.

**Code Smell**

1. **"===" and "!==" should be used instead of "==" and "!="**

The == and != perform type coercion before comparing and therefore mask the typeerrors. So use === and !== which have less side-effects.

### "[type=...]" should be used to select elements by type

It is faster in performance as it uses the native DOMquerySelectorAll() method.

### "continue" should not be used

It is unstructured control flow and makes the code less testable, readable and maintainable respectively.

### "delete" should not be used on arrays

If we use delete in arrays then it will create a hole as the indices/keys will not move or rearrange by themselves.

### "find" should be used to select the children of an element known by id

It is used for top level selection and reduces the effort on jquery engine. It makes the query faster.

### "future reserved words" should not be used as identifiers

It might produce an error in javascript strict mode.

### "if ... else if" constructs should end with "else" clauses

It is used for the purpose of defensive programming and is similar to that of default in switch-case.

### "indexOf" checks should not be for positive numbers

Most checks against an indexOf call against a string or array compare it with -1 because 0 is a valid index. Any checks which look for values >0 ignore the first element, which is likely a bug.

### "strict" mode should be used with caution

Not all browsers support strict mode and therefore careful implementation of strict mode is needed.

### "undefined" should not be assigned

Undefined is the value given to variables that are not yet created. Therefore it can’t be used for assignment purposes.