**Wisconsin Beach App Javascript Documentation**

**Introduction**

The purpose of this document is to explain the logic and the flow of the various javascript files utilized by the Wisconsin Beach App. This is intended to be a working document that will be updated as major changes are made to the javascript files. Please reference the in-code comments and documentation for the most up to date information. To save time, for here on out, javascript files will be referred to as JS.

**Basic Setup**

The JS of the project is broken up into several files so as to make it easier to read, and organize, they are named as follows (\* denotes outside library):

* *application.js*
* *beaches.js*
* *beaches\_sites\_get.js*
* *Bootstrap-dialog.js \**
* *Bootstrap.min.js \**
* *change.js*
* *dialog-polyfill.js \**
* *display.js*
* *faves.js*
* *fill.js*
* *jquery-ui.min.js \**
* *jquery.min.js \**
* *localforage.js \**
* *material.min.js \**
* *mdl-selectfield.min.js \**
* *navigation.js*
* *page\_validation.js*
* *survey.js*
* *surveyFunc.js*
* *survey\_post.js*
* *util.js*

All of these JS files makeup the backbone of the application. All library files are not edited by us and are therefore not explained below. If you need their documentation try out these links:

* Bootstrap: <https://getbootstrap.com/docs/3.3/css/>
* Material Design Lite (Google MDL): <https://getmdl.io/components/index.html>
* JQuery: <https://api.jquery.com/>
* LocalForage: <https://github.com/localForage/localForage>

**application.js**

This is the very basic file of our JS. By way of grails, all javascript files are technically compiled into one file, and the comment section at the top of application.js denotes which ones to use. Since we use the tag: ***//= require\_tree .*** We technically do not need to explicitly define other JS files. You may notice that we do this anyways. This just comes down to conventions and Grails sometimes not picking up a file when it should.

The main body of the file contains all important global variables, which all of our other user defined JS files work off of. The rest of the file completes all of the setup required before the user starts interacting with the application. Reference the *application.js* file for more specific or up to date information.

**beaches.js**

This file currently only defines the global ***beaches*** array and fills it with the correct data which is then used by other functions to autofill the beach selection page fields.

The rest of this file is commented out, as it used to define test data. However, we have not deprecated or removed this file in case we ever need to use this predefined test data again.

**beaches\_sites\_get.js**

This file performs the get request from the Wisconsin Beach Health server for Beach, Site, and Lake data. We use this get request to populate fields related to beach selection, and is also used to correctly format beach and site data for the POST requests when uploading a survey. There is also predefined test data formatted for JSON and for correct POST format in case the get request fails.

**change.js**

This file is where all functions go that handle changing fields displayed based on user input. The current functions are:

* ***OtherChange(numberField, descriptionField)***
  + Description: Setup and removal of the description field for the second param based on the value of the first param
  + *numberField:* HTML Field
  + *descriptionField:* HTML Field
* ***OtherCheckbox(otherField, descriptionField)***
  + Description: Setup and removal of the description field for the second param based on the status of the first param
  + *otherField:* HTML Field
  + *descriptionField:* HTML Field
* ***RainfallChange()***
  + Description: Setup and removal of the description field for rainfall based on it html attributes
* ***OdorChange()***
  + Description: Setup and removal of the description field for odor based on its html attributes
* ***TurbidityOrNtuChange()***
  + Description: Setup and removal of the description field for Turbidity or NTU based on its html attributes

**display.js**

All functions that are created only to display information to the user outside of working on the survey (toasts, popups, etc) are placed in this file. The current functions are:

* ***showSaveToast()***
  + Description: Display the “survey saved” toast pop up
* ***closeDrawer()***
  + Description: Closes the drawer displaying the various pages on the survey
* ***deleteCountdown()***
  + Description: Displays a delete countdown to prevent accidental deletions of surveys

**faves.js**

This file has all functions related to modifying, creating, and loading favorites working with the local forage library.

* ***loadFavorites()***
  + Description: Loads stored favorites from localforage api
* ***saveFavorites()***
  + Description: Saves changes made to favorites
* ***applyFavorites()***
  + Description: Applies favorites and populates the dropdown on the beach selection page.
* ***addFavorite()***
  + Description: Takes currently filled out beach selection fields, and creates a new favorite entry in localforage
* ***fillFavorite()***
  + Description: Applies values to the beach select fields based on the currently selected favorite site. If none selected, do nothing.

**fill.js**

This file is where all functions related to populating the beach select fields are located. The functions are as follows:

* ***fillCounties()***
  + Description: Populates the counties drop down on the beach selection page
* ***fillLakes()***
  + Description: Populates the lakes dropdown on the beach selection page
* ***fillBeaches()***
  + Description: Populates the beaches dropdown on the beach selection page
* ***fillSites()***
  + Description: Populates the sites drop down on the beach selection page
* ***tryPropogate()***
  + Description: Attempt to propagate the fields based on the previous selections in the form. In other words, display only relevant lakes for a county, only relevant beaches for a lake, and only relevant sites for a beach

**navigation.js**

This function is where all functions that aid in page navigation are located.

* ***toPage(page, toDelete)***
  + Description: Go to a specified page
  + *page:* Page to jump to
  + *toDelete:* Boolean stating whether or not to delete survey
* ***btnPrev()***
  + Description: Function for previous button behavior. Goes to page directly before current page
  + *Note: Logic checking for array out of bounds is in toPage()*
* ***btnNext()***
  + Description: Function for next button behavior
  + *Note: Logic checking for array out of bounds is in toPage()*
* ***toReview()***
  + Description: Display the Review page if the user is at the end of survey
* ***displayBtns()***
  + Description: Display buttons based upon current page
  + *Note: toPage() calls this after loading the new page and setting curPage [GLOBAL] to the new current page*

**page\_validation.js**

This file is where all functions that do any sort of page validation are located. These are the functions:

* ***myAlert(msg)***
  + Description: Displays an alert
  + *msg:* String for the alert
* ***isEmptyOrIsNonnegativeInteger(candidate)***
  + Description: Check to see if the parameter given is empty or a Non-negative integer
  + *candidate:* A string to be checked
  + Returns: boolean value, true if candidate is empty or a Non-negative integer, false otherwise
* ***isEmptyOrIsInteger(candidate)***
  + Description: Check to see if the parameter given is empty or an integer
  + *Candidate:* A string to be checked
  + Returns: boolean value, true if candidate is empty or an integer
* ***isEmptyOrIsIntegerDegree(candidate)***
  + Description: Check to see if the parameter given is empty or is an integer with the units Celsius or Fahrenheit (user selected)
  + *candidate:* A string to be checked
  + Returns: boolean value, true if candidate is empty, or an integer within the allowable values for degrees (C or F).
* ***isNumeric(val)***
  + Description: Check if a val is numerical
  + *val:* Value to be checked
  + Returns: True if val is numeric, false if val is not
* ***isEmptyOrIspH(candidate)***
  + Description: Checks to see if the parameter is empty, or a float within bounds for a PH level
  + *candidate:* Value to check
  + Returns: True if is empty or within bounds for a PH level, false if not
* ***validatePage(curPage)***
  + Description: Checks fields based on what page number is supplied. Warns if invalid input is present.
  + *curPage:* Current page to validate
  + Returns: True if the page is valid, false if any page is invalid.

**survey.js**

This file is where all functions related to creating, storing, modifying, and deleting survey objects with local forage. Functions in this file are as follows:

* ***Survey(id, data)***
  + Description: Generates a survey object with an id and data
  + *id:* The guid that identifies the survey
  + *data:* The survey data
* ***Survey.getAll(callback)***
  + Description: Get all surveys form localforage and executes a callback on results
  + *callback:* The function to be executed after retrieving surveys from localforage
* ***Surveys.add(survey, callback)***
  + Description: Adds a survey to the stored list of survey ids in localforage
  + *survey:* The survey to be added
  + *callback:* A function to be executed after adding the survey
* ***Surveys.getById(id, deferred, callback)***
  + Description: Gets a survey from localforage from an id
  + *id:* Id of survey to get
  + *deferred:* The promise that will be resolved/rejected
  + *callback:* The callback to be executed on the survey after retrieving it from localforage
* ***Surveys.remove(id, callback)***
  + Description: Removes a survey by id from localforage
  + *id:* Id of survey to remove
  + *callback:* The callback to be executed after removing survey from localforage

**surveyFunc.js**

This file is contains all functions related to modifying surveys or survey related fields. The functions located in this file are as follows:

* ***newSurvey()***
  + Description: Clears all fields and starts a new survey, navigating to the beach selection page. Sets onbeforeunload function to warn user before refreshing page.
* ***saveSurvey()***
  + Description: Saves current survey information to localforage
* ***loadSurvey(id)***
  + Description: Loads a survey from local forage
  + *id:* Id of survey to be loaded
* ***getSurveys()***
  + Description: Create html tags for the two sections, unsubmitted and submitted and populate the appropriate html lists with surveys on the home screen
* ***uploadSelected()***
  + Description: Uploads all selected submitted surveys from the homepage
* ***downloadCSV()***
  + Description: Creates a csv of the current survey, and downloads it to the users device
* ***downloadSelected()***
  + Description: Creates a csv of the currently selected and submitted surveys on the homepage
* ***deleteSurvey()***
  + Description: Remove the current survey from the webapp and local forage
* ***deleteSelected()***
  + Description: Remove the currently selected surveys from the webapp and local forage by calling *deleteSurvey()* multiple times
* ***submit()***
  + Sets the submission value of the survey to true. It saves the survey, concatenates the comments (for purposes of uploading to WI Beach health server), downloads the current survey and redirects the user to the homepage.

**survey\_post.js**

This file handles posting to the WI Beach health server. It handles uploading surveys in JSON formatted to match the WI Beach health servers specifications.

**utility.js**

This file contains all utility functions, or functions that do not have a better home or do not belong in application.js.

* ***guid()***
  + Description: Generate a random “globally unique identifier” for identifying surverys
* ***getDateFormatted(date)***
  + Description: Take in a date and returns a formatted string for surveys in the homepage
  + *date:* Date to format
* ***dateToLocalDate(d, isDisplay)***
  + Description: Changes a date to the localized date
  + *d:* Date to localize
  + *isDisplay:* Boolean describing whether or not the date will be used as a displayed date. Server dates include seconds and milliseconds while the button versions do not.
* ***checkDirtyNumber(e)***
  + Description: Checks to see if a number has been modified
  + *e:* Window event to check.
* ***completePage(nextPage)***
  + Description: Checks to see if a page is completed and marks it as such
  + *nextPage:* The index of the next page to jump to
* ***getAllFields()***
  + Description: Get each field in the survey and return the entire thing as a map. This will include undefined fields, and is used to build the csv and the JSON post payloads
* ***clearAllFields()***
  + Description: Sets the value of each field in the survey to its default value. This includes blank for text entry, a default radio button, or a default drop down selection
* ***updateSeq(input, list, stored)***
  + Description: Called when using selectfield boxes (mostly on the beach selection page), to populate other fields on the page based on the data given.
  + *input:* Selector to find the value of the input to check
  + *list:* List of elements to check as valid
  + *stored:* Elements value to change based on given data
* ***collectSampleNow()***
  + Description: Creates and sets two versions of a new Date instance. This creates two versions, the first version is for the browser to display, the second version is for the server to store.
* ***clearBeachFields()***
  + Description: Clears all fields on the beach selection page
* ***concatComments()***
  + Description: Collects all comment fields from the survey and puts the values into their respective domain sections