# Coding Conventions and Standards

## General conventions

|  |  |  |
| --- | --- | --- |
| Convention Number | Convention Type | Convention |
| 1 | Naming Convention for general front-end and back-end | All variable global and local naming will be in lower camelCase.  Global variables are to be defined with the imported modules above all methods and functions at the top of the file to make them distinct from local variables which will be defined within the methods where they are needed.  Functions and methods will also be named in lower camelCase.  Constructors and Classes will however be in PascalCase. An example of this will be the constructors used to put data into the json format to add to the database like the UserAccounts constructor.  Regular JavaScript files and Node.js files should be named in lower CamelCase for example index.js or retrieveQuestions.js |
| 2 | Naming Convention for Vue.js component files | Vue.js component file names should be named in PascalCase for example App.vue or UserLogin.vue as defined by the Vue.js style guide.  When importing Vue.js components, they should also follow PascalCase as defined by the Vue.js style guide. As defined here:  Vuejs.org. 2020. Style Guide — Vue.Js. [online] Available at: <https://vuejs.org/v2/style-guide/> [Accessed 10 December 2020]. |
| 3 | Variable Naming | As stated above all variables will be in lower camelCase but the name itself should also be relevant to what the variable is for or its actual purpose. Giving each other knowledge on its purpose |
| 4 | Style Guide | We will be following the JavaScript Standard Style guide for all JavaScript code for our node.js back-end and vue.js front-end code outlined here:  Standardjs.com. 2020. JavaScript Standard Style. [online] Available at: <https://standardjs.com/rules.html> [Accessed 10 December 2020].  As well as this we will follow additional front-end Style guide for Vue.js components defined here:  Vuejs.org. 2020. Style Guide — Vue.Js. [online] Available at: <https://vuejs.org/v2/style-guide/> [Accessed 10 December 2020]. |
| 5 | File Structure | Code relevant to the back-end will be in the server folder and code relevant to the front-end will be in the client folder. Both folders will have a node modules file with all the libraries needed to run the code within them.  In the server folder we will have a file called index.js which handles the requests made to the backend it will call functions from the src folder within the server folder such as a hashing function.  In the client folder we will also have a src folder which will have the files main.js and App.vue responsible for initialising the frontend. The file main.js will call the router function in the router folder within src. This router function will be responsible for routing the user to the different page components for example /login to the Login.vue component. These components will be within the component folder in the src folder. |
| 6 | Header Comments | The format will be as follows  Name of Module  Date of Creation  Date of Last Edit  Authors  High level overview of the module |
| 7 | Line Length | To improve readability of the code the lines should be kept under 80 characters |
| 8 | Operators | Spaces should be present before and after operators such as var num = 10 + 10 |
| 9 | Commas | Commas should have a space after them but not before for example 1, 2, 3, 4, 5 |
| 10 | Indentation | An Indentation of 2 spaces will be used. As it is the part of the JavaScript Standard Style guide rules we are following. |
| 11 | Quotation Marks | Use only single quotation marks for example (‘Hello World’) in the JavaScript code as it can cause escape errors if you use double quotations in JavaScript code. However double quotes can be used in the html code templates. |
| 12 | Nested Block spacing | Nested blocks should start on a new line and the opening curly braces on that same new line with the keyword such as an if. Nested blocks after a previous nested block will start on a new line below the closing curly braces. To add to this there is an exception else statements should open on the same line as the closing if statement curly braces. |
| 13 | Functions and method length | Each function and method should only preform one task. This will allow us to easily test separate functions in unit tests. |
| 14 | Functions and method naming | As defined earlier they will be in lower camelCase but the name itself should also be relevant to the function or methods actual purpose. So that we all have an idea of what it will do. |
| 15 | Importing functions and methods | We will be importing differently in the server side to the client side due to the way the official docs for each framework deal with imports.  Client side written with Vue.js will follow the import x from y (tree shaking)  Server Side written with node.js will follow the require format for importing const x = require(y) |
| 16 | No Semicolons to end lines | JavaScript can use semi colons to end lines but they are not required as JavaScript uses Automatic Semicolon Insertion. Making them optional to include. On top of this the styling guide we are following JavaScript Standard Style is against their use and the Vue.js docs also don’t use them to terminate statements either. So we have decided not to use them. |
| 17 | No trailing white space to end lines | There should be no white space at the end of lines to avoid confusion. |