****

**General Automation**

Get Steam (steamui) directory

Copy files over

For enabling/disabling CSS

Finding comment headers

Adding /\* or \*/ on same line

Error check if not existing

Run from executable directory

Subdirectory for css/js files

Copy only necessary files

Is this confusing?

**What goes into oldglory.py and backend.py?**

old\_glory.py contains functions that generate and interact with the UI, as well as the “first level” of functions that process input values from the GUI (eg: Checkbutton values, and validation for these Checkbuttons)

backend.py deals with functions that load config and modify files

**Install CSS Tweaks**

Copy over libraryroot.custom.css

**Box Play Button**

if CSS not already uncommented out

Uncomment Box Play Button CSS

Error detection if cannot find comment header

**Vertical Nav Button only**

if CSS not already uncommented out

Uncomment Box Play Button CSS

Error detection if cannot find comment header

**Classic Layout only**

Vertical Nav Bar must be selected as well

if CSS not already uncommented out

Uncomment Box Play Button CSS

Error detection if cannot find comment header

**Install with Dark Library (steam-library)**

Run steam-library\_compat.bat. Could later transfer the batch file over to Python

**Reload Config**

Main uses:

Reread from libraryroot.custom.css to update CSS options page

Reread from fixes.txt to update JS Options page

Run js\_tweaker or similar

1 button for all these purposes seems too much, yet having too many buttons complicates things

**Config File Format (backend)**

**CSS Options Page**

Backend:

load from libraryroot.custom.css :root section or default config, the CSS Variables as options

find start and end sections in the CSS file

error checking if missing, redirect to default config

Format is nested dictionary:

Property Group: {

Property Name: {default value, current value, options {a, b, c}, desc}

}

Frontend: Ttk Combobox entry + dropdown

Generated from backend’s loaded CSS options

.py multiple files testing vs singular .exe testing.

Sigh.

Add in python testing?

Really not feeling like doing so

Something like it may be necessary down the road.

General Maintenance

When Steam updates with a new version of JavaScript, all the minified variable names get changed.

Biggest source of maintenance, last update (Oct 8), half the JS tweaks had to change their variable names to work again

Can automate the process by wildcarding variable names?

Fortunately this doesn’t happen often enough, that automating is a necessity to avoid insanity.

Nothing gets broken, just that JavaScript tweaks stop working.

So the more JS tweaks I add, the bigger potential breakdown and loss in added functionality provided by my tweaks

v0.7 Requirements left stage

done = currently implemented

todo1 = needed for beta 1.0 patch to be released

todo2 = can be done later, but ideally done with todo1

todo3 = extension functionality to be done after todo2

**On Program Startup**

(done) Load CSS config preset values from libraryroot.custom.css :root {

(done) Load JS fixes preset values from fixes.txt

**On CSS Options Button Click**

(done) Populate GUI with selected radio buttons based on values in file

(todo1) Populate GUI with selected radio buttons based on modified values (since start)

(todo2) Show if a custom value is selected

(todo2) what that value is

(todo2) programmatically create radio button groups

**On JS Options Button Click**

(done) Populate GUI with selected Checkbuttons based on values in file

(done) Populate GUI with selected Checkbuttons based on modified values (since start)

**On Install Click**

(todo1) get current modified CSS and JS config values

**(todo1) set css and js modified according to checkboxes**

(todo2) programmatically create these for easier maintainability in the future

On Reload Config

(todo1) update GUIs with values from file, overwrite modified values

General

(todo1) Stop confusing yourself