Coding Standards

Triple Parity

September 20, 2018

1 Introduction

This document gives a brief overview of the coding standards we plan on using in our project. To ensure consistency throughout our project we will cover how we will be deigning our coding documents written in Angular (front end), and JavaScript (back end). We will be using TypeScript in Angular and therefore will be keeping the same coding standards for both front and back end JavaScript.

2 Required and optional items

2.1 Files Headers

The file header should appear clearly at the beginning of every program file and should include information to help the programmers better understand the file. The file header should include the file name, version number, project name, organization, list of classes, related documents.

```
/*
*           fileName: dockerUI.component
*           versionNumber: 2.0
*           projectName: Docker Swarm Manager
*           organization: Compiax
*           listOfClasses: CardUI, Container
*           relatedDocuments: dockerUI.html, dockerUI.css

*/
```

The above examples will be the viewed as the coding requirements any other fields such as update history, test cases and assumptions can be seen as conventions.

2.2 Description of classes

Our project will also require us to create a description for each class. The description will have to include the classes purpose, A description of the classes methods, a brief description of each field including the name of the field and finally In-code comments. Where ever there are sections of code that require a base understanding of what the code will be doing an In-code comment must be added.

3 Format and language

3.1 Naming Conventions

In Angular 5 the convention for file names is as follow: files begin with a lowercase letter, if a file name where to be separated with a space instead use a "." to seperate multiple words. Files will either end with the .ts (TypeScript) .html (The html page) .css(style sheet). Function names inside the document will be begin with a lowercase letter, if there are multiple words in function description camel casing will be used to separate the words for example dockerUserInterface. attributes and local variables will follow the same structure. Global variables however will begin

with a capital letter and then follow camel casing if need be. Names for constants are all in capital letters for example CONTAINERS.

3.2 Formating conventions

For JavaScript and TypeScript

- Every line after a function header, if statement, while loop or for loop will be indented exactly once.
- Functions method calls will be aligned together by one indent space.
- The beginning bracket will appear on the same line as a function header or while loop statement for example while{
  .... }
- All imports should appear at the top of the page, with no line in between them each appearing on a new line.
- variables and attributes must be declared at the top of a function or class

For HTML

- All sub elements or nested elements ie. a <tr> element inside a <table> element or a nested inside another div should be indented exactly once.
- All empty elements (input fields and image tags) must be closed properly.
- All elements should be organized in divs with appropriate ID or class names.
- Script fields should only appear in the header and not in the HTML body field.
- Multiple attributes inside a element should be separated with a single space.

For CSS

- For every selection a single bracket will appear on the same line as the element just like before in the while loops and function calls in JavaScript and TypeScript
- After every element the properties that we wish to change with CSS will appear on a new line ans indented exactly once.

These formating standards can be related to the built in format standards of the IDE webStorm so we will likely stick to using this IDE for coding in the future.

3.3 In-code comment conventions

Every inline comment will be marked with the following "//". Every block comment will be marked with the follow1ing "/***/", A comment becomes a block comment as soon as it is over 2 lines. Each method call in every class and each helper function should be commented according to JSDocs. This ensures a good description of each function call as well as ensure that our code is well documented. JSDocs requires that a function description is given, function parameters are defined, and that what the function should be returning is defined. Whenever a function is called from a file that does not contain that function a comment should be added stating in which file that function can be found.

4 Rules and responsibilities

These standards should be applied to all programs created. This will force our developers to get into the habit of creating well documented code that is easily maintainable and user friendly. Test case programs should follow their own conventions for documentation and do not need to be documented with JSDocs. Each group member will have to take up the responsibility to stick to these codding standards and help others who are lacking in this field. Any new coding standards will have to be discussed as a group as changes may be time consuming and therefore redundant, thus a vote must be taken to decide if a new coding standard should be added or an old one removed.