DELIVERABLE #6: CODE INSPECTION

University of Toronto Scarborough

Authors: Brandon Chow, Daniel Persaud, Jason Zheng, Kenny Lam, Wilfred Wong

Date : Dec 3, 2015

Table of Contents

1 Code Inspection #1: citeOverlay.js

Reviewer name: Brandon Chow

Code author: Wilfred Wong

Date: November 24, 2015

Code under review: citeOverlay.js

Description:

This file's task is to create the window in which custom fields will be presented in. It creates a tab that is added item's sidebar which displays current custom fields fetched from the local database, creates another window in which custom fields can be created and removed, and saves the custom field data into the local database. This database is separate from Zotero and uses a previously created API to access this data.

Poor code logic: Inefficient Code

Location: onLoad method

Problem: The onload method contains code that is used to generate the user interface. This can be all compact into a xul or at the very least, abstracted from the backend. Only dynamic UI should be used in the backend

Suggestion: Compact the UI setup into another module or create an XUL file. Further research may be necessary to incorporate UI enhancements to previous existing UI (ie, from Zotero).

Poor code style: Code does not follow style guidelines

Location: Various locations of the file

Problem: Lines span greater than 100 characters. Typically 80 characters in length, while Zotero uses 100 characters.

Suggestion: Compact lines, make temporary variable, add newlines, rename variable for compactness.

Poor code style: Code does not follow style guidelines

Location: Various locations of the file

Problem: Inconsistent indentations. Developer switches between 4 spaces and tabs (of width 4). Zotero uses tabs only

Suggestion: Easily editable using popular IDEs today or a simple find and replace.

Poor code style: Code does not follow style guidelines

Location: Top of file

Problem: Code should comply with JavaScript strict mode, as used by Zotero

Suggestion: Add "use strict" to the top of the file.

Poor code logic: Redundant Code

Location: changeFormat method

Problem: At the end of each branch, an empty return statement is used.

Suggestion: If the "else" is not needed, developer can omit it.

Vulnerabilities: Data inputs not checked

Location: User Interface

Problem: CSL Editor allows access to XML edits without proper checking

Suggestion: While this is meant to be a feature that offers more flexibility to the end user if they have the technical skill and this does not have the potential to harm security, a warning should be provided to the user if they want access.

2 Code Inspection #2: customFields.js

Reviewer name: Wilfred Wong

Code author: Kenny Lam

Date: November 24, 2015

Code under review: customFields.js

Description:

This file's task is to create the window in which custom fields will be presented in. It creates a tab that is added item's sidebar which displays current custom fields fetched from the local database, creates another window in which custom fields can be created and removed, and saves the custom field data into the local database. This database is separate from Zotero and uses a previously created API to access this data.

Poor code logic: Dead code

Location: Functions this.onUnload and this.onBeforeUnload.

Problem: These functions are unused and can be removed.

Suggestion: Remove functions.

Poor code logic: Global variables

Location: Global variables tab and tabpanels.

Problem: These variables are not needed.

Suggestion: Remove declaration of variables

Unreadable code:

Location: Line 92

Problem: Commented out code

Suggestion: Remove the line

Vulnerabilities: Data input not checked

Location: Function hideTextbox

Problem: Field data is not checked to fit field type.

Suggestion: Validate the input to fit the field type

Vulnerabilities: Data input not checked

Location: Function hideTextbox

Problem: Field data is not checked for repeating field names

Suggestion: Validate unique field names

Poor Testing: No test files

Problem: Test files were not created

Suggestion: Write down test used to check

3 Code Inspection #3: batchEditor.js

Reviewer name: Kenny Lam

Code author: Brandon Chow

Date: November 24, 2015

Code under review: batchEditor.js

Description:

This file handles the batch editing of tags. It can search for, add, remove, and merge tags. The scope of the change can be filtered by collection, or individually selected items or the user can choose to have no filter.

Poor code logic: Global Variables

Location: Global area

Problem: Uses two global variables for configuration and to carry data between windows.

Suggestion: Can use a window.openDialog() call instead of a regular window.open() call.

Poor code style: Code does not follow style guidelines

Location: Entire file

Problem: Does not declare variables in the smallest scope possible.

Suggestion: Refactor code to move variable declarations outside of loops.

Poor code style: Code not modular

Location: Multiple functions/methods

Problem: Code lacks modularity in a lot of instances.

Suggestion: Refactor code to become more modular, easily readable, and easily modifiable.

Missing documentation: Lack of function descriptions/docstrings

Location: Multiple functions/methods

Problem: Many functions lack a description.

Suggestion: Document code

Poor code logic: Inefficient code

Location: Whole file

Problem: Relies too heavily on Zotero's search API, which does an SQL call per function call, which is incredibly slow and inefficient.

Suggestion: A code refactor on flow of data can increase performance. Another idea is to use a better data structure.

Bugs: Incorrect loading of tags

Location: onLoad method, when searching for items

Problem: Items within groups are not properly loaded into the list.

Suggestion: Check how items are being searched in Zotero. There may be a current limitation in the search that prevents searches in external libraries.

4 Code Inspection #4: dbapi.js

Reviewer name: Jason Zheng

Code author: Daniel Persaud

Date: December 2, 2015

Code under review: dbapi.js

Description: This file handles the database storage/loading for custom fields. It is able to get/set field names and field types given an item type.

Vulnerabilities: 3rd party requirement of sqlite3, Zotero has it's own DB object, no need for inclusion of another library.
Location: line 4
Fix suggestion: check DB.js in zotero xpcom/ folder in source, there is applicable methods in there.

Vulnerabilities: makeDB should be called in the init, making a DB is part of all the other methods and it doesn't make sense to make the user call makeDB after creating the object every single time. More clean to simply create the database upon intialization.
Location: init() method and makedb() method.
Fix Suggestion: let init take in an optional path, creates a db if not set at a predefined location, or checks the path if there is a db, if not add a db.

Missing Documentation: In getitems documentation, there is no information on the parameter path. Should describe acceptable paths, behaviour on faulty path pass in (exceptions, etc) and format of the pass in.
Location: getitems() method
Fix suggestion: Add doc strings/comments.

Missing Documentation: In add_field there is no information on what the function should do, nor any info on the parameter pass in
Location: add_field() method

Fix suggestion: Add doc strings for the method

5 Code Inspection #5: couple.js

Reviewer name: Daniel Persaud

Code author: Jason Zheng

Date: November 24, 2015

Code under review: couple.js

Description:

This file's task is to create a link between a document in the user's library and a group library. When a field in one document is updated, it will be reflected in the other.

Poor Documentation: Commenting

Location: Every Function except init(), notifier_callback(), updatelink(), and sync()

Problem: These functions lack sufficient comments (usually no comments present)

Suggestion: add comments.

Vulnerabilities:

Location: line 105-7

Problem: console.log() is left on all three lines and they output information to firefox console

Suggestion: Remove these lines, they're probably left over from testing.

Vulnerabilities:

Location: Line 48

Problem: console.log() has the potential to output group info to console

Suggestion: Remove the line

Poor Style: Reduced Readabiltiy

Location: line 64

Problem: commented out code is present

Suggestion: remove line

Poor Documentation: DocStrings

Location: All functions

Problem: No Docstrings (or JS equivalent of doc strings) are present

Suggestion: Add Docstrings

Poor Testing: No test files

Problem: Test files were not created

Suggestion: Write down test used to check