Unit 5 Learning Diary

Creating Javascript

I understand that I am required to create 3 program ideas and use only one. However, there's not that much time left in the course contract and I'm not keen on spending much more time on this material.

So, I came up with a simple, useful idea for the site using a javascript program that is entirely my creation. This program is for my Development page, and I'm bearing in mind potential customers (some of the personas I expect to be using the site). The script would allow you to zoom in on images of products being developed and collapse them again after you've looked. For those interesting in investing in Turri's as a future customer, you probably don't just want little thumbnails to get a good idea of what you're dealing with. You probably want to see the production process in greater detail.

Since the development HTML and CSS is substantial for this page, I don't want to develop my script within these pages, especially when I haven't programmed much Javascript in the past. So I started afresh with a blank page, and went about creating a prototype for my program, with simple red blocks on a page. All I wanted was to be able to hover over them, see some text saying "click to open" so a user knows what to do, have the block expand once you've clicked it, and then have it collapse again after another click.

This code required accessing all the elements in a page of a certain class and changing their CSS styling (width, height, visibility, etc.), as well as their HTML properties (the text within the divs). Once again, I had trouble getting this to happen in an external HTML file. So, for the time being, I developed the script entirely within the HTML file. Of course, it worked there. When I referenced an HTML element within the HTML file, it would actually store that element within a variable rather than just creating a variable with no value ("null" or "undefined"). At least I was able to create a working script this way.

Right now, the prototype is finished. Before I implement it in my actual Development page, I'll talk about how I created this one and the steps I went through to create a working prototype. I will also include the prototype in my submission for the next unit, for your convenience. It's a folder titled "eventListener" that contains three files named **eventListener.html**, **eventListener.css**, and **eventListener.js**.

The concept behind this script is simple. You have a bunch of pictures on the page, and you want to look at them closely, so you click to enlarge them. When you hover over the image, a translucent div with the text "Click to open" appears, prompting you to click. When you do, the script toggles the picture's class to another with fixed positioning on the page, different height & width attributes, and a higher z-index. "Click to open" text becomes "Click to close".

This is why I use z-indexes. Once the image is enlarged, I don't want you to be able to click on anything else in the page other than the enlarged picture, to collapse it again. Preventing the user from interacting with other elements decreases chances for bugs. To make this happen, when the image is enlarged, another translucent element comes in between the picture and the rest of the page. The picture has the highest z-index of all the page elements, the translucent div has the second highest, and everything else has the lowest. When you collapse the image again, the translucent div (**id="gray"**), becomes invisible again, allowing you to interact with the rest of the page. To make the translucent div invisible, the Javascript toggles a class on the element called "invisible" which has one CSS property – **visibility: hidden**.

In the prototype, each box is actually two containers – one container that remains on the page before and after the image is enlarged ("small_container"). I created this parent container so that when the image is enlarged, the space it USED to hold isn't empty. If that space becomes empty, other elements on the page might move, and I don't want that. I want everything else to remain stationary.

The child container (**"photo_container"**) holds the picture and the "closer" – the translucent div that has the test "click to open" or "click to close". The picture and closer both inherit the dimensions of this container. Inheriting those dimensions makes the javascript and CSS simpler and less of a hassle.

Of course, these interactive features require a lot of class toggling, which requires JS. In the JS file, I store all the "**photo_container**"s in an array, as well as all the "**closer**"s. One function handles the picture zooming as well as the toggling of the page-wide translucent div (id="gray"): **zoomIn(photo, closer).** The argument **photo** is one of the divs with the class **photo_container** and the argument **closer** is one of the divs with the class **closer**.

zoomIn(photo, closer) uses a variable declared outside of the function, called **start**, which is initially set to 0. The value 0 in this case symbolizes a clicking event. The first loop within the function is an **if** loop, and it evaluates whether or not start = 0. If this is true, the function gives the **photo** argument (if valid) the **zoomed** class, which contains the bigger height and width attributes as well as a different z-index. It also changes the value of start to 1. A value of 1 means that the photo is enlarged and other elements on the page can't be interacted with. The second loop within the zoomIn() function contains the statements that are executed when start=1: the **photo** argument has its **zoom** class taken away (it becomes smaller again), the page-wide translucent div becomes invisible again, and the closer text becomes "click to open" again.

I want this function to be applied to all the pictures on the page. I've tried storing all divs of a certain class in an array, and applying a function to that array, but with no success. In particular, I tried using the **queryAll(".classname")** selector, and failed. So, I used the sure-fire method of giving each div an ID and storing each of those elements in an array manually. I have two arrays in my JS, **photos** and **closers.** The name "photos" is misleading, because the elements in this array aren't actually the photos, but the containers that hold the photos. For all intents and purposes though, the name works. The "closers" array actually does contain all the divs with the text "click to open".

I then used a for-loop at the end of the Javascript file to loop through all items in both arrays and perform 3 operations: add a eventListener("click") that executes the zoomIn() function when a photo is clicked, a "mouseover" listener that makes "click to open!" visible when you hover over an image, and a "mouseout" listener that makes "click to open!" invisible when your mouse exits the photo on your screen.

After this prototype was complete, I had all the code and structure I needed. I just adapted the HTML in my products page to fit this prototype, added the relevant classes to my CSS, and copied the JS into the development.js file. I had to make a couple insignificant changes to the existing structure, but that's it.

Once again, this script and the respective HTML/CSS changes are made with potential customers in mind. In particular, persona 1 and scenario 2.

Links to resources I used/benefited from

Where I learned to select elements by their ID using Javascript

Where I learned about the viewport width CSS property, which allowed me to create a div that covered the entire page

Where I learned how to change elements' CSS properties using Javascript

Where I learned about other event handlers like "mouseover"

How to add classes to an element with Javascript

How to remove classes from an element with Javascript

How to get an external javascript file to run after other resources have been loaded

Side-notes

I added a small feature to the home page's CSS. The green, white, and red boxes with the questions ("What do we make?", etc.) will change color when you hover/click on them. It just adds to the aesthetics for the page and prepares for the Jquery that I'm going to add in the next unit.

I made the CSS comments less spacious.

I compressed all the images and gifs on the site and made them on average 50% smaller. That should increase the site's performance.