Bugs:

Bug #1: More than one puzzle piece can be dropped into a drop zone at any given time. There should only be 1 piece in 1 zone at any given time.

Bug #2: The pieces remain in their zones when the drop zone is changed. They should be reset to their initial position when the drop zone is changed.

Bug #3 (Optional): When the background images change, the puzzle pieces do not change their appearance when the background does and display the pieces for the first/default puzzle.

Solution Theories:

Bug #1: There may potentially be a javascript code that could have content awareness, and if this code detects something in a div/area, it will block other scripts and content from running

Bug #2: There may potentially be a javascript code that, when targeting a specific const id, will reset the const and its elements to its previous state, before the javascript was executed and finished.

Bug #3: In theory, the code used to change the background from the class example could be applied to this bug and have the pieces change when the background changes by altering the source image's id.

Research:

Bug #1: An if statement will detect if there's any children of an element, and depending on whether there are or not, it will execute a function or block a function, depending on what one codes. Adding the append child event to the if statement will block the drop event whenever there is a child of an element.

Bug #2: The hint that was given was that when "return" is used, it will stop any function, object, array, or other javascript elements and return it to its prior state, if a condition is met. However, after in class discussion, as a prior code was used to append a function, the same could be used for this bug as well.

Bug #3: This bug is a slightly more complicated bug as we defined the puzzle pieces in the code with a class. This can be changed by giving the pieces an id and targeting those in javascript, much like how a previous in class example was targeting an id and displaying it in the console log. However, adding this seemed to break the whole code, so a new direction was taken that targets the same thing but

Code Used:

Bug #1: function handleDrop(e) {

```
e.preventDefault();
  if (!this.hasChildNodes()) {
    this.appendChild(draggedPiece);
```

```
    }
}
```

Bug #2:

```
puzzlePieceDiv = document.querySelector ('.puzzle-pieces'),

Function ChangeBGImage {
resetPuzzle();
}

function resetPuzzle() {
    dropZones.forEach(zone => {
        while (zone.firstChild) {
            puzzlePieceDiv.appendChild(zone.firstChild);
        }
    })
}
```

Bug 3 (Unfinished):

(First Version)

```
puzzlePieces.forEach(piece => {
                    let newPiecePath = `images/topRight${this.id}.jpg`;
                    piece.setAttribute('src', newPiecePath);
            });
```

(Second Version)

```
puzzlePieces.forEach((piece, index) => {
        piece.src = images/puzzle${this.id}_${index + 1}.jpg;
    });
```

(Third Version)

HTML (added id)

```
<img src="images/bottomRight0.jpg" class="puzzle-image" alt="bottom right" id="bottomRight">

bottomRight = document.querySelector ('#bottomRight')

function changebottomRight () {
        let newPiecePath = "images/this.id" + index+1 + ".jpg";
        puzzlePieceDiv.style.bottomRight= `url(images/${this.id}${index+1}.jpg)`;
        id = this
}
```

Results:

Bug #1: After some trial, error, and edits, the result was achieved using the above code. In prior lines of code, we defined the class of "dropzone" in HTML as "dropZone" in html. We gave "dropZone" an event that whenever a "drop" happens, it would execute a function.

As well, we defined the puzzle pieces images as "puzzlePieces", this is significant because there is an event listener that was added for "puzzlePieces" that was called "handleStartDrag". In "handleStartDrag", we defined that draggedPiece = this. This means that any instance of "this" being used is referring to the puzzle piece when it is being dragged.

In the above code, we are telling Javascript that when a puzzle piece is being dragged, it is being defined as "this". Upon drop into the "dropZone", the if statement will check if the element has any child nodes. If there is one, it will terminate "this" from occurring and block the script, aka, it will not let the dragged puzzle piece drop.

Bug #2: Thanks to a really amazing individual, it was figured out that in order to return the pieces to their original positions, we had to make a new function that specifically targets the div element they came in, aka .puzzlepieces. From the original source, we were targeting specifically the images and not the area, so creating a new definition and then targeting that with a function solved the problem.

Not only that but the specific function used is what we described as a reset function. This function appends any child nodes that are inside the div, and the "while" element used continues the code going in a loop until the conditions are no longer true. By putting the title of this function inside of the changeBGImage function, we are calling upon the reset when we activate the changeBGImage function. As the background image is changed, javascript will remove and reset the childnodes within the dropzones and return them back to their original positions.

Bug #3: The code used above is a modified version of the theory where using similar elements to the changeBGImage function. This code is functional to an extent, as it will indeed execute the function to

change the images into the desired pathway/destination. However, the problem is that there is a raw “topRight” element in the pathway, and thus, because it is targeting every single piece, every piece will change to the images with the “topRight” name with the id.

Another version of this code was also discovered that solves the “topRight” part by changing that portion to be the id, and then a value is given. The problem with this code is that, for some reason, it completely breaks the rest of the functions.

In theory, a solution to fix this is to give each piece a specific id, and then target those in Javascript. Then, using that target, a function or pathway can be created that will target that id in the image’s pathway, and using the index+1 element from the second version, the pathway can turn out to be something like “topRight1.jpg” and will only apply to the image with the “topRight” id. However, the problem with this code remained much the same as the second version that, when tried, broke the whole thing or didn’t work. Even with tweaks, there was no success.

After referring to another source for help, there was an alternative, and completely different, method that was proposed, however, as this was an optional bug to solve, I elected to hold off for the time being. I want to see if I can fix my solution at a later point in time. The first version is functional to a certain extent and I want to see if I can gain knowledge in the future to solve it on my own where it can be fully functional. I did want to make acknowledgement and record of the attempts in this document.