University of Pretoria
Software Engineering - COS 301

Coding standards document

Contrapositives
May 2019

Authors:

Brendan Bath	**u16023359**
Musa Mathe	**u15048030**
Jessica da Silva	**u16045816**
Natasha Draper	**u16081758**

Coding standards

contrapositives

May 2019

1 Backend

1.1 Python

1. Separate each disjointed statement to be on it's single line.
2. Use positional system to pass arguments to functions e.g send(message,recipient)
3. For private properties and implementation details make sure to prefix all "internals" with an underscore.
4. When a function grows in complexity avoid returning meaningful values from many output points in the body.
5. If you know the length of a list or tuple, you can assign names to its elements with unpacking e.g for index,item in enumerate(some$_list$) :
6. If you need to assign something (for instance, in Unpacking) but will not need that variable, use underscore

2 Frontend

2.1 Vue.js style

1. Add a component on the views sub-directory in the frontend directory
2. All imports should occur at the top of the page.
3. Component names should always be meaningful words, e.g Vue.component('Login',) , Vue.component('Register',)
4. In component **data** you must not specify the function e.g Vue.component('some-comp', data: return email: 'email'
5. **Prop** definitions should be as detailed as possible by atleast specifying types e.g props:status: String)

6. Always use **key** with v-for e.g `<ul><li v-for="todo in todos" :key="todo.id" {todo.text }</li></ul>` in order to maintain internal component state down the subtree.

7. Each component should be on its own file to quickly find a component when you need to edit it or review how to use it.

8. Filenames of single-file components should either be always PascalCase or always kebab-case for autocompletion in code editors, for consistency with how we reference components in JS(X) and templates. e.g MyComponent.vue or my-component.vue

9. Base components that apply app-specific styling and conventions should all begin with a specific prefix, such as App e.g AppFooter

10. Components with no content should be self-closing in single-file components, string templates but never in DOM templates. e.g single-file ,string template `<MyComponent/>` , DOM template `<my-component></my-component>`

11. You may want to add one empty line between multi-line properties to avoid components to be cramped and difficult to read.

12. All imports of files should be included at the top of the file.

2.2 Typescript

1. Only single quotes can be used. e.g name:'home'

2. Use only 2 spaces for indentation.

3. You can use underscores at the beginning.

4. Do not use semicolons at the end of every statement e.g

5. Arrow functions with one parameter must not have parentheses.

6. Only use lower CamelCased or UPPER$_{C}ASEDvariablenames$.

3 General(Style, Structure and layout)

3.1 Variables, Methods, Attribute:

1. Should be short and descriptive.
2. Should begin with lower case letter.
3. Upper-case letter should be used immediately after the first name description of the variable.
4. Should not be a single character e.g. string y

3.2 Indendation and layout

1. Lines should be kept to a sensible length to make the code easier to read and print.
2. Single blank lines should be used to separate methods and to emphasise blocks of code.
3. Blocks that are nested more should be indented more.

3.3 Directories

1. All tests should be placed in a 'tests' directory.
2. All vue components should be placed inside 'view' directory.

3.4 Exceptions

1. Exceptions should be used where necessary.
2. Instead of throwing basic Exception classes, sub-classes should be made with more meaningful names.
3. Appropriate catch statements should be put in place to allow the program to recover if need be.

3.5 Commenting

1. Use in-line commenting to help the next developer who might be editing your code.
2. Inline comments should appear on the line above the code you are commenting.
3. Comments should be added within the body of a method if they are used within that method.

3.6 Version control

1. No commented out code must be committed unless you have a very good reason that is clearly described in a comment by the code you are committing.

4 Repository

4.1 Repository definition

1. Do not commit to master branch.
2. Do not commit any code with errors , test first to make sure your code is working without breaking any of the main code.
3. For each feature you create, create a new feature branch then merge to develop.
4. Always create continuous commits as you develop a certain feature.
5. All the files should belong to a specific folder.
6. Folder names should be descriptive enough to show what they contain.

4.2 Repository structure

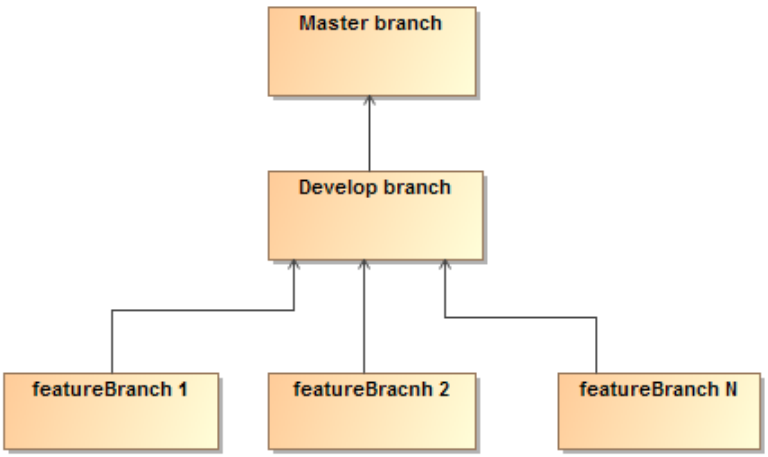

Figure 1: Git structure