Mt Wilson

Hypertext

# Background

Mt Wilson is an extensible application, and there needs to be a way to integrate new features into the user interface after deployment. This blueprint describes how this is done and what a feature developer must do in order to successfully integrate the feature into the main user interface.

# Architecture

The entry point to the application is the index.html file.

A list of plugins is available and the main application in the index.html file can load available plugins and allow them to insert hooks into the user interface.

# Javascript Services

The main application guarantees to plugins that the “require” script will be available. This script is written similarly to “RequireJS” and with a similar purpose but does not require use of its module format and makes it easy to load arbitrary scripts just once.

A plugin must declare its dependency on any other scripts using the “require” mechanism.

For example, a plugin might declare dependencies on jQuery and Knockout. If they are already loaded by the main application or another plugin, the “require” mechanism will skip a redundant download.

A plugin may load HTML and CSS files using a similar mechanism “requireCSS” and “requireHTML” and write them into the page as needed. These resources are also tracked to prevent repeated downloads of the same resource.

# References

## Components

JQuery

RequireJS

Knockout

Bootstrap

# Roadmap

At this time, the “require” script does not manage dependency versions; this should be added so that if there are dependencies on different versions of the same library, the framework should either be able to 1) use semantic versioning to guess compatibility, and automatically use the first-loaded library when any other compatible version of it is requested, or 2) use the plugin’s JSON metadata descriptor to determine if it has a declared minimum required version of the library in question (in other words, known not to work with any earlier version of it) - note the minimum required version may be lower than the requested version; and 3) cancel execution of the plugin’s callback script if its dependencies are not available or conflict with already-loaded scripts, and 4) notify the administrator via the UI which plugin was not loaded due to dependency conflicts

The “require” script should be configurable with default javascript path (for example “js/”) so dependencies can write [“knockout.js”, “jquery.js”] and not have to know which directory they are in.

The “require” script should make a require function available like “RequireJS” for automatic compatibility with scripts that use that library (but without the module support).

Provide a mechanism to “force reload” a resource - just delete its cache entry and then load it.