Coding Standards

Binary Ninjaz

Contents

1 Introduction

The project and design we've undertaken means that we will need to use multiple languages and IDE's to complete the project. As such we hav decided that the best common practice and standards of each language shall be the standard that is used for that respective language/environment.

2 Language Style Guides

2.1 HTML/CSS

The HTML and CSS standards we shall follow will be the ones of Google's HTML/CSS style guide

2.2 JavaScript

For JavaScript we will be using Google's JavaScript style guide.

2.3 Java

Similarly we will use Google's Java style guide.

2.4 Swift

For Swift we will use Apple design guidelines.

3 Git Structure

3.1 Branching

Rule one, always, is never work out of `master`. A new branch is created when someone wants to implement, or experiment with something that is not theirs, or in the `master`, so they will branch off of the relevant branch, to make their proposed changes.

3.1.1 Naming

A branch shall never contain an individuals name, rather it must be informative as to what the intention of the branch is, however, it is in the best interests of readability to keep the name short. An example of a terrible name is `Joe`; another is `JoeWebsiteChanges`, where the name contains a member name, and gives very little information as to what is going on in the branch. Bad, but barely acceptable names are `WebsiteChanges`, or `WebsiteExperimentation`. Ideal names are as follows: in the case of a branch where the website is ultimately being assembled, by merging other branches into it, and the only modifications taking place are to structure the files, or a quick fix, such as changing the colour of a button, `Website`; in the case that a new homepage is created for the website, where the homepage is created, and merged into `Website` on completion, `Website-Homepage`; finally, if infamous *Joe*, wants to propose a change to the `Website-Homepage` branch, where he wants to change the colour of a button, but Joe has not been assigned, or is not the primary benefactor to `Website-Homepage`, he creates a new `Website-Homepage-ButtonColourChange` branch. Notice that in the ideal examples it is possible to track down the source of a branch, to understand exactly what it's purpose is.

3.2 Merging

After changes have been made on the branch, the creator, who wants to merge, shall then make a pull request on GitHub, and announce the request on Slack, after which it shall become the responsibility of the entire team, but more importantly that of the original branch maintainer to analyze and comment on the changes; the exact location of the discussion is trivial, however GitHub provides a better, and more concrete platform for this discussion to take place, where inline comments can be made. After a sufficient amount of time and discussion has passed, either the branch maintainer (if it is a minor change), or the entire team (if it is to master, or a major change) shall decide to merge or not.

3.3 The Bulleted Rules

- *Never* make modifications directly to `master`.

- A `Developer` branch is used, to act in much the same way as `master`, but as a proxy, so that before merging that branch to master, the entire system can be reviewed there before a more solid commitment is made to `master`.
- Never directly make modifications to somebody else's code, rather branch, make the change, submit a pull request, then announce the request on Slack.
- If you absolutely feel the need to work in a branch that you did not create, message the branch creator and ask for permission.
- Never approve your own pull request, let the team leader, or the active benefactor to the branch you want to merge with make the merge.
- Allow a sufficient amount of time and discussion to pass before merging—this applies to any merge, no matter how small, the key is in judging the time and discussion that needs to pass: the larger the merge, the larger the discussion.
- Before merging, every member involved with the issue must comment, even if it is to say "yes." If someone doesn't comment, follow up with them.
- Before making a pull request, ensure that there are currently no pull requests to merge into the branch that you want to have merged with the other branch. Example: you want to merge `Website-Homepage` into `Website`, but there is a pull request to merge `Website-Homepage-ButtonColourChange` into `Website-Homepage`; in this instance, the later, active, pull request (`Website-Homepage-ButtonColourChange` into `Website-Homepage`) must be delt with first.
- Once a branch has been merged into it's source, and the branch is no longer needed, delete it.
- If you decide to abandon a branch that you created, and nobody else is working on, delete it.
- If you suspect that somebody else's branch is abandoned, reach out to them and offer to delete it.

- Never put your own name in the name of a branch.
- Give branches an informative name.
- Name the branch such that the branch it is modifying can be identified.