# 

# conferenceManager.js Documentation

### function startConferenceManager()

initializes the conference and event forms and hides them, then loads the conference chooser, and the edit, view and delete buttons

### function initializeConferenceForm()

function initializeConferenceForm. Dynamically generate the html for the conference form, used for editing and creating conferences This html could be hard coded into the index.php file in the admin folder, but was never re-written. This function uses the code from generateHTML.js, which just returns strings of html. data validation was never done on this form, so the regular expression at the end of each element call was left blank. Populates the div with the id of conferenceFormRegion in index.php.

### function initializeEventForm()

Dynamically generate the html for the event form, used for editing and creating events. This html could be hard coded into the admin/index.php file, but was never re-written. This function uses the code from generateHTML.js, which just returns strings of html. Data validation was never done on this form, so the regular expression at the end of each create element call was left blank. Populates the div with the id of eventFormRegion in index.php.

### function resetForm(event)

This function is a helper function for the event and conference forms. Checks with the user if they want to reset their form or not, bound to the reset buttons in the conference and event forms. If false is returned, the default event handler for the reset button will not be fired.

* **Parameters:** event — Event — - The javascript click event.
* **Returns:** boolean — - Either true if the user clicks yes, or false if no.

### function setupAjaxForConferenceNames()

makes an ajax call to get data from the conference table. Uses the proxy to insert the adminID into the query such that only conferences associated with this adminID will be returned. On success initializeConferenceChooser is called, and is given the return data, on failure, catchEmptyValue is called

### function catchEmptyValue(error)

This function is called on failure of the ajax call to the proxy. A check is made if the http status code is 204, which means no data was found. If the status code is 204 just pass an empty array to the function call initializeConferenceChooser. This is done to make sure that the initializeConferenceChooser is called, if this function was not called on empty value, then no controls would be created at all. if any status code other than 200 is returned, it will hit this fail function

* **Parameters:** error — Error — - the javascript ajax error object

### function initializeConferenceChooser(data)

Generates the html for the list box that contains all of the conferences that are associated with this adminID. Uses the data returned from the ajax call to create options that have text for the conference name. Populates the divs with the ids of headingRegion1, and mainContentRegion1.

* **Parameters:** data — string[] — - array object from ajax

### function getSelectedConference(event)

This is the event handler for the submit button for the conference chooser. Gets the conference that the user chose, and makes the ajax call to get the conference information for that conference.

* **Parameters:** event — Event — - javascript onclick Event object

### function setupAjaxForConferenceInformation(conferenceName)

Makes ajax call to search the conference table where the conference\_name is equal to the user’s chosen conference. Conference names must be unique, so one result should only ever be returned. On success, getConferenceEditor is called

* **Parameters:** conferenceName — string — - string, the name of the conference to query for

### function getConferenceEditor(data)

The success callback from the ajax call for getting a conference’s information. This displays the conference information, and binds a callback to an edit conference button. The information for the conference data could be formatted nicer, however a very small number of people will actually have access to what the admin dashboard looks like. The ajax call for the conference events is called here.

* **Parameters:** data — string[][] — - The array that is returned from ajax call that should contain only one record with all of the conference information for the requested conference.

### function returnToConferenceChooser(event, message)

Prompts the user if he or she would like to return to the main menu (conference chooser). Calls setupAjaxForConferenceNames if user input is yes.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + message- — string — string, the message to prompt the user with

### function returnToSelectedConference(event, conferenceName, message)

Prompts the user if they would like to return back to the previously viewed conference page. Calls setupAjaxForConferenceInformation if the user says yes.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + conferenceName — string — - string, the name of the conference to view information for
  + message — string — - string, the message to prompt the user with

### function setupAjaxForEventInformation(conferenceID, conferenceName)

Queries the database for all events for the given conferenceID. Calls createEventEditor on success to generate the html for the event table.

* **Parameters:** conferenceID — int — - A numeric value that uniquely identifys a conference @param {string} conferenceName - The name of the conference associated with the conferenceID

### function createEventEditor(data, conferenceName)

Generates a table containing all of the events for the requested conference, and creates edit and delete controls for each event. Populates the div with the id mainContentRegion2, sinse the div with the id mainContentRegion1 already contains the conference information.

* **Parameters:**
  + data — string[][] — - The array returned from the ajax call for querying for conference events
  + conferenceName — string — - The name of the conference these events are associated with

### function collectFormData(controlsClassName, attrs, values)

This will gather the data from a form that has been submited into an array of data names (attrs), and data values (values), based on the given class name, which must exist on each control that will be accounted for in this function. The conference and event forms use this function, and both have class names on each input control spacific to their form. This function also depends on the custom attribute data-name to be on each control, which is the name in the database associated with each control.

* **Parameters:**
  + controlsClassName — string — - a html class name that is associated with each control in the form that data is being collected for
  + attrs — string[] — - The array of names that identify the collected data. initially empty, and filled with the values of the custom attribute data-name
  + values — string[] — - The array of values from the form. Initially empty, and filled with the values from each form control

### function populateFormData(controlsClassName, data)

Populates the form controls that are associated with the provided class name with the provided data. Both the conference and event forms have class names associated with each input control. This is used for editing either events or conferences, and loading the currently existing data that is associated with the event or conference.

* **Parameters:**
  + controlsClassName — string — - A html class name that is associated with each control in the form that data is being collected for
  + data — string[] — - A 1d array returned from an ajax call to retrieve the data to fill the targeted form

### function setupConferenceFormForInserting(event)

Perform the necessary actions to prepare the conference form for inserting a new conference. Clear all regions, populate the div header with the id insertHeading2, reset the form, show the form, and associate the appropriate onclick handler for inserting a conference for the submit button.

* **Parameters:** event — Event — - javascript onclick Event object

### function setupConferenceFormForUpdating(event, data)

Perform the necessary actions to prepare the conference form for updating an existing conference clear all regions, populate the div header with the id insertHeading2, reset the form, show the form, associate the appropriate onclick handler for inserting a conference for the submit button, and populate each input control with the provided data.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + data — string[] — - The 1d array returned from the ajax request whichc retrieved the conference data to fill this form

### function insertConference(event)

This is the initial onclick function to handle inserting a new conference into the database. Makes a GET ajax request to chek if the conference name given already exists in the database before inserting. This function is called on click of the save conference information button, and prevents the default action of the submit button, so that the page does not refresh. On success, call checkIfConferenceNameExists.

* **Parameters:** event — Event — - javascript onclick event

### function checkIfConferenceNameExists(data)

If there is no conference with that name, call processConferenceInsertion and continue, otherwise, show a javascript alert box to the user notifying them that they can not choose that conference name.

* **Parameters:** data — string[][] — - The array returned from an ajax get request which should contain 1 or 0 records.

### function processConferenceInsertion()

on success, call createdConferenceSuccessfully

### function createdConferenceSuccessfully(data)

* **Parameters:** data — string — - The success message back from the ajax post call from inserting the new conference

### function updateConferenceInformation(event, conferenceID)

Makes a put ajax request to update the existing record in the database with the new provided data. This function is called onclick of the update conference information button, and prevents the default action of the submit button, so that the page does not refresh. On success, call updatedConferenceSuccessfully.

* **Parameters:**
  + event — Event — - javascript onclick event
  + conferenceID — int — - A numeric value that uniquely identifys a conference to update

### function updatedConferenceSuccessfully(data)

* **Parameters:** data — string — - the success message back from the ajax put call from updating the existing conference

### function ajaxSetupForDeleteConference(event)

Makes an ajax get request to retrieve the conferenceID of the given conferenceName, to be used in querying the event table. Gets the conference name from the conference chooser list box on the front page.

* **Parameters:** event — Event — - javascript onclick Event object

### function deleteConferenceAndEvents(data)

On success of deletion of all of the events, and the conference records, call setupAjaxForConferenceNames, right after alerting the user.

* **Parameters:** data — string[][] — - the array returned from an ajax get request, wich retrieved the conferenceID for the selected conference. Should return only 1 record

### function setupEventFormForInserting(event, conferenceID, conferenceName)

Clear all regions, populate the div header with the id insertHeading2, reset the form, show the form, and associate the appropriate onclick handler for inserting a event for the submit button.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + conferenceID — int — - A numberic value that uniquely identifies a conference @param {string} conferenceName - The name of the conference associated with the given conferenceID

### function setupEventFormForUpdating(event, conferenceEvents, conferenceName)

Clear all regions, populate the div header with the id insertHeading2, reset the form, show the form, associate the appropriate onclick handler for updating an event for the submit button, find which event has the associated eventID that is being targeted, and populate each input control with the provided data.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + conferenceEvents — string[][] — - the array returned from the ajax get request, made when creating the event table
  + conferenceName — string — - the name of the conference that is associated with the targeted event

### function insertConferenceEvent(event, conferenceID, conferenceName)

Makes a GET ajax request to chek if the event name given already exists in the database before inserting. This function is called on click of the save event information button, and prevents the default action of the submit button, so that the page does not refresh. On success, call checkIfEventNameExistsInConference

* **Parameters:**
  + event — Event — - javascript onclick event
  + conferenceID — int — - A numberic value that uniquely identifies a conference @param {string} conferenceName - The name of the conference associated with the given conferenceID that is associated with the new event

### function checkIfEventNameExistsInConference(data, conferenceID, conferenceName)

If there is no event with that name, call processEventInsertion and continue, otherwise, show a javascript alert box to the user notifying them that they can not choose that event name.

* **Parameters:**
  + data — string[][] — - the array returned from an ajax get request which should contain 1 or 0 records.
  + conferenceID — int — - A numberic value that uniquely identifies a conference @param {string} conferenceName - The name of the conference associated with the given conferenceID that is associated with the new event

### function processEventInsertion(conferenceID, conferenceName)

On success, call createdEventSuccessfully.

* **Parameters:** conferenceID — int — - a numberic value that uniquely identifies a conference @param {string} conferenceName - The name of the conference associated with the given conferenceID that is associated with the new event

### function createdEventSuccessfully(data, conferenceName)

* **Parameters:**
  + data — string — - the success message back from the ajax post call from inserting the new conference
  + conferenceName — string — - the name of the conference that is associated with the new event

### function updateConferenceEvent(event, eventID, conferenceName)

Makes a put ajax request to update the existing record in the database with the new provided data. This function is called onclick of the update event information button, and prevents the default action of the submit button, so that the page does not refresh. On success, call updatedEventSuccessfully.

* **Parameters:**
  + event — Event — - javascript onclick event
  + eventID — int — - A numeric value that uniquely identifys an conference event to update
  + conferenceName — string — - The name of the conference that is associated with the existing event

### function updatedEventSuccessfully(data, conferenceName)

* **Parameters:**
  + data — string — - the success message back from the ajax put call from updating the existing conference
  + conferenceName — string — - The name of the conference that is associated with the existing event

### function deleteConferenceEvent(event)

Makes an ajax delete call to delete the chosen event. On success, call deletedEventSuccessfully.

* **Parameters:** event — Event — - javascript onclick Event object

### function deletedEventSuccessfully(data, deleteButton)

then dynamically removes the associated row in the html table by getting the grandparent of the deleteButton, which is the html tr object, finds the index, and deletes the row from the table by that index. The table is found by getting a list of all tables on the page, and making the asumption that there is only one table on the page. This could be a problem in the future.

* **Parameters:**
  + data — string — - the success message from deleting a record from the database
  + deleteButton — Button — - the button dom object that was triggered to delete the event.

### function clearAllRegions()

Removes all potential event handlers from buttons, clears the contents from each region, and hides the conference and event forms. Not all of these actions are necessary all the time, however, none of these actions will throw an error if they are unneeded.

# 

# databaseFunctions.js Documentation

### function getRecord(valuesToSelect, tableNames, attrs, values, callback, type, formatFlag, orderBy, proxyflag)

* **Parameters:**
  + valuesToSelect — Array<String> — - The name(s) of the columns to return.
  + tableNames — Array<String> — - The name(s) of the tables to GET from. Entering multiple table names results in a Natural Join.
  + attrs — Array<String> — - The name(s) of the columns to select by
  + values — Array<String> — - The value(s) of the columns to select by
  + callback — Function — - The function that will execute after the GET request finishes.
  + type — String — - The data type to be returned. Most common use here is “json”.
  + formatFlag — String — - A string that determines if you wish to format the input values. Deprecated.
  + orderBy — Array<String> — - The name(s) of the columns to sort the returned array by. Multiple names can be entered.
  + proxyflag — Boolean — - A boolean (true or false) that determines if you want to use the Proxy.

### function delRecord(tablename, idname, idvalue, callback, proxyflag)

* **Parameters:**
  + tablename — String — - The name of the table.
  + idname — Array<String> — - The name(s) of the columns to search by
  + idvalue — Array<String> — - The value(s) of the data you are searching for
  + callback — Function — - The function that will execute after the DELETE request finishes.
  + proxyflag — Boolean — - A boolean (true or false) that determines if you want to use the Proxy.

### function postRecord(tablename, attrs, values, callback, formatFlag, proxyflag)

* **Parameters:**
  + tablename — String — - The name of the table.
  + attrs — Array<String> — - The name(s) of the columns to select by
  + values — Array<String> — - The value(s) of the columns to select by
  + callback — Function — - The function that will execute after the POST request finishes.
  + formatFlag — String — - A string that determines if you wish to format the input values. Deprecated.
  + proxyflag — Boolean — - A boolean (true or false) that determines if you want to use the Proxy.

### function putRecord(tablename, attrs, values, idname, idvalue, callback, formatFlag, proxyflag)

* **Parameters:**
  + tablename — String — - The name of the table.
  + attrs — Array<String> — - The name(s) of the columns to select by
  + values — Array<String> — - The value(s) of the columns to select by
  + idname — Array<String> — - The name(s) of the columns to put new data into
  + idvalue — Array<String> — - The value(s) of the data you are putting
  + callback — Function — - The function that will execute after the PUT request finishes.
  + formatFlag — String — - A string that determines if you wish to format the input values. Deprecated.
  + proxyflag — Boolean — - A boolean (true or false) that determines if you want to use the Proxy.

# 

# fileRecursor.js Documentation

# 

# generateHTML.js Documentation

### function createForm(id)

Creates html for an empty form and gives the form an id.

* **Parameters:** id — string — - The id value to be added to the id attribute for the form element.
* **Returns:** string — the html of the empty form

### function createTextbox(label, id, className, dataName, regex, isRequired)

Creates an html label and textbox. Constructs the html attributes based on the given parameters.

* **Parameters:**
  + label — string — - the text that is used to label the textbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
  + regex — string — - A regular expression value for the custom attribute data-verify, used to verify the user data.
  + isRequired — boolean — - A boolean value that is used for determining if the textbox is required or not.
* **Returns:** string — - the html string of the textbox

### function createPhoneTextbox(label, id, className, dataName, regex)

Creates an html label and phone textbox. Constructs the html attributes based on the given parameters. Uses a value of phone for the type attribute rather than text.

* **Parameters:**
  + label — string — - the text that is used to label the textbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
  + regex — string — - A regular expression value for the custom attribute data-verify, used to verify the user data.
* **Returns:** string — - the html for the phone textbox

### function createEmailTextbox(label, id, className, dataName, regex)

Creates an html label and email textbox. Constructs the html attributes based on the given parameters. Uses a value of email for the type attribute rather than text.

* **Parameters:**
  + label — string — - the text that is used to label the textbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
  + regex — string — - A regular expression value for the custom attribute data-verify, used to verify the user data.
* **Returns:** string — The html for the email textbox

### function createTextarea(label, id, className, dataName, regex)

Creates an html label and textarea. Constructs the html attributes based on the given parameters.

* **Parameters:**
  + label — string — - the text that is used to label the textarea.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
  + regex — string — - A regular expression value for the custom attribute data-verify, used to verify the user data.
* **Returns:** string — - the html for the textarea

### function createRadioButtons(legendName, radioButtonInfo, name, className, dataName)

Creates the html for a group of radio buttons. Constructs the attributes and elements based on the provided parameters.

* **Parameters:**
  + legendName — string — - The name of the radio button group
  + radioButtonInfo — string[][] — - A 2d array that contains the information about each radio button. Each sub-array must contain: id, label, and value.
  + name — string — - the value for the name attribute assigned to each radio button.
  + className — string — - The value that is for the class attribute assigned to each radio button.
  + dataName — string — - The value associated with the custom attribute data-name, which defines the name of the field in the database.
* **Returns:** string — - The html string that is the radio button group.

### function createButton(label, type, id, className)

Creates the html for a button. Constructs the attributes based on the given parameters.

* **Parameters:**
  + label — string — - the value that is associated with the value attribute for labeling this button.
  + type — string — - Defines the type of button, usually only button, or submit.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
* **Returns:** string — - the html string of the requested button.

### function insertHeading2(value, id)

Creates the html for a heading level 2, then inserts the element into the region with the provided id.

* **Parameters:**
  + value — string — - The text for the heading
  + id — string — - The id of the region that this heading is being inserted into.

### function insertHeading3(value, id)

Creates the html for a heading level 3, then inserts the element into the region with the provided id.

* **Parameters:**
  + value — string — - The text for the heading
  + id — string — - The id of the region that this heading is being inserted into.

### function createListBox(label, options, id, className, dataName)

Creates html for a listbox. constructs the elements and attributes based on the provided parameters.

* **Parameters:**
  + label — string — - the text that is used to label the listbox.
  + options — string[] — - An array of strings that are used as the text values for each option in the listbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
* **Returns:** string — The html for the listbox

### function createListOfStates(label, id, className, dataName)

Creates html for a listbox of states. The options are already hard coded with all of the states as options. constructs the elements and attributes based on the provided parameters.

* **Parameters:**
  + label — string — - the text that is used to label the listbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
* **Returns:** string — The html for the listbox

### function createListOfCountries(label, id, className, dataName)

Creates html for a listbox of countries. The options are already hard coded with all of the countries as options. constructs the elements and attributes based on the provided parameters.

* **Parameters:**
  + label — string — - the text that is used to label the listbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
* **Returns:** string — The html for the listbox

# 

# javadoc-to-markdown.js Documentation

### var JavadocToMarkdown = function (fName)

Generate Markdown from your Javadoc, PHPDoc or JSDoc comments

Usage: Create a new instance of JavadocToMarkdown and then call either fromJavadoc(), fromPHPDoc() or fromJSDoc()

* **Constructor**

### function fromDoc(code, headingsLevel, fnAddTagsMarkdown)

Generates Markdown documentation from code on a more abstract level

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
  + fnAddTagsMarkdown — function — the function that processes doc tags and generates the Markdown documentation
* **Returns:** string — the Markdown documentation

### this.fromStaticTypesDoc = function(code, headingsLevel)

Generates Markdown documentation from a statically typed language’s doc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
* **Returns:** string — the Markdown documentation

### this.fromDynamicTypesDoc = function(code, headingsLevel, fnFormatType, fnFormatTypeAndName)

Generates Markdown documentation from a dynamically typed language’s doc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
  + fnFormatType — function — the function that formats a type information (single argument)
  + fnFormatTypeAndName — function — the function that formats type and name information (two arguments)
* **Returns:** string — the Markdown documentation

### this.fromJavadoc = function(code, headingsLevel)

Generates Markdown documentation from Javadoc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
* **Returns:** string — the Markdown documentation

### this.fromPHPDoc = function(code, headingsLevel)

Generates Markdown documentation from PHPDoc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
* **Returns:** string — the Markdown documentation

### this.fromJSDoc = function(code, headingsLevel)

Generates Markdown documentation from JSDoc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
* **Returns:** string — the Markdown documentation

### function fromSection(section, headingsLevel, fnAddTagsMarkdown)

Generates Markdown documentation from a given section

The function processes units of documentation, a line of code with accompanying doc comment

* **Parameters:**
  + section — object — the section that consists of code line and doc comment
  + headingsLevel — number — the headings level to use as the base (1-6)
  + fnAddTagsMarkdown — function — the function that processes doc tags and generates the Markdown documentation
* **Returns:** string — the Markdown documentation

# 

# conferenceManager.js Documentation

### function startConferenceManager()

initializes the conference and event forms and hides them, then loads the conference chooser, and the edit, view and delete buttons

### function initializeConferenceForm()

function initializeConferenceForm. Dynamically generate the html for the conference form, used for editing and creating conferences This html could be hard coded into the index.php file in the admin folder, but was never re-written. This function uses the code from generateHTML.js, which just returns strings of html. data validation was never done on this form, so the regular expression at the end of each element call was left blank. Populates the div with the id of conferenceFormRegion in index.php.

### function initializeEventForm()

Dynamically generate the html for the event form, used for editing and creating events. This html could be hard coded into the admin/index.php file, but was never re-written. This function uses the code from generateHTML.js, which just returns strings of html. Data validation was never done on this form, so the regular expression at the end of each create element call was left blank. Populates the div with the id of eventFormRegion in index.php.

### function resetForm(event)

This function is a helper function for the event and conference forms. Checks with the user if they want to reset their form or not, bound to the reset buttons in the conference and event forms. If false is returned, the default event handler for the reset button will not be fired.

* **Parameters:** event — Event — - The javascript click event.
* **Returns:** boolean — - Either true if the user clicks yes, or false if no.

### function setupAjaxForConferenceNames()

makes an ajax call to get data from the conference table. Uses the proxy to insert the adminID into the query such that only conferences associated with this adminID will be returned. On success initializeConferenceChooser is called, and is given the return data, on failure, catchEmptyValue is called

### function catchEmptyValue(error)

This function is called on failure of the ajax call to the proxy. A check is made if the http status code is 204, which means no data was found. If the status code is 204 just pass an empty array to the function call initializeConferenceChooser. This is done to make sure that the initializeConferenceChooser is called, if this function was not called on empty value, then no controls would be created at all. if any status code other than 200 is returned, it will hit this fail function

* **Parameters:** error — Error — - the javascript ajax error object

### function initializeConferenceChooser(data)

Generates the html for the list box that contains all of the conferences that are associated with this adminID. Uses the data returned from the ajax call to create options that have text for the conference name. Populates the divs with the ids of headingRegion1, and mainContentRegion1.

* **Parameters:** data — string[] — - array object from ajax

### function getSelectedConference(event)

This is the event handler for the submit button for the conference chooser. Gets the conference that the user chose, and makes the ajax call to get the conference information for that conference.

* **Parameters:** event — Event — - javascript onclick Event object

### function setupAjaxForConferenceInformation(conferenceName)

Makes ajax call to search the conference table where the conference\_name is equal to the user’s chosen conference. Conference names must be unique, so one result should only ever be returned. On success, getConferenceEditor is called

* **Parameters:** conferenceName — string — - string, the name of the conference to query for

### function getConferenceEditor(data)

The success callback from the ajax call for getting a conference’s information. This displays the conference information, and binds a callback to an edit conference button. The information for the conference data could be formatted nicer, however a very small number of people will actually have access to what the admin dashboard looks like. The ajax call for the conference events is called here.

* **Parameters:** data — string[][] — - The array that is returned from ajax call that should contain only one record with all of the conference information for the requested conference.

### function returnToConferenceChooser(event, message)

Prompts the user if he or she would like to return to the main menu (conference chooser). Calls setupAjaxForConferenceNames if user input is yes.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + message- — string — string, the message to prompt the user with

### function returnToSelectedConference(event, conferenceName, message)

Prompts the user if they would like to return back to the previously viewed conference page. Calls setupAjaxForConferenceInformation if the user says yes.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + conferenceName — string — - string, the name of the conference to view information for
  + message — string — - string, the message to prompt the user with

### function setupAjaxForEventInformation(conferenceID, conferenceName)

Queries the database for all events for the given conferenceID. Calls createEventEditor on success to generate the html for the event table.

* **Parameters:** conferenceID — int — - A numeric value that uniquely identifys a conference @param {string} conferenceName - The name of the conference associated with the conferenceID

### function createEventEditor(data, conferenceName)

Generates a table containing all of the events for the requested conference, and creates edit and delete controls for each event. Populates the div with the id mainContentRegion2, sinse the div with the id mainContentRegion1 already contains the conference information.

* **Parameters:**
  + data — string[][] — - The array returned from the ajax call for querying for conference events
  + conferenceName — string — - The name of the conference these events are associated with

### function collectFormData(controlsClassName, attrs, values)

This will gather the data from a form that has been submited into an array of data names (attrs), and data values (values), based on the given class name, which must exist on each control that will be accounted for in this function. The conference and event forms use this function, and both have class names on each input control spacific to their form. This function also depends on the custom attribute data-name to be on each control, which is the name in the database associated with each control.

* **Parameters:**
  + controlsClassName — string — - a html class name that is associated with each control in the form that data is being collected for
  + attrs — string[] — - The array of names that identify the collected data. initially empty, and filled with the values of the custom attribute data-name
  + values — string[] — - The array of values from the form. Initially empty, and filled with the values from each form control

### function populateFormData(controlsClassName, data)

Populates the form controls that are associated with the provided class name with the provided data. Both the conference and event forms have class names associated with each input control. This is used for editing either events or conferences, and loading the currently existing data that is associated with the event or conference.

* **Parameters:**
  + controlsClassName — string — - A html class name that is associated with each control in the form that data is being collected for
  + data — string[] — - A 1d array returned from an ajax call to retrieve the data to fill the targeted form

### function setupConferenceFormForInserting(event)

Perform the necessary actions to prepare the conference form for inserting a new conference. Clear all regions, populate the div header with the id insertHeading2, reset the form, show the form, and associate the appropriate onclick handler for inserting a conference for the submit button.

* **Parameters:** event — Event — - javascript onclick Event object

### function setupConferenceFormForUpdating(event, data)

Perform the necessary actions to prepare the conference form for updating an existing conference clear all regions, populate the div header with the id insertHeading2, reset the form, show the form, associate the appropriate onclick handler for inserting a conference for the submit button, and populate each input control with the provided data.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + data — string[] — - The 1d array returned from the ajax request whichc retrieved the conference data to fill this form

### function insertConference(event)

This is the initial onclick function to handle inserting a new conference into the database. Makes a GET ajax request to chek if the conference name given already exists in the database before inserting. This function is called on click of the save conference information button, and prevents the default action of the submit button, so that the page does not refresh. On success, call checkIfConferenceNameExists.

* **Parameters:** event — Event — - javascript onclick event

### function checkIfConferenceNameExists(data)

If there is no conference with that name, call processConferenceInsertion and continue, otherwise, show a javascript alert box to the user notifying them that they can not choose that conference name.

* **Parameters:** data — string[][] — - The array returned from an ajax get request which should contain 1 or 0 records.

### function processConferenceInsertion()

on success, call createdConferenceSuccessfully

### function createdConferenceSuccessfully(data)

* **Parameters:** data — string — - The success message back from the ajax post call from inserting the new conference

### function updateConferenceInformation(event, conferenceID)

Makes a put ajax request to update the existing record in the database with the new provided data. This function is called onclick of the update conference information button, and prevents the default action of the submit button, so that the page does not refresh. On success, call updatedConferenceSuccessfully.

* **Parameters:**
  + event — Event — - javascript onclick event
  + conferenceID — int — - A numeric value that uniquely identifys a conference to update

### function updatedConferenceSuccessfully(data)

* **Parameters:** data — string — - the success message back from the ajax put call from updating the existing conference

### function ajaxSetupForDeleteConference(event)

Makes an ajax get request to retrieve the conferenceID of the given conferenceName, to be used in querying the event table. Gets the conference name from the conference chooser list box on the front page.

* **Parameters:** event — Event — - javascript onclick Event object

### function deleteConferenceAndEvents(data)

On success of deletion of all of the events, and the conference records, call setupAjaxForConferenceNames, right after alerting the user.

* **Parameters:** data — string[][] — - the array returned from an ajax get request, wich retrieved the conferenceID for the selected conference. Should return only 1 record

### function setupEventFormForInserting(event, conferenceID, conferenceName)

Clear all regions, populate the div header with the id insertHeading2, reset the form, show the form, and associate the appropriate onclick handler for inserting a event for the submit button.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + conferenceID — int — - A numberic value that uniquely identifies a conference @param {string} conferenceName - The name of the conference associated with the given conferenceID

### function setupEventFormForUpdating(event, conferenceEvents, conferenceName)

Clear all regions, populate the div header with the id insertHeading2, reset the form, show the form, associate the appropriate onclick handler for updating an event for the submit button, find which event has the associated eventID that is being targeted, and populate each input control with the provided data.

* **Parameters:**
  + event — Event — - javascript onclick Event object
  + conferenceEvents — string[][] — - the array returned from the ajax get request, made when creating the event table
  + conferenceName — string — - the name of the conference that is associated with the targeted event

### function insertConferenceEvent(event, conferenceID, conferenceName)

Makes a GET ajax request to chek if the event name given already exists in the database before inserting. This function is called on click of the save event information button, and prevents the default action of the submit button, so that the page does not refresh. On success, call checkIfEventNameExistsInConference

* **Parameters:**
  + event — Event — - javascript onclick event
  + conferenceID — int — - A numberic value that uniquely identifies a conference @param {string} conferenceName - The name of the conference associated with the given conferenceID that is associated with the new event

### function checkIfEventNameExistsInConference(data, conferenceID, conferenceName)

If there is no event with that name, call processEventInsertion and continue, otherwise, show a javascript alert box to the user notifying them that they can not choose that event name.

* **Parameters:**
  + data — string[][] — - the array returned from an ajax get request which should contain 1 or 0 records.
  + conferenceID — int — - A numberic value that uniquely identifies a conference @param {string} conferenceName - The name of the conference associated with the given conferenceID that is associated with the new event

### function processEventInsertion(conferenceID, conferenceName)

On success, call createdEventSuccessfully.

* **Parameters:** conferenceID — int — - a numberic value that uniquely identifies a conference @param {string} conferenceName - The name of the conference associated with the given conferenceID that is associated with the new event

### function createdEventSuccessfully(data, conferenceName)

* **Parameters:**
  + data — string — - the success message back from the ajax post call from inserting the new conference
  + conferenceName — string — - the name of the conference that is associated with the new event

### function updateConferenceEvent(event, eventID, conferenceName)

Makes a put ajax request to update the existing record in the database with the new provided data. This function is called onclick of the update event information button, and prevents the default action of the submit button, so that the page does not refresh. On success, call updatedEventSuccessfully.

* **Parameters:**
  + event — Event — - javascript onclick event
  + eventID — int — - A numeric value that uniquely identifys an conference event to update
  + conferenceName — string — - The name of the conference that is associated with the existing event

### function updatedEventSuccessfully(data, conferenceName)

* **Parameters:**
  + data — string — - the success message back from the ajax put call from updating the existing conference
  + conferenceName — string — - The name of the conference that is associated with the existing event

### function deleteConferenceEvent(event)

Makes an ajax delete call to delete the chosen event. On success, call deletedEventSuccessfully.

* **Parameters:** event — Event — - javascript onclick Event object

### function deletedEventSuccessfully(data, deleteButton)

then dynamically removes the associated row in the html table by getting the grandparent of the deleteButton, which is the html tr object, finds the index, and deletes the row from the table by that index. The table is found by getting a list of all tables on the page, and making the asumption that there is only one table on the page. This could be a problem in the future.

* **Parameters:**
  + data — string — - the success message from deleting a record from the database
  + deleteButton — Button — - the button dom object that was triggered to delete the event.

### function clearAllRegions()

Removes all potential event handlers from buttons, clears the contents from each region, and hides the conference and event forms. Not all of these actions are necessary all the time, however, none of these actions will throw an error if they are unneeded.

# 

# databaseFunctions.js Documentation

### function getRecord(valuesToSelect, tableNames, attrs, values, callback, type, formatFlag, orderBy, proxyflag)

* **Parameters:**
  + valuesToSelect — Array<String> — - The name(s) of the columns to return.
  + tableNames — Array<String> — - The name(s) of the tables to GET from. Entering multiple table names results in a Natural Join.
  + attrs — Array<String> — - The name(s) of the columns to select by
  + values — Array<String> — - The value(s) of the columns to select by
  + callback — Function — - The function that will execute after the GET request finishes.
  + type — String — - The data type to be returned. Most common use here is “json”.
  + formatFlag — String — - A string that determines if you wish to format the input values. Deprecated.
  + orderBy — Array<String> — - The name(s) of the columns to sort the returned array by. Multiple names can be entered.
  + proxyflag — Boolean — - A boolean (true or false) that determines if you want to use the Proxy.

### function delRecord(tablename, idname, idvalue, callback, proxyflag)

* **Parameters:**
  + tablename — String — - The name of the table.
  + idname — Array<String> — - The name(s) of the columns to search by
  + idvalue — Array<String> — - The value(s) of the data you are searching for
  + callback — Function — - The function that will execute after the DELETE request finishes.
  + proxyflag — Boolean — - A boolean (true or false) that determines if you want to use the Proxy.

### function postRecord(tablename, attrs, values, callback, formatFlag, proxyflag)

* **Parameters:**
  + tablename — String — - The name of the table.
  + attrs — Array<String> — - The name(s) of the columns to select by
  + values — Array<String> — - The value(s) of the columns to select by
  + callback — Function — - The function that will execute after the POST request finishes.
  + formatFlag — String — - A string that determines if you wish to format the input values. Deprecated.
  + proxyflag — Boolean — - A boolean (true or false) that determines if you want to use the Proxy.

### function putRecord(tablename, attrs, values, idname, idvalue, callback, formatFlag, proxyflag)

* **Parameters:**
  + tablename — String — - The name of the table.
  + attrs — Array<String> — - The name(s) of the columns to select by
  + values — Array<String> — - The value(s) of the columns to select by
  + idname — Array<String> — - The name(s) of the columns to put new data into
  + idvalue — Array<String> — - The value(s) of the data you are putting
  + callback — Function — - The function that will execute after the PUT request finishes.
  + formatFlag — String — - A string that determines if you wish to format the input values. Deprecated.
  + proxyflag — Boolean — - A boolean (true or false) that determines if you want to use the Proxy.

# 

# fileRecursor.js Documentation

# 

# generateHTML.js Documentation

### function createForm(id)

Creates html for an empty form and gives the form an id.

* **Parameters:** id — string — - The id value to be added to the id attribute for the form element.
* **Returns:** string — the html of the empty form

### function createTextbox(label, id, className, dataName, regex, isRequired)

Creates an html label and textbox. Constructs the html attributes based on the given parameters.

* **Parameters:**
  + label — string — - the text that is used to label the textbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
  + regex — string — - A regular expression value for the custom attribute data-verify, used to verify the user data.
  + isRequired — boolean — - A boolean value that is used for determining if the textbox is required or not.
* **Returns:** string — - the html string of the textbox

### function createPhoneTextbox(label, id, className, dataName, regex)

Creates an html label and phone textbox. Constructs the html attributes based on the given parameters. Uses a value of phone for the type attribute rather than text.

* **Parameters:**
  + label — string — - the text that is used to label the textbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
  + regex — string — - A regular expression value for the custom attribute data-verify, used to verify the user data.
* **Returns:** string — - the html for the phone textbox

### function createEmailTextbox(label, id, className, dataName, regex)

Creates an html label and email textbox. Constructs the html attributes based on the given parameters. Uses a value of email for the type attribute rather than text.

* **Parameters:**
  + label — string — - the text that is used to label the textbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
  + regex — string — - A regular expression value for the custom attribute data-verify, used to verify the user data.
* **Returns:** string — The html for the email textbox

### function createTextarea(label, id, className, dataName, regex)

Creates an html label and textarea. Constructs the html attributes based on the given parameters.

* **Parameters:**
  + label — string — - the text that is used to label the textarea.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
  + regex — string — - A regular expression value for the custom attribute data-verify, used to verify the user data.
* **Returns:** string — - the html for the textarea

### function createRadioButtons(legendName, radioButtonInfo, name, className, dataName)

Creates the html for a group of radio buttons. Constructs the attributes and elements based on the provided parameters.

* **Parameters:**
  + legendName — string — - The name of the radio button group
  + radioButtonInfo — string[][] — - A 2d array that contains the information about each radio button. Each sub-array must contain: id, label, and value.
  + name — string — - the value for the name attribute assigned to each radio button.
  + className — string — - The value that is for the class attribute assigned to each radio button.
  + dataName — string — - The value associated with the custom attribute data-name, which defines the name of the field in the database.
* **Returns:** string — - The html string that is the radio button group.

### function createButton(label, type, id, className)

Creates the html for a button. Constructs the attributes based on the given parameters.

* **Parameters:**
  + label — string — - the value that is associated with the value attribute for labeling this button.
  + type — string — - Defines the type of button, usually only button, or submit.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
* **Returns:** string — - the html string of the requested button.

### function insertHeading2(value, id)

Creates the html for a heading level 2, then inserts the element into the region with the provided id.

* **Parameters:**
  + value — string — - The text for the heading
  + id — string — - The id of the region that this heading is being inserted into.

### function insertHeading3(value, id)

Creates the html for a heading level 3, then inserts the element into the region with the provided id.

* **Parameters:**
  + value — string — - The text for the heading
  + id — string — - The id of the region that this heading is being inserted into.

### function createListBox(label, options, id, className, dataName)

Creates html for a listbox. constructs the elements and attributes based on the provided parameters.

* **Parameters:**
  + label — string — - the text that is used to label the listbox.
  + options — string[] — - An array of strings that are used as the text values for each option in the listbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
* **Returns:** string — The html for the listbox

### function createListOfStates(label, id, className, dataName)

Creates html for a listbox of states. The options are already hard coded with all of the states as options. constructs the elements and attributes based on the provided parameters.

* **Parameters:**
  + label — string — - the text that is used to label the listbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
* **Returns:** string — The html for the listbox

### function createListOfCountries(label, id, className, dataName)

Creates html for a listbox of countries. The options are already hard coded with all of the countries as options. constructs the elements and attributes based on the provided parameters.

* **Parameters:**
  + label — string — - the text that is used to label the listbox.
  + id — string — - An id value used in the id attribute for the element.
  + className — string — - The value used in the class attribute for the element.
  + dataName — string — - The value for the custom attribute data-name, which reflects the name of the associated field in the database the data will be inserted into.
* **Returns:** string — The html for the listbox

# 

# javadoc-to-markdown.js Documentation

### var JavadocToMarkdown = function (fName)

Generate Markdown from your Javadoc, PHPDoc or JSDoc comments

Usage: Create a new instance of JavadocToMarkdown and then call either fromJavadoc(), fromPHPDoc() or fromJSDoc()

* **Constructor**

### function fromDoc(code, headingsLevel, fnAddTagsMarkdown)

Generates Markdown documentation from code on a more abstract level

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
  + fnAddTagsMarkdown — function — the function that processes doc tags and generates the Markdown documentation
* **Returns:** string — the Markdown documentation

### this.fromStaticTypesDoc = function(code, headingsLevel)

Generates Markdown documentation from a statically typed language’s doc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
* **Returns:** string — the Markdown documentation

### this.fromDynamicTypesDoc = function(code, headingsLevel, fnFormatType, fnFormatTypeAndName)

Generates Markdown documentation from a dynamically typed language’s doc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
  + fnFormatType — function — the function that formats a type information (single argument)
  + fnFormatTypeAndName — function — the function that formats type and name information (two arguments)
* **Returns:** string — the Markdown documentation

### this.fromJavadoc = function(code, headingsLevel)

Generates Markdown documentation from Javadoc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
* **Returns:** string — the Markdown documentation

### this.fromPHPDoc = function(code, headingsLevel)

Generates Markdown documentation from PHPDoc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
* **Returns:** string — the Markdown documentation

### this.fromJSDoc = function(code, headingsLevel)

Generates Markdown documentation from JSDoc comments

* **Parameters:**
  + code — string — the code that contains doc comments
  + headingsLevel — number — the headings level to use as the base (1-6)
* **Returns:** string — the Markdown documentation

### function fromSection(section, headingsLevel, fnAddTagsMarkdown)

Generates Markdown documentation from a given section

The function processes units of documentation, a line of code with accompanying doc comment

* **Parameters:**
  + section — object — the section that consists of code line and doc comment
  + headingsLevel — number — the headings level to use as the base (1-6)
  + fnAddTagsMarkdown — function — the function that processes doc tags and generates the Markdown documentation
* **Returns:** string — the Markdown documentation

# 

# conferenceManager.js Documentation

### function startConferenceManager()

initializes the conference and event forms and hides them, then loads the conference chooser, and the edit, view and delete buttons

### function initializeConferenceForm()

function initializeConferenceForm. Dynamically generate the html for the conference form, used for editing and creating conferences This html could be hard

# 

# loginAJAX.js Documentation

### var OUTPUT\_REGION\_ID = "outputRegion"

This is a global variable that defines the id of the region that all output is directed to.

### function submitForm(event)

This is called on success of a form submition. An ajax request is made based on the contents of the form that was submitted. All data is gathered using form.serializeArray, which collects all values from the form, and converts it to an array to be passed to the server. On success outputResult is called, and on failure, outputError is called. The error callback is for testing only.

* **Parameters:** event — Event — - a javascript submit event object

### function outputError(error)

Used for testing, so that any errors that are thrown back from the server will be seen. Populates the output region with the error messages.

* **Parameters:** error — Error — - The javascript ajax Error object.

### function outputResult(data)

The success call for the ajax request on form submition. see submitForm for more information about the form submition process. examines the given data, and responds accordingly. This expects an associative array that contains one key value pair. Expects the key of the array to be one of the following: error - an error was thrown, and the value contains a string error message. errorRedirection - An error was thrown, and the value is a URL to be redirected to. successMessage - The operation was performed successfully and a success message is printed to the output region. success - The operation was performed successfully, and the usual action is performed, where the value is a URL that is redirected to. else a generic message is printed to the output region and no action is performed.

* **Parameters:** data — string[] — - The array returned from the ajax call that should contain a single key value pair, a status, and a value.

# 

# mainSchedule.js Documentation

# 

# menu.js Documentation

### function removeSideBar(barId, iconId)

* **Parameters:**
  + barId — \* —
  + iconId — \* —

### function openSidebar(sidebarType, headingId)

* **Parameters:**
  + sidebarType — \* —
  + headingId — \* —

### function closeLeftSideBar()

### function closeCenterSideBar()

### function closeRightSideBar()

### function hideContentPage()

### function showContentPage()

### function isMobile()

* **Returns:** u —

### function toggleBodySidebar()

### function getPageWidth()

* **Returns:** u —

### function isMobileScreenWidth()

* **Returns:** u —

### function closeMenus()

### function changeSize(element, style, size)

* **Parameters:**
  + element — \* —
  + style — \* —
  + size — \* —

### function setCurrentFontDisplay()

### function toggleGraystyle()

### function toggleInvertColor()

### function turnOnGrayStyle()

### function turnOnColorDefault()

### function turnOnInverseStyle()

### function removeCurrentColorSetting()

### function toggleAriaButtonPress(elementId)

* **Parameters:** elementId — \* —

### function changeFontScreen()

### function setCookie(cname, cvalue)

* **Parameters:**
  + cname — \* —
  + cvalue — \* —

### function getCookie(cname)

* **Parameters:** cname — \* —
* **Returns:** u —

### function onloadCook()

### function onFontChange()

When the one of the following ids [‘increase-font’, ‘decrease-font’, ‘increase-font’] are clicked, the respective clickEvent will change the global variable that contains the current size setting. onFontChange will adjust the new currentSizing by calling the respective methods that adjusts sizes.

# 

# userAccountRegistration.js Documentation

# 

# userSchedule.js Documentation

# 

# userSettings.js Documentation

# 

# util.js Documentation

### function parseDate(dateString)

* **Parameters:** dateString — String — - a date in the format “YYYY-MM-DD.”
* **Returns:** string —

### function parseTime(timeString)

* **Parameters:** timeString — String — - a time string in the format “HH:MM:ss.”
* **Returns:** string —

### function addZero(i)

* **Parameters:** i — int — - an integer expected to be within 0 - 12.
* **Returns:** String — - returns a string with either just i or i with a ‘0’ in front of it, such as “01” or “09.”

### function onShowHiddenElement(elementId)

* **Parameters:** elementId — String — - the id of the element to be toggled.

### function onShowHiddenElementWithAria(elementId, ariaMsg)

* **Parameters:**
  + elementId — String — - the id of the element to be toggled.
  + ariaMsg — String — - the message to have the screen reader play after the element has been toggled.

### function onShowHiddenRow(elementId)

* **Parameters:** elementId — String — - the id of the row to be toggled.

### function onShowHiddenRowWithAria(elementId, ariaMsg)

* **Parameters:**
  + elementId — String — - the id of the row to be toggled.
  + ariaMsg — String — - the message to have the screen reader play after the row has been toggled.

### function notifyScreenreader(message)

* **Parameters:** message — String — - the message to be read by the screen reader.