Code Quality JS

Sukesh Kancharla

1. Bugs:
    - "in" should not be used with primitive types:
        - In method can be called to check the properties of an object. It cannot be used for primitive data types where it gives Type error.
    - Function calls should not pass extra arguments:
        - Even if we pass more than required arguments, the function execution will not be affected but passing more parameters is waste of resources and memory.
    - "NaN" should not be used in comparisons:
        - Since NaN != NaN, comparison doesn't do our work as we need.
    - "new" operators should be used with functions:
        - New can be used only with functions which are constructors and using it any other place gives type error.
    - A "for" loop update clause should move the counter in the right direction:
        - Counter should be moved in proper direction because incrementing where decrement is needed may lead to infinite loop.
    - Calls should not be made to non-callable values:
        - Call should not be invoked on non-callable values. Here in java script, we won't get compile errors since it is not strongly typed language but it gives run time exceptions when we try to invoke variables considering them as functions.
    - Comma and logical OR operators should not be used in switch cases:
        - Comma operator returns the second value only which is not preferred in switch case where as OR operator returns left value only which is not advisable.
    - Function argument names should be unique:
        - If we use duplicate argument names the right most parameter overwrites all the left parameters having the same name. Those left parameters can be accessed using arguments[i] but to avoid ambiguity and confusion, duplication is not preffered.
    - Non-existent operators '=+', '=-' and '=!' should not be used:
        - Using operators =+,=-,=! don't give a compile error or runtime error but the results are not expected as (+=,-=,!=).
    - Setters should not return values:
        - Functions using 'set' keyword return the value we are passing. If a value is returned explicitly, it will be ignored.
2. Vulnerabilities:
    - Code should not be dynamically injected and executed:
        - 'eval' function is used to execute code dynamically but it is slow and has a potential security issue where arguments are not proper.
    - "alert(...)" should not be used:

    - Alert will be used in debugging stage but in production stage it has to be avoided where it may reveal sensitive information to users.
- Cross-document messaging domains should be carefully restricted:
    - HTML5 provides an option to send info from different domains. It raises this issue to give a warning to avoid sensitive info to be sent as message.
- Debugger statements should not be used:
    - Debugger can be used where it acts as a break point. So in production phase all the debugger statements have to be removed from the source code.
- Local storage should not be used:
    - Local and session storage are provided by html5. These can be used for speed access of data but have to be avoided because there is no encryption which makes it can be accessed by the code.
- Untrusted content should not be included:
    - Don't include scripts which are from untrusted sources which may attract attackers to hack your site.
- Console logging should not be used:
    - Console logging can be used while development whereas in production logging statements should be removed because the code will be in hands of client and debugging statements can be seen which leaks sensitive information.
- Web SQL databases should not be used:
    - It was implemented by W3C and is not available in all the browsers.

3. Code Smells:
    - Octal values should not be used:
        - Numbers starting with 0 will be considered in octal format which is not expected.
    - "delete" should not be used on arrays:
        - Delete can be used for arrays but it leaves a hole at that places since it doesn't perform shift operation.
    - Arguments to built-in functions should match documented types:
        - Calls should be valid to get results from the in-built functions. Otherwise, results will be unexpected.
    - Array indexes should be numeric:
        - Named indices can be created in arrays but it is not suggested to have named indices instead, use objects which are also as easy of creating arrays.
    - Jump statements should not be used unconditionally:
        - Unconditional use of break, continue, return is useless in loops where loop will be executed once in break and return case where as continue will make loop to be executed always.
    - Nested blocks of code should not be left empty:

    - Empty blocks should be removed or filled with required or else it consumes execution time unnecessarily.
- "switch" statements should have at least 3 "case" clauses:
    - Switch case should have at least 3 case clauses or else we can use if-else statements.
- A "while" loop should be used instead of a "for" loop:
    - If a for loop is having only condition, it is preferred to use while loop.
- “if ... else if" constructs should end with "else" clauses:
    - After the last else if statement, else statement is preferred in view of defensive programming where it handles the unexpected value of a variable or it may show why no action has been performed in comments section.
- "switch" statements should end with "default" clauses:
    - It is practice of defensive programming where we can handle unexpected values and can show why no action has been taken.