# JavaScript and Ajax

## EXERCISE 3: PHOTO ALBUM CLIENT-RENDERED VERSION C

### Photo Album Client-Rendered Version C

The next version of the photoalbum adds the ability to delete photographs from the album. The first thing this requires is a link displayed next to each photograph to allow it to be deleted. This link needs to be added to the **createPhotoHTML** function. Like the PHP photoalbum, this link needs to embed the id number of the photograph. Unlike the PHP version, it does not need to have the href attribute pointing back to the PHP script, in fact, it does not even need to be a link using the <a> tag at all! This is because when we click on delete it is going to run a JavaScript function that will do everything else.

Make a copy of ***photoalbum-client-rendered-version-b.html*** and call it ***photoalbum-client-rendered-version-c.html.*** Then add the highlighted lines to the existing **createPhotoHTML** function. Note that this function is given a JavaScript object called ***photoInfo*** which has named fields for the *image*, *name*, *description* and *photoId* values.

|  |
| --- |
| function createPhotoHTML( **photoInfo**) {  return " <div class='w3-container w3-row w3-col l4'>" +  " <div class='w3-container w3-card-8 w3-margin-8'>" +  " <div class='w3-container w3-half w3-image'>" +  " <a class='example-image-link' data-lightbox='example-1' href='images/" + **photoInfo.image** + "'><img src='images/" + **photoInfo.image** + "' class='example-image w3-circle'></a>" +  " <div class='w3-title w3-text-purple'>" + **photoInfo.name** + "</div>" +  " </div>" +  " <div class='w3-container w3-half'>" + **photoInfo.description** +  // next line should be conditional on whether logged in or not  " <span class='link deletelink' id='" + **photoInfo.photoid** + "'>(Delete)</span>" +  " </div>" +  " </div>" +  " </div>";  } |

The link that will be added beside each picture will look something like this (for the photo with id 12):

<span class='link deletelink' id='12'>(Delete)</span>"

Note that this is just a <span> element, not an <a> element. This does not matter, because we can attach a JavaScript ***onClick*** event handler to any element in the DOM, so a span works just as well.

To add an onClick handler to an element we need to get a reference to the element and assign a function to its onclick property. Fortunately, JQuery provides us an easy way to do this. Every one of the delete link spans will have the CSS class **deletelink**. JQuery lets us select DOM elements using CSS selector rules, so **$(“.deletelink”)** will return a collection of all DOM elements with that class, and in our case that means the delete <span>s for every photograph on the page. JQuery also allows us to add a single handler function to ALL the elements it finds in one line. Try the following example in a new file:

|  |
| --- |
| <html>  <head>  <script src='http://code.jquery.com/jquery-1.11.3.min.js'></script>  <script>  **$(document).ready(**  **function() {**  **$("h1").click( function( event) {**  **alert( "You clicked: " + event.target.innerHTML);**  **})**  **}**  **);**  </script>  </head>  <body>  <h1>Click Me</h1>  <h1>Click me too!</h1>  </body>  </html> |

**$("h1")** selects all the h1 elements in the document.

**$("h1").click( *FUNCTION* )** adds the given *FUNCTION* as a click handler to each of the h1 elements in the document.

In this case the function is :

**function( event) {**

**alert( "You clicked: " + event.target.innerHTML);**

**}**

Note that the function takes a parameter called **event**. This is an object provided by the JavaScript engine describing a user interface event. In this case it will be a mouse click, but this object also has a field called **target** which is a reference to the DOM element that was clicked on. This function uses the built in function ***alert*** to display a dialog box which displays the content of the target of the event – that is, the text in the h1 tag you clicked on.

So from this it follows that **$(“.deletelink”).click( *FUNCTION*)** could be used to add click handlers to all the photographs on the page, as they all have the class **deletelink** assigned to them.

It is important to realise that this can only be done once they exist as elements in the DOM, and the HTML for the photographs is dynamically added by JavaScript in the **loadPhotos()** function. Therefore, we must add the click handlers after the photos have been added to the DOM, at the end of the code in the loadPhotos() function.

Modify loadPhotos() as shown below to add a click handler to all delete links. This adds a function similar to the example above, which simply displays a dialog box and tells you the id number of the photograph we want to delete.

|  |
| --- |
| function loadPhotos() {  $.getJSON( "json.txt",  function( photos) {  $('#album').empty();  for (i = 0; i < photos.length; i += 3) {  var rowDiv = $( "<div class='w3-row'>");  $('#album').append( rowDiv);  for( j = i; j < i + 3; j++ ) {  if ( j < photos.length) {  rowDiv.append( createPhotoHTML( photos[j]));  }  }  }  **// add listener to all deletelinks**  **$('.deletelink').click(**  **function( event) {**  **alert(**'**Photo id:** ' **+ event.target.id);**  **}**  **);**  }  );  } |

We now have links which run a function when clicked on, and that function knows the id number of the photo in question. What we need now is a function that will actually delete the photo. Remember that the photographs are actually stored on the server, so we will have to ask the server to do the deletion. JQuery helps here by allowing us to send a POST request to the server easily, and handle its response to make sure it succeeded. Here is the JavaScript we need. Add it to your file.

|  |
| --- |
| function deletePhoto( event) {  $.post( "json-photoalbum-delete.php",  "deletionid=" + event.target.id,  function( data) {  if ( data.result != "success") {  alert( "ERROR: " + data.error);  } else {  loadPhotos();  }  });  } |

This function needs to be called with the mouse click event object generated by clicking on one of the delete links. It then uses the JQuery **$.post** function to create an HTTP request using the POST method. This request is sent to the URL given as the first parameter to **$.post**, in this *case json-photoalbum-delete.php*. The second parameter is the name value pairs to send in the body of the message, in this case *deletionid* with the value taken from the delete link that was clicked on. The third parameter is a function to be executed when the response from the server is received. This will be given an object constructed from the body of the response message. The PHP script this is calling will return a JSON encoded object which JQuery will automatically turn into a corresponding JavaScript object. It will contain a field called **result** with the value *success* or *failure*, depending on whether the server was able to delete the requested photograph or not. The function given will display an error in an alert if one happened, but otherwise it will just call the **loadPhotos()** function again. This will discard the existing photographs and reload them from the server so that the deleted picture will no longer appear.

The next thing you need to do is change the code that adds a click handler to the delete links so that instead of doing **alert( ‘Photo id: ‘ + event.target.id);** it does **deletePhoto( event);** instead.

The JavaScript is now complete, but we need a PHP script on the server for it to invoke. Save the following as ***json-photoalbum-delete.php***.

|  |
| --- |
| <?php  header("content-type: application/json");  require\_once("lib/dbutils.php");  require\_once("photoalbum-common.php");  $pdo = connect();  if ( isset( $\_POST['deletionid'])) {  $errorMessage = deletePhotograph( $pdo, $\_POST['deletionid']);  } else {  $errorMessage = "No id provided in post";  }  if ( $errorMessage == "") {  print "{\"result\": \"success\"}";  } else {  print "{";  print "\"result\": \"failure\"";  print ",";  print "\"error\": \"$errorMessage\"";  print "}";  }    ?> |

Note that this script begins by emitting a content-type header declaring that it will return JSON data as its result. If you view it directly in a web browser you will see the JSON data encoding an error message.

Next note that it imports the file photoalbum-common.php. This file contains the deletePhotograph function as used by the PHP photoalbum. We use the same function here.

The script then checks to see if the request contains a deletionid value. If it does, it calls the deletePhotograph function, giving it the database connection and the id number to delete. This function returns an empty string on success, or an error message if something went wrong. If there is no deletionid in the request (as there would not be when you viewed the file directly in a web browser) the error message is set appropriately.

The last part of the script prints out the JSON data for the response message. This consists of the field result with the value success or failure, and in the case of failure, another field called error which contains the errormessage..

With this script on the server and all the JavaScript changes in place, your photoalbum should allow the deletion of photographs.

|  |
| --- |
| **EXERCISE 3: PHOTO ALBUM CLIENT-RENDERED VERSION C** |
| ***Required in your report (basic exercise).*** |
| * A clickable link to photoalbum-client-rendered-c.html. * Reflection on the tasks and all activities involved. |
| ***Extended Tasks*** |
| * No extended tasks are defined for this exercise. |
| *Original additional work:*  *Investigate and experiment with any related subject matter that interests you.* |