# JavaScript and Ajax

## EXERCISE 1: PHOTO ALBUM CLIENT-RENDERED VERSION A

### Photo Album Client-Rendered Version A

Create a new file called **photoalbum-client-rendered-version-a.html** and put the following HTML code in it, changing the text in red to be your name and email address. This file needs to be located in the same place as your existing photoalbum scripts so that it has access to the lightbox files and the **photoalbum.css** file.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>JSON Photo Album</title>  <meta name='viewport' content='width=device-width, initial-scale=1'>  <link rel='stylesheet' href='http://www.w3schools.com/lib/w3.css'>  <link rel='stylesheet' href='lib/lightbox2/css/lightbox.css'>  <link rel='stylesheet' href='photoalbum.css'>  <script src='http://code.jquery.com/jquery-1.11.3.min.js'></script>  </head>  <body>  <header class='w3-container w3-orange w3-center'>  <h1>My Photo Album</h1>  <h2>Your Name</h2>  <p>Here are some photographs and descriptions.</p>  </header>  <div id='message'>  </div>  <div id='album'>  </div>  <footer class='w3-container w3-orange w3-padding-jumbo'>  <div class='w3-container w3-center'>  <a href='mailto:youremail@lsbu.ac.uk' class='link'>Contact me</a>  </div>  </footer>  <script src='lib/lightbox2/js/lightbox.js'></script>  </body>  </html> |

Note the following:

* This is a plain page of HTML not PHP.
* Although it is declares the top and bottom of the photoalbum page, there is nothing in the middle. Instead there are two empty divs which have id attributes. We can use JavaScript running in the page to add content to those divs, and this is how this version of the photoalbum will work.
* The last line in the <head> of the page imports the JQuery JavaScript library, allowing us to make use of JQuery functions in this file.

Add the following code as the last thing inside the <head>…</head> tags:

|  |
| --- |
| <script>    $(document).ready(  function() {  $('#album').append( "<div style='border: dashed magenta 8px;'> <h1>Here is some content for the 'album' div</h1></div>");  }  );  </script> |

This code uses the JQuery library to do a few things. The JQuery library defines a special function denoted by a dollar sign. It can take a wide variety of different kinds of parameters and provides a wide range of useful functions. The first use in this code is ***$(document)***. In this case the parameter is the predefined JavaScript variable identifying the root DOM element for the currently displayed document. The JQuery object returned by the $ function has a function called ***ready()*** which is used to execute some JavaScript code when the page has finished loading. It is very commonly used to start some script running on a page automatically. It requires as a parameter a JavaScript function, and the code above gives in an anonymous function for this purpose. Anonymous functions have no name and are simply placeholders for code to be executed by the function they are given to.

**$(document)**.**ready(**

**function() {**

**// code to run in here**

**}**

**)**;

In the example above the code to run consists of the line show abbreviated below:

**$('#album')**.**append(** "<div ... /div>"**)**;

This again uses the JQuery $ function, but this time gives it a string containing an ID string. This selects the DOM element which has that ID (in this case one of the empty divs in the page) and wraps it in a JQuery object. Then the JQuery function ***append()*** is called to create new DOM elements and add them to the existing children of the selected element. In this case we give append a string containing HTML code which is automatically converted to DOM elements and added in. View the page in a web browser (you do not need to upload it to a server to do this) and then use the developer tools options to inspect the DOM and see that the div with the id album is no longer empty but contains a div and h1 element.

Of course what we want to do is insert a whole grid of photographs so we need a more complicated function. Replace all the code you have already with this version:

|  |
| --- |
| $(document).ready(  function() {  for (i = 0; i < 9; i += 3) {  var rowDiv = $( "<div class='w3-row'>");  $('#album').append( rowDiv);  for( j = i; j < i + 3; j++ ) {  if ( j < 9) {  rowDiv.append( "<div style='border: solid blue 2px;' class='w3-third'>Place for photo</div>");  }  }  }  }  ); |

This code does the following:

1. It iterates through two nested loops to counting up to nine in three groups of three..
2. For each group of three it prints out a div with the w3-row class and creates a new div element to represent a row and appends it to the “album“ div.
3. For each member of the group of three it appends a div to the row div, with a blue border and containing a string which says “Place for a photo”.

View this in a web browser and inspect the DOM to see the elements which have been added to album div.

The result is a 3x3 grid of divs with blue borders and the w3-third class. This is more like what we want, but in fact what we need to do is iterate over a collection of photographs and for each one add elements to the page to create the same HTML we have had in earlier versions, that is something like:

<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 href='images/b1.png'>

<img src='images/b1.png' class='example-image w3-circle'>

</a>

<div class='w3-title w3-text-purple'>John</div>

</div>

<div class='w3-container w3-half'>

<p>This is John Lennon. He was one of the Beatles.</p>

<p>He was one of the two main song-writers.</p>

</div>

</div>

</div>

To do this we need three things:

1. the photograph information stored somewhere, for example in an array
2. a function to pass to ***$(document).ready*** which iterates over that array and for each item in it calls
3. a function to create the required HTML elements for a given array element

Although we could declare an array directly in JavaScript, for reasons which will become clear in the next version, it’s better to declare it as a JSON string. JSON allows complex data structures to be easily described.

This is the JSON we want (with the descriptions shortened for clarity):

[

{

"image": "b1.png",

"name": "John",

"description": "<p>This is John Lennon.</p>"

},

{

"image": "b2.png",

"name": "Paul",

"description": "<p>This is Paul McCartney.</p>"

},

{

"image": "b3.png",

"name": "George",

"description": "<p>This is George Harrison.</p>"

},

{

"image": "b4.png",

"name": "Ringo",

"description": "<p>This is Ringo Starr.</p>"

}

]

The outermost square brackets denote an array. Inside that there are four elements, each enclosed in curly brackets and separated by commas, which define objects. Each of these objects have the same named fields (image, name and description) with specific values.

Here is the same data as JavaScript code joining together shorter strings with + signs and assigning the end result to a global variable called *json*. This code should be added to your <script>...</script> block as the first thing in it.

|  |
| --- |
| var json = '[' +  '{ "image": "b1.png", "name": "John", "description": "<p>This is John Lennon. He was one of the Beatles.</p><p>He was one of the two main song-writers.</p>" },' +  '{ "image": "b2.png", "name": "Paul", "description": "<p>This is Paul McCartney. He was also one of the Beatles.</p><p>He was the other main song-writers.</p>" },' +  '{ "image": "b3.png", "name": "George", "description": "<p>This is George Harrison. He was also one of the Beatles.</p>" },' +  '{ "image": "b4.png", "name": "Ringo", "description": "<p>This is Ringo Starr. He was also one of the Beatles.</p>" }' +  ']'; |

Now replace the ***$(document).ready()*** code block with this one:

|  |
| --- |
| $(document).ready(  function() {  var photos = $.parseJSON( json);  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( "<div style='border: solid blue 2px;' class='w3-third'>" + photos[j].name + "</div>");  }  }  }  }  ); |

This version is similar to the previous one but does the following:

1. It converts the JSON string stored in the *json* variable into an actual array of objects and assigns them to the variable *photos*. This is achieved using the ***$.parseJSON*** function provided by the JQuery library.
2. It iterates through each element in the *photos* array, using nested loops to handle them three at a time.
3. For each row of three it prints out a div with the w3-row class and creates a new div element to represent the row and appends it to the “album“ div.
4. For each object it appends a div to the row div, with a blue border and containing the value of the *name* field of the photo object.

View this in a web browser and inspect the DOM to see the elements which have been added to album div.

So what we have now for each photograph is something like

<div style='border: solid blue 2px;' class='w3-third'>John</div>

But remember what we want is something like this:

<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 href='images/b1.png'>

<img src='images/b1.png' class='example-image w3-circle'>

</a>

<div class='w3-title w3-text-purple'>John</div>

</div>

<div class='w3-container w3-half'>

<p>This is John Lennon. He was one of the Beatles.</p>

<p>He was one of the two main song-writers.</p>

</div>

</div>

</div>

The best way to deal with this is to create the elements for the photographs in a separate function so that the code does not become unreadable. Here is the function to do it:

|  |
| --- |
| 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** +  " </div>" +  " </div>" +  " </div>";  } |

Although the line wrapping makes it difficult to read, you should be able to see that this function assembles a string containing HTML matching what we want and returns it to the calling code. It takes an object called *photoInfo* as a parameter, and this will be one of the objects from the array. The values of the fields of this object are inserted into the HTML string at the appropriate places and this is shown highlighted.

Add this function to your script after the JSON declaration and before the ***$(document).ready*** code.

Now all that remains is to change the line in the $(document).ready code that appends an individual photo element from

rowDiv.append( "<div style='border: solid blue 2px;' class='w3-third'>" + photos[j].name + "</div>");

To a similar line which calls the function above with the relevant photo object:

rowDiv.append( createPhotoHTML( photos[j]));

Make these changes and you should have a photoalbum page which is rendered on the client machine, not the server. In functionality, it is almost exactly the same as the first PHP version you did, with the photographs hard coded into an array.

Upload the file to Daydream so it can be linked to from your report.

|  |
| --- |
| **EXERCISE 1: PHOTO ALBUM CLIENT-RENDERED VERSION A** |
| ***Required in your report (basic exercise).*** |
| * A clickable link to photoalbum-client-rendered-a.html. * Reflection on the tasks and all activities involved. |
| ***Extended Tasks*** |
| * Add another photograph to the JSON data. * Use JQuery hide and fadeIn functions to hide the photoalbum while loading the images and cause them all to fadeIn instead of just appearing. |
| *Original additional work:*  *Investigate and experiment with any related subject matter that interests you.* |