ScriptOven – Developer Guide

Please note, it is recommended that you first read the user guide before reading this document, to better understand the functionality of the application.

Contents:

Generating a New Build:

Whenever you make any changes to the application, you will want to regenerate the build file. You can do this using the “make_build.js” Node script. To set the options for this build process, see the “build_config.json” file.

The build script will go through all of the .js files (not including .test.js) in the scripts folder (defined in “build_config.json”), and load all of their contents. It will then do the same thing for all .css files in the styles folder (also defined in the build config). Then, once it has all the styling and scripts loaded, it will load the “template.html” file. The template file should contain a script tag and a styling tag (e.g. “{{ <tag name> }}”), and these tags will be replaced with the scripts/styles content.

Application Functions:

<u>Event Handlers:</u>

inTextFileInputEventListener(e)

The event listener for the input text file upload. This reads the text file that was uploaded, and places the content in the input textarea.

downloadFilenameInputKeyUp(e)

A keyup event for when the download filename is changed. This will change the outTextDownloadLink download filename. Better to use keyup, than onchange, to be sure that the filename is always updated correctly, if focus is only lost by clicking on the download button.

runButtonOnClick(e)

The event listener for the click of the “Run Function” button. Will start the process of loading in the script and running the user-provided functions.

<u>Assertion Testing:</u>

testAssertions(assertionArray, mainFunc, optionsObj)

This function takes 3 parameters.

The first is an array of assertions (this will be an array of arrays, and each array will have 2 values. The first value is the text to be inputted to main(), and the second is the expected output from main()):

E.g. assertionArray = [["input 1", "expected output 1"], ["input 2", "expected output 2"]];

The second parameter is a reference main() function.

The third parameter is an object containing the user's defined options. If a required option hasn't been provided, its default value is used.

Options:
- alertWhenAssertionsFailed (default value: true)
- consoleLogWhenAssertionsFailed (default value: false)

This function will return a string. If all assertions pass, the string will be "pass". Otherwise, the error text will be returned.

generateFailedAssertionsResult(failedTestResults)

This takes an array of arrays. In each array, the first value is the expected result, and the second value is the result main() actually returned.

This will return a string to be displayed as error text.

Helper Functions – String to Array:

lineArrayToMultilineString(lines)

Takes an array of string values, and converts them to one string, separated by newlines.

stringToLineArray(stringToSplit)

Takes a string (with multiple lines), and returns an array where each line is an item.

Helper Functions – CSV to Objects:

convertStringToCSVCompliantString(valueToConvert)

There are certain formatting rules that should be followed in CSV files.

csvLineToStringArray(line)

Converts a line taken from a CSV file into an array of string values.

csvToObjectsArray(csvLines, headers = [])

Takes an array of strings (each one being a line of CSV data).

Optionally, can also take an array of strings to be treated as the column headers. If the headers array is not provided, the first line of the CSV array will be used as the headers instead.

This will return an array of objects, where the property names are the column headers, and the data is mapped to the correct properties.

If one of the passed csv lines doesn't contain enough fields, this will throw an exception.

objectsToCSVLinesArray(orderedHeaders, objArrayToConvert)

This will take 2 parameters. The first is an array of headers. The second is the array of objects to be converted to an array of CSV lines (as strings), which will then be returned.

objectToCSVLine(orderedHeaders, objectToConvert)

This will take 2 parameters. The first is an array of headers. The second is the object to be converted to a CSV line.

This will return a CSV formatted string of the object's values, in the order defined in the orderedHeaders array (all data values will need to be of a type that inherits from Object -- which is most types).

This may display an alert (or do whatever actions handleError() is performing) and throw an exception, if the given object doesn't have one of the parameters defined in orderedHeaders.

Other Functions:

handleError(errorText)

Defines the way that general errors will be handled.

mainFunctionDefinitionIsCorrect(scriptText)

Tests that the script text contains a main function, and it has the correct amount of parameters.

populateDynamicScript(scriptText)

Populates the dynamicScript element with the given script text.

runFunctionOnInputText(functionText)

Called by the FileReader callback (in runButtonOnClick()). Takes the text from the script file.

setDownloadButtonDisabledStatus()

Will check if the out-text download button should be enabled, and then set it accordingly.

updateOutTextDownloadURL()

Set the URL on the download link for the output text.