**ICON Calendar Selection Page**

Documentation

CRL ICON OPAC page

*serve newspaper data*

ICON database

calendar\_selection\_

functions.js

*(more detail on p.3)*

*pass*

*LCCN*

*parameter*

*includes*

calendar.php

*(more detail on p.2)*

*handle user selections*

server\_calendar\_selection\_controls.html

*(more detail on p.4*

*other*

*includes*

|  |  |  |  |
| --- | --- | --- | --- |
| **File name** | **File type** | **Author** | **Description** |
| calendarFunctions.php | PHP | AE | Provides debugging functionality during development |
| styles.css | CSS | AE | Contains CRL styles for OPAC pages |
| selection\_page\_stylesheet\_ie.css | CSS | CJ | Contains calendar selection page styles for Internet Explorer |
| selection\_page\_stylesheet\_ff.css | CSS | CJ | Contains calendar selection page styles for Firefox |
| crlFunctions.js | JavaScript | AE | Provides custom CRL functionality for OPAC pages |
| json2.js | JavaScript | External | Provides a data-handling hack for Internet Explorer |
| jquery-1.7.1.min.js | JavaScript | External | Provides enhanced JavaScript functionality |
| jquery-ui-1.8.9.custom.min.js | JavaScript | External | Provides functionality for calendar selection widgets |
| initialize\_variables.js | JavaScript | CJ | Initialize global variables needed in JavaScript code |
| toplogo\_s36.html | HTML | AE | CRL header with logo and page title |
| botlogo.html | HTML | AE | CRL footer |

# File Name: *calendar.php*

**Author:** AE

**Purpose:** Main script container

**Pseudocode**

* Receives LCCN parameter from OPAC and saves it as PHP variable $pub\_id.
* Saves $pub\_id value in JavaScript variable selectedPubId.
* Includes JavaScript calendar selection code.
* Includes calendar selection page CSS.
* Includes calendar selection page HTML.

# File Name: *calendar\_selection\_functions.js*

**Author:** CJ

**Purpose:** JavaScript calendar selection processing

**Pseudocode**

* Display an animated gif in calendar control area while page loads.
* Define jQuery ready() handler. It will execute on page load/reload, after all elements (other than calendar controls) are available.
* Display pub id in navy blue bar.
* Retrieve issue counts by year from database and call function to populate ‘all issues’ object with these values.
* Populate available months object with abbreviations of month names where there are available issues in first year.
* Define handler for year dropdown change event. Whenever user selects a different year from dropdown, this handler will:
* Delete existing content in issue date display section of navy blue bar.
* Delete existing content in issue information area element and hide that element.
* Display animated loading gif in issue information area element while date controls are being repopulated.
* Call function that styles all month selection buttons to look unavailable while data controls are being repopulated.
* Set flag to indicate that a different year has been selected.
* Capture selected year value from dropdown and assign it to global 'selected year' variable.
* Call function to display blank calendar table during data retrieval.
* Call function to populate available months object for selected year and style available month buttons.
* Call function to style all month selection buttons to look unavailable.
* Define handler for month selection button 'click' event. Whenever user clicks on a month selection button, this handler will:
* Delete existing content in issue date display section of navy blue bar.
* Delete existing content in issue information area element and hide that element while date controls are being repopulated.
* Display animated loading gif in issue information area element while date controls are being repopulated.
* Remove 'selected' class attribute from all available month buttons by looping through available months object. For each element:
* Assign available month name abbreviation to a local 'temporary month box id' variable.
* Find month button element with id that matches 'temporary month box id' and assign it to a local 'temporary month box' variable.
* Assign only available – not selected – class to this element.
* Add 'selected' class to clicked month button if there are available issues for this month:
* Assign the class name property of clicked month button to a local 'temp class' variable.
* Test whether 'available' class name appears in this string:
* If it does, this month button is selectable, so add 'selected' class to it.
* Store id property of clicked month button element (an abbreviated month name) to global 'selected month' variable. **HERE**
* Call function to retrieve month index for 'selected month' from month name array and store this number in 'selected month' variable.
* Call function to display blank calendar table during data retrieval.
* Retrieve all available issue dates for selected year and month. Call function to populate an array with these available issue dates.
* Define handler for year bar 'click' event – whenever user clicks on a year bar, this handler will:
* Delete existing content in issue date display section of navy blue bar.
* Delete existing content in issue information area element and hide that element.
* Display animated loading gif in issue information area element during data retrieval.
* Disable year dropdown. It will be re-enabled after month buttons are restyled in response to year bar click.
* Call function to style all month selection buttons to look unavailable.
* Set flag to indicate that year has been changed.
* Capture selected year value from bar element’s id property and assign it to global 'selected year' variable.
* Call function to display blank calendar table during data retrieval.
* Call function to retrieve available months for selected year and style available month buttons.
* Define handler for year bar 'mouseover' event – whenever mouse cursor passes over a bar, this handler will:
* Change background color of bar to light green.
* Define handler for year bar 'mouseout' event – whenever mouse cursor leaves a bar, this handler will:
* Change background color of bar back to medium blue.
* Define function to populate array with issue counts by year.
* If there was a database error:
* Stop execution and display error message.
* Otherwise data retrieval code executed normally, so test number of issue count records retrieved:
* If any issue count records were retrieved, populate array with issue counts by year.
* Call function to complete page initialization.
* Define function to complete page initialization once issue counts by year are known.
* If no issue counts were retrieved:
* Hide loading gif.
* Display 'No issues in database' message.
* Otherwise issue counts were available, so:
* Set startup flag to true. This value will be used to determine selected issue date when page loads or reloads.
* Assign year dropdown element to global ꞌyear boxꞌ variable.ꞌ
* Set global ꞌselected yearꞌ variable equal to first year in data response.
* Calculate maximum number of issues available in a single year. Will be used to proportionally size year bars.
* Call function to display year bar chart.
* Call function to retrieve available months from database and style available month buttons
* Define function to display year bar chart:
* Initialize variables to store issue count data and construct chart elements.
* Loop through all elements of issue count array. For each array element:
* Store year in local variable.
* Store issue count in local variable.
* Calculate bar size as a percentage of maximum issue count and store in a local variable.
* Store final digit of year in a local variable.
* Construct a class name using final year digit, to determine bar placement. A bar element with class ‘y0’ will appear in the leftmost position of its parent decade element. A class 'y9' bar will appear in the rightmost position, etc.
* Add '0' to first three digits of year and store result in a local variable representing first year in that decade.
* Find anchor element on page with text containing 'first year in decade' value.
* Find next unordered list element.
* Construct a new list item element.
* Append list item to unordered list.
* Assign 'y*n*' class name to list item.
* Create a new anchor element.
* Append anchor to list item.
* Create a new span element.
* Assign 'count' class name to span. This will cause bar to change color under mouse movement.
* Add tooltip text to span consisting of year and issue count.
* Set span id equal to year.
* Set span size equal to calculated percentage.
* Append span to anchor created above. Span id allows capture of selected year in response to mouse click on a bar.
* Define function to populate available months object for selected year and style available month buttons:
* If there was a database error:
* Stop execution and display error message.
* Otherwise, data retrieval code executed normally and provided available month names and month numbers for selected year.
* Initialize variables to store available month name and month number data.
* Call function to reset available month buttons by styling them all to look unavailable.
* Loop through all available month names. For each available month name:
* Convert first three characters of name to lower case and store in global array of available month abbreviations.
* Loop through array of available month abbreviations. For each available month abbreviation:
* Store value in a local 'month box id' variable.
* Find month button with matching id and add 'available' class name. Available month buttons will be styled to have a light green background.
* Test flags to see whether user has just loaded page or changed year. If so, user has not yet had a chance to select a month. Therefore application should provide a default selected month.
* Store first available month number in global 'selected month' variable.
* If user has just loaded page, selected month will be first month with available issues in first available year.
* If user has changed year, selected month will be first month with available issues in selected year.
* Store first available month name abbreviation in a local 'month box id' variable.
* Find month button with matching id and add 'selected' class name. Selected month button will be styled to have a medium blue background.
* Now selected year and month are known, so retrieve all available issue dates for that year and month. Call function to populate an array with these available issue dates.
* Call function to populate year dropdown with all available years for selected title.
* Set selected year in dropdown:
* If user has just loaded page, selected dropdown year will be first year with available issues for selected title.
* If user has changed year, selected dropdown year will be user selection.
* Enable year dropdown control. It may have been disabled during data retrieval when a year bar was clicked.
* Define function to populate array with available issue dates (if any) for selected year and month
* If there was a database error:
* Stop execution and display error message.
* Otherwise, data retrieval code executed normally.
* Clear out existing array of available issue dates, if one exists.
* Test whether response provided a 'no issues published' message, or available issue dates for selected month and year.
* If response provided a 'no issues published' message, user selected a month with no available issues. Set calendar default date equal to first day of selected month.
* Construct a date object corresponding to first day of selected year and month.
* Store date object in global 'first day' variable. It will be used to set default calendar date.
* Increment value stored in global 'selected month' variable by 1, to compensate for month index incompatibility between JavaScript and PHP.
* Pass value stored in global 'selected month' variable to function that will pad it with a leading zero if necessary.
* Construct a string in ‘YYYY-MM-DD’ format from selected year, padded selected month, and '01'.
* Store date string in global 'selected date' variable. It will be used to display date in navy blue bar.
* Pass 'first day' date object to calendar control’s ꞌset dateꞌ method. This will trigger calendar ꞌonChangeMonthYearꞌ and ꞌbeforeShowDayꞌ events, allowing date cells to be restyled for selected month.
* Display ꞌNo issues available this monthꞌ message in navy blue bar.
* Hide animated issue info loading gif.
* Otherwise, response provided available issue dates for selected month and year. *[4/11-12 Note - issue records with a dummy issue\_date value of "0000-00-00" are excluded from results. Per Andy, these records will gradually be assigned real issue\_date values.]*
* Store issue dates in global 'available day' array.
* Store first issue date in global ꞌselected dateꞌ variable.
* Extract last 2 characters (the day part) from ꞌselected dateꞌ and store it in local ꞌtemp default dateꞌ variable.
* Parse ꞌtemp default dateꞌ variable into an integer value. [To avoid known JavaScript bug when trying to convert string value ꞌ8ꞌ to an integer, set radix parameter equal to ꞌ10ꞌ in parseInt() function. Otherwise JavaScript will convert string to an octal rather than a decimal integer.]
* Store day integer value in global ꞌselected dayꞌ variable.
* Construct a date object using ꞌselected yearꞌ, ꞌselected monthꞌ, and ꞌselected dayꞌ variables.
* Store date object in global 'default date' variable. It will be used to set default calendar date.
* Test flag to determine whether page load has occurred:
* If page was just loaded or reloaded, calendar control should be created or reset with its default date equal to first available day in first available month and year.
* Initialize calendar control options as follows:
* Disable built-in calendar year and month controls because user will select year from bar chart or dropdown, and month from button display.
* Set calendar default date equal to default date object constructed above.
* Set calendar date format equal to 'yyyy-mm-dd', because that is the format in which dates are retrieved from the database.
* Set calendar ꞌbeforeShowDayꞌ option equal to function that will style calendar date cells. Function will be called automatically whenever calendar needs to be redisplayed – i.e., on page load or when year or month is changed.
* Set calendar ꞌonChangeMonthYearꞌ option equal to function that will test length of available day array. If zero, ꞌNo issues available for this monthꞌ will be display in navy blue bar. Function will be called automatically on page load and whenever month or year is changed.
* Set calendar ꞌonSelectꞌ option equal to a function that receives a date object as a parameter whenever a calendar day cell link is clicked. Function:
* Extracts day portion of date parameter and stores it in global ꞌselected dayꞌ variable.
* Increments selected month variable by 1, to compensate for month index incompatibility between JavaScript and PHP.
* Pass ꞌselected monthꞌ variable to function that will pad it with a leading zero if necessary.
* Using selected year, month, and day values, constructs a date string to display in navy blue bar.
* Calls function to display top-level filter options for issue records.
* Reset startup and ꞌyear changeꞌ flags to false.
* Hide animated calendar loading gif.
* Display ꞌselected dateꞌ in navy blue bar.
* Call function to display top-level filter options for issue records.
* Unhide issue display area.
* If page load has not occurred, then this function was called because year or month was changed. Redisplay calendar control with its default date equal to first available day in selected month and year.
* Pass default date object to calendar control’s ꞌset dateꞌ method. This will trigger calendar ꞌonChangeMonthYearꞌ and ꞌbeforeShowDayꞌ events, allowing date cells to be restyled for selected month.
* Call function to display top-level filter options for issue records.
* Define function to populate year dropdown with all available years for selected title:
* Clear existing options from year dropdown.
* Loop through ꞌissue countꞌ array. For each element:
* Pass array year value to function that will add corresponding option to year dropdown.
* Define function to style calendar date cells.
* Date cell styles:
* Selected (publication date) cell will appear as a clickable link on a medium blue background.
* Available (publication-date) cells will appear as clickable links on a light green background.
* Non-available (non-publication-date) cells are not clickable and will appear on a white background.
* This function is called automatically by jQuery calendar widget and passed a date object parameter for each day in selected month.
* Test length of ꞌavailable daysꞌ array.
* If array length is greater than zero, this date cell might need to be styled as selected or available.
* Increment the month value of date parameter by 1, to compensate for month index incompatibility between JavaScript and PHP.
* Pass this month value to function that will pad it with a leading zero if necessary.
* Assign month value to a local ꞌtemp monthꞌ variable.
* Pass day value of date parameter to function that will pad it with a leading zero if necessary.
* Assign day value to a local ꞌtemp dayꞌ variable.
* Construct a string in ꞌYYYY-MM-DDꞌ format from year value of date parameter and temp month and temp day variables.
* Assign date string to local ꞌtemp date stringꞌ variable.
* Test whether ꞌtemp dateꞌ string appears in ꞌavailable dayꞌ array.
* If ꞌtemp dateꞌ string does appear in ꞌavailable dayꞌ array:
* Convert string value stored in ꞌtemp dayꞌ variable to an integer.
* Test whether ꞌtemp dayꞌ integer equals values stored in ꞌselected dayꞌ variable.
* If ꞌtemp dayꞌ does equal ꞌselected dayꞌ:
* Assign ꞌselected publication dateꞌ class to this date cell
* If ꞌtemp dayꞌ does not equal ꞌelected dayꞌ:
* Assign ꞌpublication dateꞌ class to this date cell (indicating that it is available but not selected)
* If ꞌtemp dateꞌ string does not appear in ꞌavailable dayꞌ array:
* Assign ꞌnot publication dateꞌ class to this date cell
* If array length is zero, assign ꞌnot publication dateꞌ class to this date cell.
* Define function to display top-level filter options for available issue records for selected day.
* Test length of ꞌavailable daysꞌ array.
* If array length is zero, no filter options should be displayed, so:
* Hide animated issue-info loading gif.
* Hide issue information area element.
* If array length is greater than zero, create and display filter links:
* Delete existing content in issue information area element.
* Append a new div element to issue information area element and assign it to local ꞌtemp Containerꞌ variable.
* Set ꞌtemp containerꞌ id attribute = tempContainer.
* Assign ꞌlink group containerꞌ class to ꞌtemp containerꞌ element for styling purposes.
* Append ꞌFilter byꞌ text and ꞌFilter by Repositoryꞌ and ꞌFilter by Formatꞌ links to ꞌtemp containerꞌ element.
* Add padding to ꞌFilter byꞌ text.
* Set onclick attribute of ꞌFilter by Repositoryꞌ link equal to a function that will retrieve all repository values for selected date and add repository links to issue information area element.
* Set onclick attribute of ꞌFilter by Formatꞌ link equal to a function that will retrieve all vormat values for selected date and add format links to issue information area element.
* Hide animated issue-info loading gif.
* Display issue information area element, now populated with top-level filter options.
* Define function to display repository links for selected day.
* If there was a database error:
* Stop execution and display error message.
* Otherwise, data retrieval code executed normally.
* Assign retrieved repository data to global ꞌrepository arrayꞌ variable.
* Delete existing content in issue information area element.
* Append a ꞌBack to filter list link elementꞌ to issue information area. In response to a click event on this element, top-level filter options will be displayed.
* Append an informational ꞌPlease select repositoryꞌ div element to issue information area.
* Set id of this div equal to ꞌtemp containerꞌ.
* Loop through ꞌrepository arrayꞌ. For each element:
* Assign repository id value to a local ꞌtemp repository idꞌ variable.
* Construct a div element enclosing an internal anchor link element.
* Set link id equal to ꞌtemp repository idꞌ.
* Set link text content equal to ꞌtemp repository idꞌ.
* Append div element to ꞌtemp containerꞌ element.
* Assign collection of internal anchor links created above to a local ꞌrepository linksꞌ array variable.
* Loop through ꞌrepository linksꞌ array variable. For each link:
* Set its onclick event equal to a function that will retrieve and displays issue records for the selected repository.
* Define function to display format links for selected day.
* If there was a database error:
* Stop execution and display error message.
* Otherwise, data retrieval code executed normally.
* Assign retrieved format data to global ꞌformat arrayꞌ variable.
* Delete existing content in issue information area element.
* Append a ꞌBack to filter list link elementꞌ to issue information area. In response to a click event on this element, top-level filter options will be displayed.
* Append an informational ꞌPlease select formatꞌ div element to issue information area.
* Set id of this div equal to ꞌtemp containerꞌ.
* Loop through ꞌformat arrayꞌ. For each element:
* Assign format value to a local ꞌtemp format idꞌ variable.
* Construct a div element enclosing an internal anchor link element.
* Set link id equal to ꞌtemp format idꞌ.
* Set link text content equal to ꞌtemp format idꞌ.
* Append div element to ꞌtemp containerꞌ element.
* Assign collection of internal anchor links created above to a local ꞌformat linksꞌ array variable.
* Loop through ꞌformat linksꞌ array variable. For each link:
* Set its onclick event equal to a function that will retrieve and displays issue records for the selected format.
* Define function to display issue records for the selected repository.
* If there was a database error:
* Stop execution and display error message.
* Otherwise, data retrieval code executed normally.
* Delete existing content in issue information area element.
* Append a ꞌBack to repository listꞌ link element to issue information area. In response to a click event on this element, repository links will be redisplayed.
* Assign retrieved issue data to global ꞌissue info arrayꞌ variable.
* Call function to create issue information display string.
* Append display string to issue information area element.
* Define function to display issue records for the selected format.
* If there was a database error:
* Stop execution and display error message.
* Otherwise, data retrieval code executed normally.
* Delete existing content in issue information area element.
* Append a ꞌBack to format listꞌ link element to issue information area. In response to a click event on this element, format links will be redisplayed.
* Assign retrieved issue data to global ꞌissue info arrayꞌ variable.
* Call function to create issue information display string.
* Append display string to issue information area element.
* Define function to create issue information display string.
* Empty existing string.
* Loop through issue info array. For each issue record:
* Populate local variables with the following field values:
* Issue id
* Pub id
* Pub date
* Repository id
* Physical condition
* Format
* Provenance
* Notes
* If the following fields were null or contained dummy values, store ꞌUnknownꞌ in the local variables:
* Physical condition
* Provenance
* Append labels and values to display string for the following fields:
* Pub date
* Repository id
* Physical condition
* Format
* Test value of note field:
* If null or dummy value, don’t append to display string
* Else if populated with a real value, append label and value to display string
* Append label and value to display string for provenance field
* Prepend line break to issue information display string
* Define function to display blank calendar simulating disabled state during repopulation of available day array.
* Set color property of the following elements equal to white:
* Unselectable (non-publication date) calendar cells.
* Unselectable (non-publication date) calendar cell links.
* Available (publication date) calendar cells.
* Selected publication date calendar cell.
* Selected publication date calendar cell link.
* Set background-color and color properties of the following elements equal to white:
* Day of week calendar table heading cells.
* Month/year calendar element.
* Define function to reset available month buttons by styling them all to look unavailable.
* Assign ꞌnot availableꞌ class o all month button elements.
* Define function to retrieve month index for selected month name abbreviation. Function:
* Receives two parameters:
* Array of available month name abbreviations
* Selected month name abbreviation
* Returns index position of selected month name abbreviation in array.
* Define function to add option to year dropdown.
* Function receives three parameters:
* Year dropdown element
* Year text in ꞌYYYYꞌ format
* Counter integer corresponding to position of year in issue count array
* Create an option element.
* Set option text equal to year text parameter value.
* Set option value equal to counter.
* Add option element to year dropdown element.
* Define function to pad a number with leading zero if less than 10.
* Function receives a numeric parameter.
* Test parameter value:
* If parameter value is less than 10, return string containing ꞌ0ꞌ.
* Else if parameter value is greater than or equal to 10, return empty string.
* Define function to test whether a particular class has been assigned to a page element. Function:
* Receives two parameters: an element and a class name string.
* Constructs a regular expression to test whether class name parameter appears anywhere within element’s class name property.
* Returns result of regex match test.
* Define function to add a class to a page element. Function:
* Receives two parameters: an element and a class name string.
* Calls function to test whether class name parameter has been applied to element parameter.
* If result is false, adds class name parameter to element parameter’s class name property.

# File Name: *server\_calendar\_selection\_controls.html*

**Author:** CJ

**Purpose:** Defines calendar selection control elements

**Pseudocode**

* xx:
* xx