# Преглед на кода (Code Review)

## Structure

* Does the code completely and correctly implement the design?
* Does the code conform to any pertinent coding standards?
* Is the code well-structured, consistent in style, and consistently formatted?
* Are there any uncalled or unneeded procedures or any unreachable code?

*No, all of the variables and functions are being used. No unreachable code present.*

* Can any code be replaced by calls to external reusable components or library functions?

*All functions that could be replaced with external libraries are already replaced with call to those external libraries.*

* Are there any blocks of repeated code that could be condensed into a single procedure?

*No, repeated code is avoided on design level.*

* Are any modules excessively complex and should be restructured or split into multiple routines?

*No, all the logic is separated in different functions for easier reading, understanding and reusability.*

## Documentation

* Is the code clearly and adequately documented with an easy-to-maintain commenting style?
* Are all comments consistent with the code?
* All source code contains @author for all authors.
* @version should be included as required.
* All class, variable, and method modifiers should be examined for correctness.
* Complex algorithms should be explained with references. For example,  document the reference that identifies the equation, formula, or pattern. In all cases, examine the algorithm and determine if it can be simplified.
* Code that depends on non-obvious behavior in external frameworks is documented with reference to external documentation.
* Confirm that the code does not depend on a bug in an external framework which may be fixed later, and result in an error condition. If you find a bug in an external library, open an issue, and document it in the code as necessary.

## Variables

* Are all variables properly defined with meaningful, consistent, and clear names?

*Camel Case convention is used. All variable names are descriptive.*

* Are there any redundant or unused variables?

*No, all variables are used.*

## Loops and Branches

* Are all loops, branches, and logic constructs complete, correct, and properly nested?
* Are the most common cases tested first in IF- -ELSEIF chains?
* Does every case statement have a default?

*Switch-case is not used.*

* Are indexes or subscripts properly initialized, just prior to the loop?
* Can any statements that are enclosed within loops be placed outside the loops?

*No, use of statements within a loop is avoided as much as possible for better performance.*

## Testing

* Unit tests are added for each code path, and behavior. This can be facilitated by frameworks like Mocha and Chai.
* Unit tests must cover error conditions and invalid parameter cases.
* Unit tests for standard algorithms should be examined against the standard for expected results.
* Check for possible null pointers are always checked before use.
* Javascript does not support pointers.
* Do not write a new algorithm for code that is already implemented in an existing public framework API, and tested.

*Multiple use of existing libraries.*

## Error Handling

* Invalid parameter values are handled properly early in methods (Fast Fail).
* Consider using a general error handler to handle known error conditions.
* Avoid using RuntimeException to avoid making code changes to implement correct error handling.
* **Don't swallow exceptions!** For example catch (Exception ignored) {}. It should at least log the exception.

## Performance

* No busy-wait loops instead of proper thread synchronization methods. For example, avoid while(true){ ... sleep(10);...}
* Do not leave debugging code in production code.
* Avoid console.log() statements in code.