Project Setup and Coding Best Practices Guide

Version: 2013.11.20

This document includes best practices around setting up an initial hybrid application, and general coding techniques. Details on each checklist item are discussed later in the document.

Coding Checklist

- ❑ Adhere to style guidelines
- ❑ Use consistent directory and file names
- ❑ Always write to the latest standards
- ❑ Code Editing
- ❑ Source Code Control

Discussion

Coding Style guides

Writing clean and maintainable code should be a priority for every developer. Sloppy programming practices is unprofessional, and shows disdain for your fellow programmers.

Each of the technology areas have a section that provides sample style guides. Organizations should standardize on acceptable style guides. Consistent coding standards improves readability, comprehension, uniformity, and maintainability

Your company should adopt a standard style guide for coding in various languages. You typically do not need to get overly picky with styling, but having rules in place will foster a cohesive feel for your applications, and aid in general maintenance.

The most important points on styling are:

- Your code should be easy to understand
- Comment complex logic, or better yet, rewrite it to be less cryptic

Listed below are a few good style guides. You can mix and match these recommendations to meet your group collective aesthetic requirements.

- Dojo Style Guide
- JQuery Style Guide
- Idiomatic JS - Good general JavaScript coding guidelines
- AirBnB Style Guide
- SemanticUI Style Guide

Use consistent directory and file names

Organizations should define standards around the following areas:

- **Project names and Structure** - Some projects types, such as IBM Worklight, have a fixed project structure. Within that structure there may be company or personal conventions to define where assets are located. Contained below are generally accepted de-facto conventions that should be followed.
    - Custom application code and modules go in “/app”
    - CSS and image files should be located into a common “/theme” or “/css” directory. This makes management and optimization of image artifacts much easier than having images scattered about the app. The possible

exception is for images that are tightly associated with given widget, or content.
    - 3rd party libraries should be located into a “lib” or “resources” directory.

- **Directory names within projects**
    - Should be lowercase standard letters.
        - Numbers are ok, after first character, but should generally be avoided unless there is a valid reason
        - Do not use spaces, underscores, or hyphens
    - Other concerns deal with JavaScript modules, Java packages, and other artifacts.

- **JavaScript Modules and file naming**
    - Use only standard letters in filenames
        - Numbers are ok, after first character, but should generally be avoided unless there is a valid reason.
        - Do not use spaces, underscores, or hyphens
    - Instantiatable Classes are ProperCamelCase with leading Capital letter
    - Singleton classes are camelCase with lowercase leading letter
    - All other files should generally be camelCase, with well defined extensions.
    - Do not embed version numbers in source file names. There is rarely -- if ever -- a valid reason to do this.

Always be aware of developer/CI operating system differences. You should never use Microsoft Windows® ‘back-slash’ style directory paths anywhere in your code. Also, never use fixed paths in your code (ie C:\dev). If you must use a fully qualified directory path, then make it a configurable variable. But really, try to figure out a way to never need this. Relative paths, or even remotely access URLs are far more preferable.

Always write to the latest standards

Writing to the latest standards; HTML5, CSS3, (EMCA) JavaScript 5, will prevent you from locking yourself into the past. For older legacy browsers/platforms, use shim toolkits to bolster capabilities. Thankfully, browsers have all started taking an “evergreen: approach where they automatically receive updates.

Code Editing

Using a proper editor will greatly improve your efficiency as a developer. It really doesn't matter if you use a fully integrated development environment (IDE), or a simple text editor and command line tools for building and deploying. The trend seems to be moving in favor of normal text editors and open source command line tools as a development environment, as this can allow developers to focus on the task of actual coding without

the distractions, slow editing, and learning curve inherent in most IDE's.

Using command line tooling for build, test and deployment, goes right in hand with continuous integration. If the developers are using the exact same process flow as the CI automation, then issues can be identified much earlier.

Source Code Control

This is a must in terms of project management, backup and, and team development. There are many options, but ensure that you use an SCCS for all development. The only recommended practice is, don't deliver any changes unless it has been properly unit tested. There is nothing worse than pulling in new changes from another developer that breaks the build or kills the app. Some shops go so far as public humiliation of sloppy and uncaring developers that cause other this sort of pain. A "hat of shame" that must be worn by the scorned can work wonders to dissuade this sort of behavior.

Third party libraries should never be checked into SCCS. Utilize a package manager to ensure that all developers and CI flows are using consistent versions of libraries.

Trademarks