- 2.1 File name
  - File names must be all lowercase and may include underscores (_) or dashes (-), but no additional punctuation. Follow the convention that your project uses. Filenames' extension must be `.js`.

- 3.4.1 Imports
  - 3.4.1.1.1 File extensions in import paths
    - The `.js` file extension is not optional in import paths and must always be included.
  - 3.4.1.2 Importing the same file multiple times
    - Do not import the same file multiple times. This can make it hard to determine the aggregate imports of a file.

- 3.4.2 Exports
  - 3.4.2.1 Named vs default exports
    - Use named exports in all code. You can apply the `export` keyword to a declaration, or use the `export {name};` syntax.
    - Do not use default exports. Importing modules must give a name to these values, which can lead to inconsistencies in naming across modules.
  - 3.4.2.3 Mutability of exports
    - Exported variables must not be mutated outside of module initialization.

- 4.1 Braces
  - 4.1.1 Braces are used for all control structures
    - Braces are required for all control structures (i.e. `if`, `else`, `for`, `do`, `while`, as well as any others), even if the body contains only a single statement. The first statement of a non-empty block must begin on its own line.
      - Exceptions: Is NOT required for a simple if statement that can fit entirely on a single line with no wrapping

- 4.8 Comments
  - MUST
    - Short /**/ multi-line block comment before every function
    - // single line comment for important statements inside of function
  - 4.8.1 Block comments are indented at the same level as the surrounding code.
    - Use //-style for single line comments. For multi-line /* … */ comments, subsequent lines must start with * aligned with the * on the previous line, to make comments obvious with no extra context.

- 5.1 Local variable declarations
  - 5.1.1 Use `const` and `let`
    - Declare all local variables with either `const` or `let`. Use const by default, unless a variable needs to be reassigned. The `var` keyword must not be used.

- **6.1 Rules common to all identifiers**
  - **Give as descriptive a name as possible, within reason. Do not use abbreviations that are ambiguous or unfamiliar to readers outside your project, and do not abbreviate by deleting letters within a word.**

- **6.2.3 Method names**
  - **Method names are written in lowerCamelCase.**
  - **Names for @private methods must end with a trailing underscore.**
  - **Method names are typically verbs or verb phrases. For example, sendMessage or stop_. Getter and setter methods for properties are never required, but if they are used they should be named getFoo (or optionally isFoo or hasFoo for booleans), or setFoo(value) for setters.**

- **6.2.5 Constant names**
  - **Constant names use CONSTANT_CASE: all uppercase letters, with words separated by underscores. There is no reason for a constant to be named with a trailing underscore, since private static properties can be replaced by (implicitly private) module locals.**