Interactivity with JavaScript and PHP

# Introduction

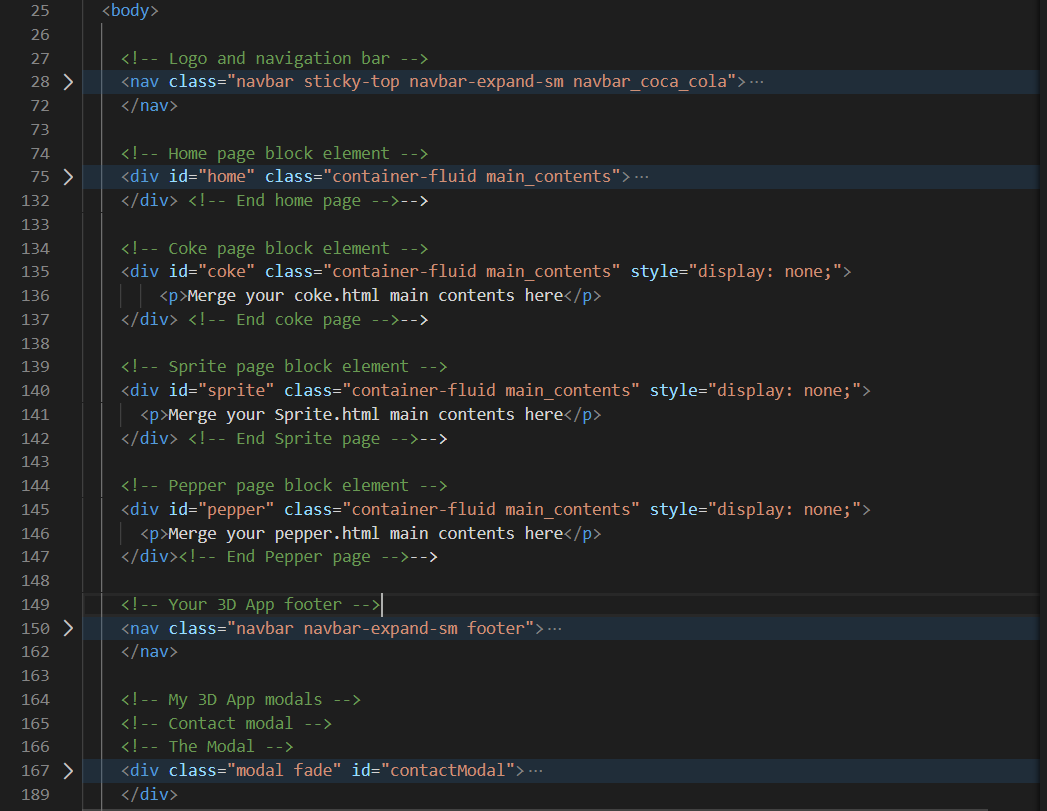


Figure 12: New HTML5 layout for the SPA to incorporate home, coke, sprite and pepper contents.

1. The home page contents block element already exists, just add the home id. Then created new block elements for the Coke, Sprite and Pepper contents. The structure should look something like that shown in Figure 12.
2. You will need to implement the JavaScript swap function (leave placeholder text for the contents as indicated in Figure 12) and test it so that you know that the JavaScript swap function is set up correctly as given in the example.
3. If you are using Bootstrap you will use a Bootstrap .row class for the main contents on each of the home, coke, sprite and pepper pages. You will also need any CSS and JavaScript libraries that were used for the X3D, etc.
   * Test one page at a time as you create the new contents
4. You can look at the **Live Feedback Site** to get an idea of what the result looks like, see Figure 13.

|  |  |
| --- | --- |
|  |  |
| Bootstrap 4.3 2019 version | Bootstrap 4.4 2020 version |

Figure 13: Home page, i.e. index.html

At this stage, if you have the structure right your Bootstrap should show an index.html as the home page — only one page now. Don’t forget to copy over the X3D libraries into the index.html page, or you won’t see your 3D!

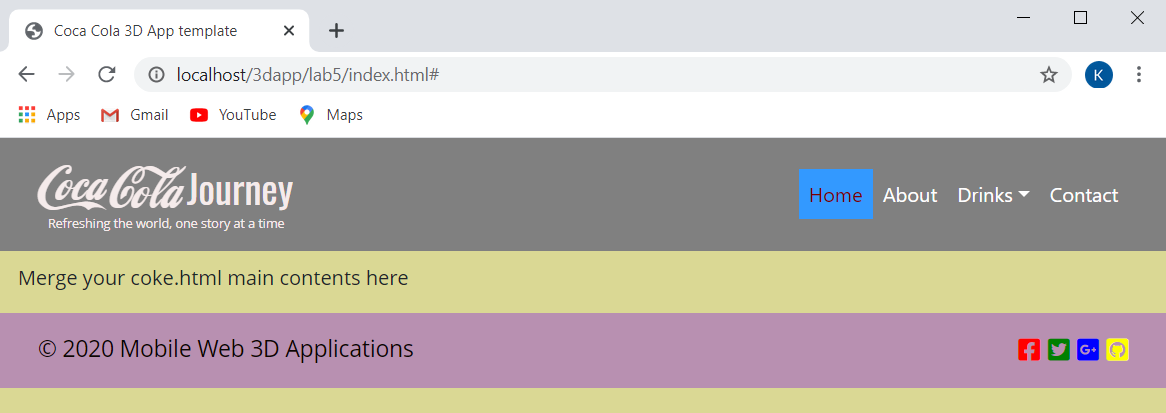


Figure 14: The sprite contents, note index.html. The coke and pepper should be the same

If you get stuck you may examine the **Live Feedback Site**. Note these URLs will show the final result, not partial results

Ok, so you should now have a single page application of your 3D App using Bootstrap components, make sure this works properly before you proceed to the next stage. Once it works, you can remove your coke.html, sprite.html and pepper.html files from this lab 5 folder.

You should now only have the index.html file and your Lab 5 folder should look like that shown in Figure 15. You can check the **Live Feedback Site** for more details. Note all extraneous CSS and JS files have been removed and all libraries installed rather than using CDNs. We have a single Sindex.html because we have just created a single page application (SPA) for our 3D App. Make sure you test it and it works ok.

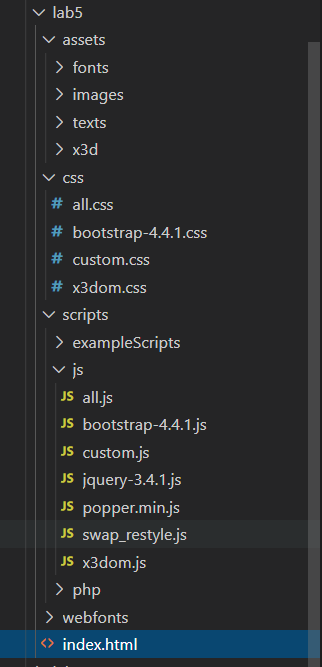


Figure 15: Your Lab 5 structure so far

Next, let’s start to add the image gallery and add some more functionality.

# Create an Image Gallery using JavaScript, JQuery and PHP

In this tutorial section, you will be creating a dynamic image gallery interface that you may find useful for your 3D App. Image galleries can use PHP (a server-side scripting language) to gather image file paths from folders, and by using some JavaScript (and/or JQuery) at the front end we can effectively generate a gallery interface dynamically. In this tutorial, the PHP server-side code is provided for you. PHP is also already enabled on the IT Web Server for students.

Your goal is to build a client-side script that reads image files from a folder dynamically, by invoking a PHP script on the server-side, to generate thumbnails on the client-side. These thumbnails can then be used as a gallery interface. You can use the gallery interface for a number of functions as previously discussed.

The gallery interface code, you will end up with, will be relatively sparse and could be improved in many ways. It serves as an introduction to PHP and provides an opportunity for you to demonstrate deeper understanding by improving and refining the code to produce a better gallery interface.

This section of the tutorial was originally developed using our own CSS rules for styling the gallery thumbnails, however since we are now using Bootstrap, we have eliminated much of the hand-built code for creating thumbnails. Thus, we now use the [Bootstrap Images](https://getbootstrap.com/docs/4.3/content/images/) and its .img-thumbnail and .img-fluid classes. The actual PHP and JavaScript code will stay largely the same though.

The code for this tutorial will be divided between four files:

* *hook.php* — a PHP script that reads image paths from a folder of your choice.
* *gallery\_generator.js* — JavaScript/JQuery code to generate the client-side gallery HTML code
* *index.html* — the gallery web page.
  + After this part of the tutorial, you will be expected to integrate the code into your Lab 5 results.
* *style.css* — the cascading style sheet of the gallery.
  + And, of course, you will need to integrate the CSS into your Lab 5 custom.css taking into account media queries.

Doing it this way makes for an independent piece of code, effectively you are creating your own Bootstrap, JavaScript, and PHP gallery component, which you can build upon. Alternatively, you could just as easily use a pre-built piece of gallery code, e.g. the Bootstrap carousel component.

## Step 1: Set up your site

We’ll start by creating a new development sub-directory in your Lab 5 web space called gallery, e.g. …/lab5/gallery — I happen to be using …/lab5/gallery because I am managing all my previous years lab codes in Lab 5. To test the gallery, choose four images (associated with the topic of interest) from the Internet and copy them in a sub folder gallery/assets/images. I choose the images illustrated in Figure 16, later on you can change these for 3D images rendered in 3ds Max, for example.

|  |  |  |  |
| --- | --- | --- | --- |
| Macintosh HD:Users:martin:Dropbox:web3d_app_2015_labs:web3d_mobile_app:lab6:gallery:images:coke_top.jpeg | Macintosh HD:Users:martin:Dropbox:web3d_app_2015_labs:web3d_mobile_app:lab6:gallery:images:sprite_top.jpeg | Macintosh HD:Users:martin:Dropbox:web3d_app_2015_labs:web3d_mobile_app:lab6:gallery:images:not_needed:dr_pepper_top.jpg | Macintosh HD:Users:martin:Dropbox:web3d_app_2015_labs:web3d_mobile_app:lab6:gallery:images:coke_can.jpg |

Figure 16: Some random themed images borrowed from the Internet

Tip! Try and choose 4 images roughly the same size and proportions, it doesn’t matter too much the actual size — adjust them in Photoshop if you have to. The following step is optional:

**Option 1:** In the past, you would have had to create thumbnails — you can find plenty of tutorials online show you how to do this. When I created these thumbnails, I happened to use the mac Preview App, this app allowed me to set a custom size to get roughly what I needed to start with — you would have then used CSS3 rules to scale down the thumbnails for desktop, tablet and mobile sized screen for the fluid grid layouts. However, because we are now using Bootstrap, you don’t actually have to create the thumbnails, the Bootstrap .img-thumbnail class will do this for you.

**Option 2:** However, I did further prepare these images as transparent images in Photoshop, that way CSS can set the background colours effectively — Tip! In Photoshop zoom in on your image and use the quick Selection Tool and Add/Subtract Selection to select the edge of your image of interest, and then select the Edge Refine adjusting parameters to get a good selection and save to a new layer, which effectively removes the background. If your images are not exactly the same size you can scale them up and down with the Image Size (to get all your images roughly to the same scale and then use Canvas Size to set the final size of the image (I chose 240x240) in Photoshop. Then save the images as png — note jpeg does not support transparent images — you can always scale/crop to get all your images thumbnails the same size.

Now, because I already have these images prepared, I will use them anyway. But, note, mine are effectively all small images already massaged into a size for thumbnails. You can choose a different set of 4 coca cola themed images, and the [Bootsrap Images](https://getbootstrap.com/docs/4.3/content/images/) content responsive and thumbnail images classes will give you what you need to create the gallery styling without the effort outlined in the option 1 above, though you still may want to better prepare your images, option 2.

So, here’s my 4 thumbnails after preparation in Apple Preview and Photoshop, you should choose whatever images you like — each image is 240x240 pixels, transparent png, see Figure 17.

|  |  |  |  |
| --- | --- | --- | --- |
| Macintosh HD:Users:martin:Dropbox:web3d_app_2015_labs:web3d_mobile_app:lab6:gallery:images:coke_can_tn.png | Macintosh HD:Users:martin:Dropbox:web3d_app_2015_labs:web3d_mobile_app:lab6:gallery:images:coke_top_tn.png | Macintosh HD:Users:martin:Dropbox:web3d_app_2015_labs:web3d_mobile_app:lab6:gallery:images:sprite_top_tn.png | Macintosh HD:Users:martin:Dropbox:web3d_app_2015_labs:web3d_mobile_app:lab6:gallery:images:dr_pepper_top_tn.png |

Figure 17: Images prepared in Photoshop

When you have created your images place them into an assets/images folder in your gallery folder, see Figure 18.

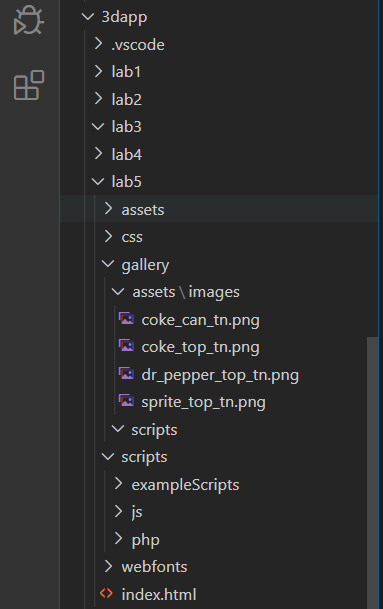


Figure 18: The gallery> assets>images> thumbnail images.

Note, that the gallery images are in the assets folder inside the gallery folder and not the main 3D App assets folder. Further, you need to also create another scripts folder inside your gallery folder because we are building the gallery as a self-contained code component that we will integrate into the 3D App later.

Note also that my images are already prepared as thumbnails from previous years, you don’t need to do this, because the Bootstrap .img-thumbnail class will prepare this for you. Instead, just put your png gallery images in the indicated image folder.

]

# 

# Integrate the Gallery into your simple 3D App

We now need to figure out how to integrate this gallery into the gallery Lab 4. In Lab 4 we simply use a .card class and stacked the images, now we need to integrate our gallery code so that these images are generated dynamically from a gallery images folder on the web server.

Once adapted, you should be able to update your 3D App at this stage to see the new gallery. The advantage with this gallery is that it dynamically, exploiting AJAX, loads thumbnails from the server. You could then use each thumbnail however you like. For example, you could use them to load the 3D models, or you could use this gallery code as an actual gallery by changing the current test images to 3D images and using a Bootstrap modal component to show the larger image — don’t forget the Bootstrap carousel component.

To integrate this gallery code into your Lab 5 results, you will need to consider the following:

* Convert the JavaScript controller, i.e. the handler, to inject div tags into the front-end view, not tables
* Modify your index.html to use the gallery:
  + That is, replace the current gallery links (3 in total) with the generated version:

<div id="gallery"></div>

* Remember the self-contained gallery generator is using identical id selectors for the 4 images. If these ids were only being used to modify styles you might get away with it (often HTML does not care depending on the browser), even though it is bad practice. So, you need to change the gallery\_cell and image\_thumbnail id selectors (Figure 31 — you will replace the img\_thumbnail with the Bootstrap .img-thumbnail class anyway) to use class selectors, and change the CSS rules to class versions too, Figure 39 illustrates.
* Look at how the HTML is being built in Figure 39:
  + For example, the ‘for loop’ initially sets a counter variable i = 0, and increments the counter until it reaches the total number of images in the response variable. The size or length of the response variable is found in response.length.
  + Take the first line of the htmlcode building code: you can clearly see that this is HTML code inside ‘ ’, which is being added as a string to the htmlcode variable. The next line then simply starts to add more HTML, i.e. '<a href="' + '' + response[i] +'">'; We can see that the + symbol is concatenating the HTML together, which is define inside the ‘ ’, the image filename returned for the response is added to the end of the URL, and so on for building the rest of the HTML code. We will use this approach in Lab 6, where we will exploit AJAX with the JQuery.getJSON() function to request data from the server via a URL and bring back a JSON object that we then parse to produce HTML for the front-end view. Opportunity! you could replace the XMLHttpRequest object with .getJSON() giving it the URL to the hook.php, and modify the hook.php to return a JSON object, etc.
* Also, you need to return the HTML code and use it in 3 blocks of HTML code id’d by the coke, sprite, pepper id selectors in the JavaScript swap function. So, we need 3 galleries, one for each HTML block, but we can’t use the same gallery id 3 times, it won’t work. A quick solution is simply to create 3 unique galleries using the same innerHTML, see Figure 39 with the return of the same innerHTML 3 times. This is possible, because I happen to have decided on 3 gallery block elements.
* Even more so, if you wanted to use this gallery in your assignment with different sets of images for each 3D object modelled, you would need to adapt the code to use the hook.php more than once, each time perhaps passing hook.php a new directory to search in.

Figure 39 shows the JavaScript code modified for the Bootstrap of your Lab 5.

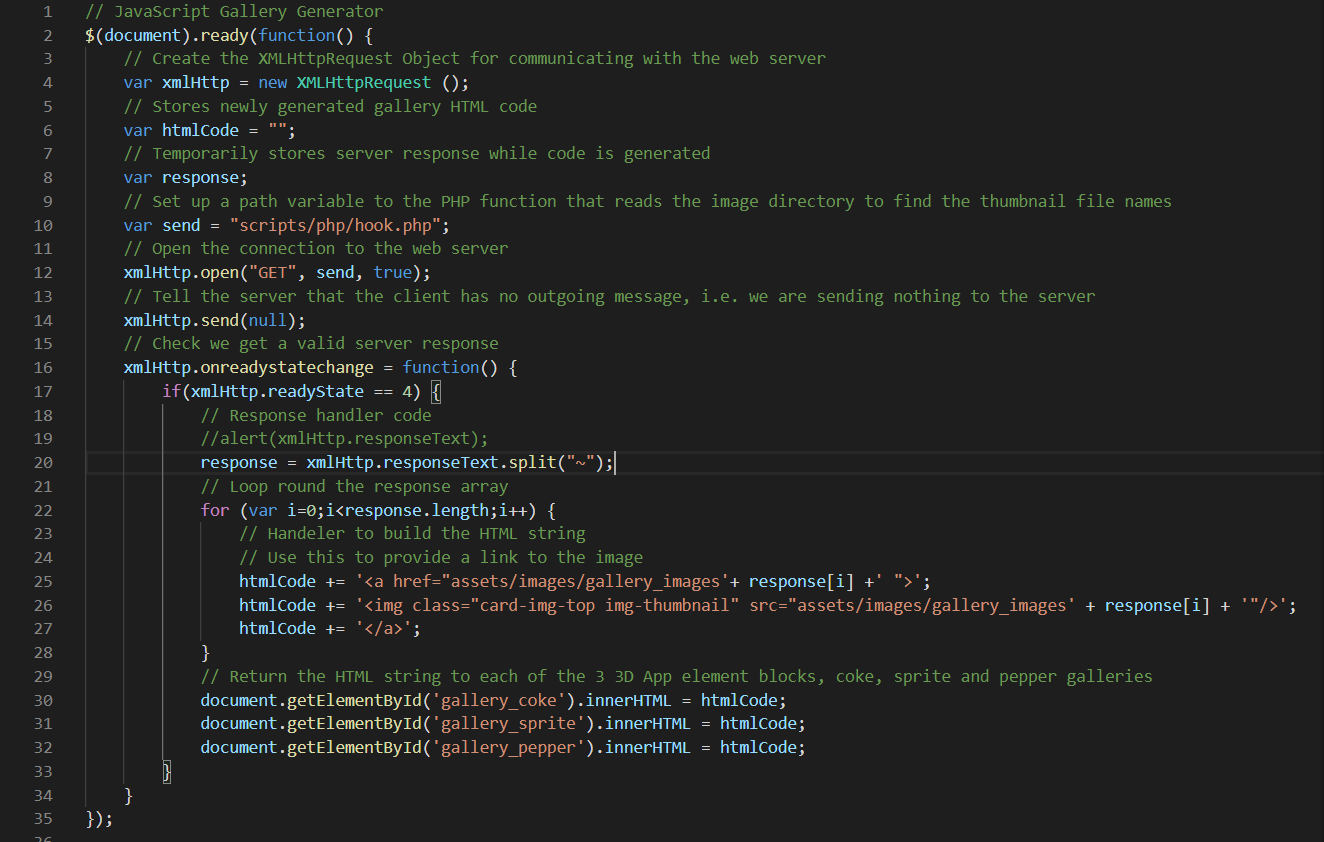


Figure 39: The JavaScript code with div tags and img-thumbnail class.

* You need to deploy the code in your Lab 5 project; at the moment, it is self-contained in the gallery folder. Compare the code for building the table in Figure 31 to that in Figure 39. We have moved the hook.php out of the /gallery/scripts folder placed it in the lab5/scripts/php folder. And we have also placed the gallery\_generator.js file in the lab5/js folder. This means changing the path to find the hook.php, and modifying the HTML code-building handler. You will note form the code shown in Figure 39 that I have insert the assets/images/gallery\_images path, which also means you don’t need hook.php to return the $directory path. This, you will need to modify the hook.php code. Note we eliminated the numberOfColumns variable, that was a nice feature, you might want to see if you can put it back to control the rows and columns of your gallery inside the .card class.
* Make sure you test your generated HTML code — check it with the Chrome inspect tool. See Figure 40 and Figure 41.

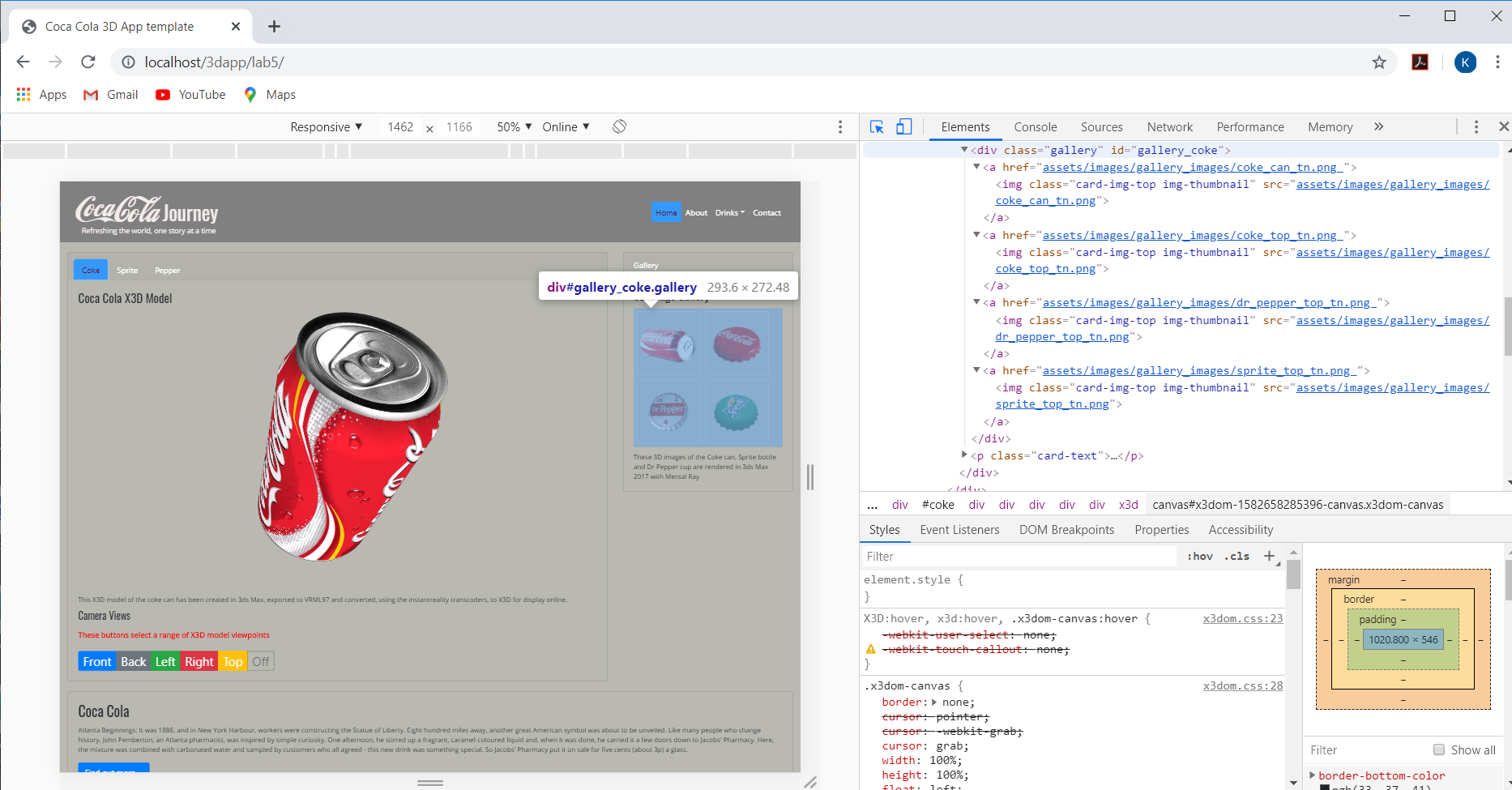


Figure 40: Chrome inspect illustrates the HTML has been correctly generated for the gallery links.

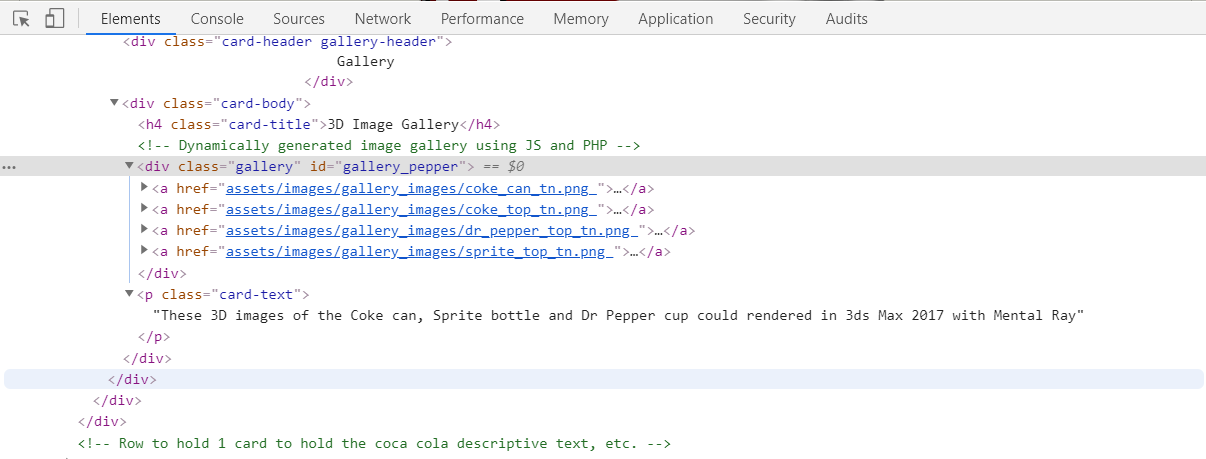


Figure 41: Inspect your new HTML gallery code

However, all we had to do was modify the Bootstrap .img-thumbnail class CSS rules a little to get a reasonable layout inside the .card class based 3D Images Gallery. Note, if we do this we will also change the home page contents images, so we need to copy the current .img-thumbnails CSS rules and create a .gallery class. Figure 44 shows the relevant HTML5 code section where re replace the hardwired 3D Image Gallery from Lab 4 with the new dynamically generated 3D Image gallery. Do the same for the sprite and pepper contents.

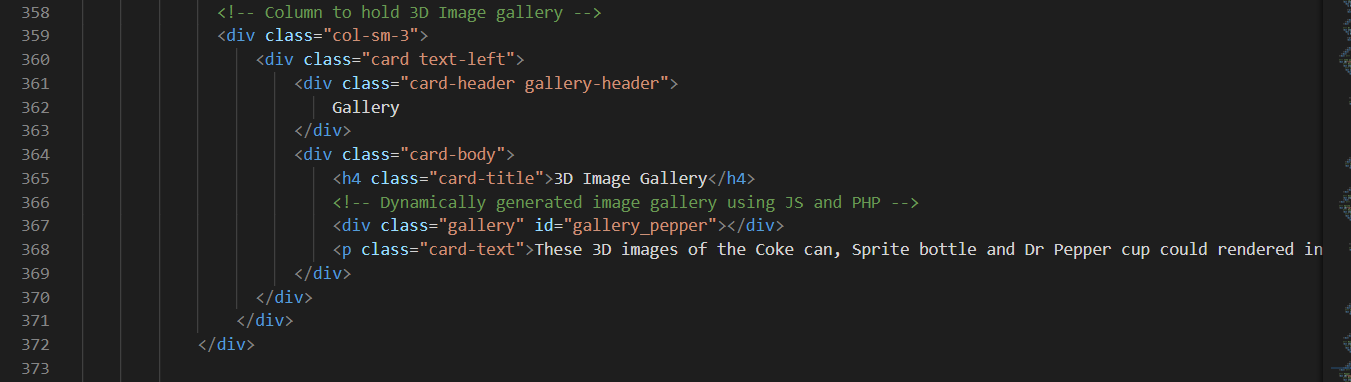


Figure 42: The Bootstrap HTML for Lab 5 for the pepper HTML block, as an example.

Figure 43 shows the new .gallery class with the .ing-thumbnail rules changed.

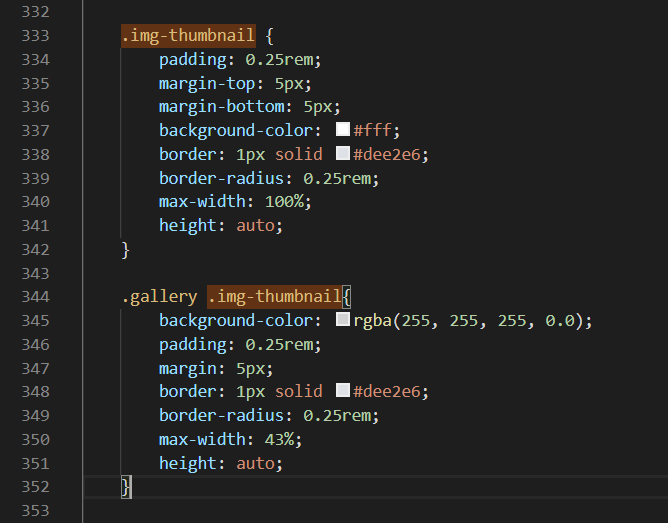


Figure 43: Corresponding Bootstrap CSS rules for mobile.

Figure 46 illustrates the new 3D Image Gallery for the coke page contents.

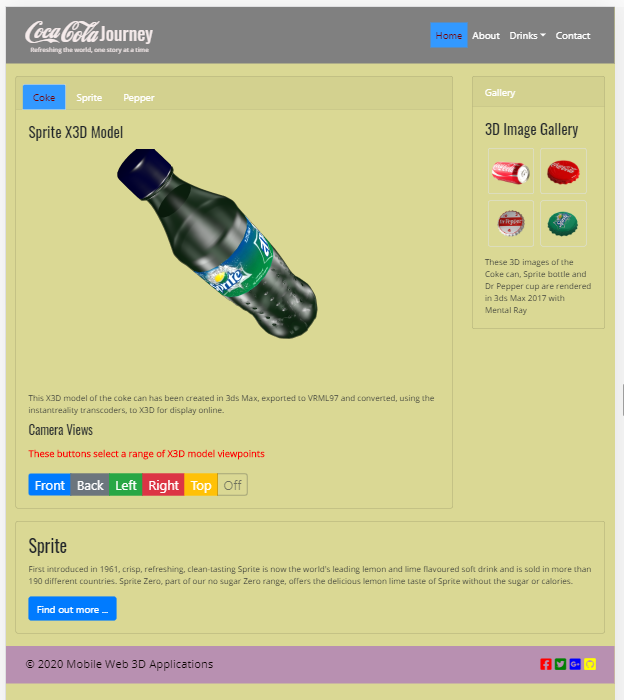


Figure 44: The new dynamic gallery in the Coke page — Bootstrap.

The gallery should look the same on the sprite and pepper pages. If you get stuck, you should check out my finished version at the **Live Feedback Site**.

# Restyle the 3D App

Back on page 12 we suggested you have a go at re-styling the 3D App. We have not done this so far, and if you haven’t either, now’s the time to do this task. As illustrated in the examples around page 11 you can adapt this code to create some useful JavaScript functions, see Figure 45. Later in Lab 6, we will use JQuery for swapping as it is more efficient, but for now use JavaScript.



Figure 45: JavaScript functions to swap content, change the look of the page, and reset or change back the look of the page.

Note that you already created the swap function, but you should create the other two JavaScript functions” changeLook(), and changeBack(), and integrate them into your 3D App. If you haven’ already done it, gather up your JavaScript functions and put them in a file called swap\_restyle.js (for want of a better file name). Put this JavaScript file in the scripts folder., and don’t forget to declare it in your index.html and set up the appropriate IDs for their associated block elements.