University of Pretoria
COS 301
HighTech

Coding Standards

Team Members:

Janaki Patil u16006110@tuks.co.za
Theo Naidu u16148861@tuks.co.za
Tristan Sander-hughes u17071390@tuks.co.za
Zi Xin Zhang u15192556@tuks.co.za
Alexandros Mendes Petrou u15291792@tuks.co.za

Introduction

This document is intended to be a guide on the coding standards for Geyser Project.
The rest of this document will show general conventions and examples where it may be needed used by each different language used throughout the project.

Javascript

Naming Convention

- **Classes**: Should be nouns in pascal casing, using whole words is prefered and avoid acronyms and abbreviations unless it is an accepted acronym or abbreviation that is accepted in the programming industry.
- **Variables**: Should use camel casing, it should be short and meaningful, rather have a variable name that is long and makes sense then short and hard to understand its function. Variable names should always start with an alphanumeric character.
- **Methods**: Should use camel casing and preferably a verb.
- **Constants**: Should use all uppercase and multi worded constants should have an underscore between the words.

Formatting Convention

- **Indentations**: Each code block should have 1 level of indentation. A level of indentation should either be 1 tab space, or 4 spaces but never both in the same file.
- **Semicolons**: Each code statement, or line of logic should be ended with a semicolon where it is applicable, and each code statement or line of logic is preferably is put in a new line.
- **Quotes**: Strings should always use double quotes.
- **Braces**: Opening and closing braces should be put in a new line
- **Commenting**: Commenting should always be kept short and descriptive. Comments should only exist if there is a need to comment out code to be used later or debugging, to explain a long sequence of code if it will save time. Code should always be self explanatory.

Javascript Example Code

```
class AuthenticationService
{
    constructor()
    {
        this.userName = "";
        this.password = "";
        this.HASHING_SALT = 37;
        this.helloWorld = "Hello World!"; //example string, it has no use
    }

    authenticate(userName, password)
    {
        /*
            imagine there is tons of code here
        */
    }
}
```

Typescript

Naming Convention

- Same naming conventions as Javascript

Formatting Convention

- **Indentations**: Each code block should have 1 level of indentation. A level of indentation should either be 1 tab space, or 4 spaces but never both in the same file.
- **Semicolons**: Each code statement, or line of logic should be ended with a semicolon where it is applicable, and each code statement or line of logic is preferably is put in a new line.
- **Quotes**: Strings should always use single quotes.
- **Braces**: Opening braces should be put on the same line as the header and closing braces should be put in a new line
- **Commenting**: Commenting should always be kept short and descriptive. Comments should only exist if there is a need to comment out code to be used later or debugging, to explain a long sequence of code if it will save time. Code should always be self explanatory.
- **Types**: Variable type and return types should preferably be shown.

Typescript Example Code

```
export class LoginPage implements OnInit {

    validationsForm: FormGroup;
    errorMessage: String = '';

    validationMessages: Object = {
      userID: [
        { type: 'required', message: 'userID is required.' }
      ],
      password: [
        { type: 'required', message: 'Password is required.' },
        { type: 'minlength', message: 'Password must be at least 5
characters long.' }
      ]
    };

    loginUser(value: any): void {
        this.authService.loginUser(value).then(successful => {
          if(successful) {
            this.router.navigate(['dashboard']);
          } else {
            this.errorMessage = 'UserID or password is incorrect.';
          }

        }, err => {
          this.errorMessage = 'UserID or password is incorrect.';
        });
      }

    /*
        imagine rest of class here
    */
}
```

HTML

Formatting Convention

- **Indentations**: Each code block should have 1 level of indentation. A level of indentation should either be 1 tab space, or 4 spaces but never both in the same file.
- **Tags**: There is no preferred way on how the tags should be handled, only that they need to make logical sense in the way they are laid out.
- **Commenting**: Commenting should always be kept short and descriptive. Comments should only exist if there is a need to comment out code to be used later or debugging, to explain a long sequence of code if it will save time. Code should always be self explanatory.

HTML Example Code

```
<ion-header>
  <ion-toolbar>
      <!--<ion-back-button (click)="goBack()"></ion-back-button>-->
    <ion-title>Images</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <h3 class="instruction">{{displayInstruction}}</h3>
  <!--<div *ngIf= 'readyToTakePicture'>{{openCam()}}</div>-->
  <div class="imgBox">
    <img src="{{image}}">
  </div>
</ion-content>
```

CSS/SCSS

Formatting Convention

- **Indentations**: Each code block should have 1 level of indentation. A level of indentation should either be 1 tab space, or 4 spaces but never both in the same file.
- **Braces**: Opening braces should be put on the same line as the header and closing braces should be put in a new line
- **Semicolons**: Each code statement, or line of logic should be ended with a semicolon where it is applicable, and each code statement or line of logic is preferably is put in a new line.

CSS/SCSS Example Code

```
.footerButton {
  display: inline-block;
}

.instruction {
  text-align: center;
}

.imgBox {
  display: flex;
  justify-content: center;
  padding: 5px;
}
```