Tackling Bugs

Bug 2

I decided to take on bug 2 first, because 1: I felt it would have been the harder bug (this would prove to be less hard but more time consuming) and 2: the reset was more important to me, as at least if you knew where the pieces would go, each set would be updated upon selecting each board. Upon first glance the problem and solution is simple: The pieces from the previous puzzle are staying in their containers when switching boards. Solution? Move them back to their starting container.

Problem Solving

My first thought is just deleting the images from the drop-zone divs and re-creating them back into their original div. Not only would this have been repetitive and time consuming, it was pointless as they wouldn't add back into their original places.

Now my plan is to target the parent (dropZones) and just remove the child nodes and have the createPuzzlePieces function create the new pieces. This was the right first step, although I keep receiving dropZones.parentNode or dropZones.removeChild() is not a function. I have spent hours on this, and clearly this is not the fix.

I have realized that I need to define a variable inside of the function for it to actually count as a function. Using length as a definition for how many images need to be on the board for reset (0), I can tell it to remove all of the child nodes from the drop-zones as long as they are inside of the drop-zone divs by using the "while" tag.

Conclusion

The final result yielded the outcome I had wanted. Defining the variable inside of the resetPuzzlePieces function reads it as a proper function. Using "while" to define that an action needs to trigger while "puzzle-image" is inside of a parent div, that parent being drop-zone. As long as there are 0 pieces on the board, it removes each child node, and allows CreatePuzzlePieces re-create the images for the new board.

Bug 1

I'm tackling bug 1 second, as it's the lesser valued bug to me. If you can get the pieces in the right spot, there is no need to worry about stacking, just the reset. That said, misclicks happen and people place pieces in the wrong spot, so that's where the stacking problem comes in. Problem? Pieces are stacking either on top of each other, or actually stacking inside of each other. Solution? Lock the ability to drop a piece in a drop zone when there is a piece already in there, without preventing the ability to move pieces around the board from zone to zone. Much, MUCH easier said than done.

Problem Solving

So obviously the "return" syntax will be useful, otherwise Trevor wouldn't have shared it with us. Question is, how. Everything I search has return only used for mathematics. My first thought is to disable the prevent default while there is an image inside of a drop-zone div.

So "while" is absolutely needed. With some research, != is a simple way of comparing whether something is not equal to another object. I can use this to define when something is inside of the drop-zone, and turn off the prevent default.

With more research, "return false" is actually something we have used before. It's used to prevent an event from firing. If I can have the script read the HTML and only fire when there is nothing inside of the drop-zone, this task will be done.

Upon realizing how stupid I am, I've already used part of what I need elsewhere. Since prevDrop is e.target, and in this case, the target is the drop-zone as the classlist.contains definition says, I can just use .length > 0 again to define that if there is something in the drop zone, to return false. And if there is nothing, use prevent default and allow an image into the div.

Conclusion

This one was really tricky, because a lot of the time I had just kept preventing myself from dropping anything inside of the drop zone since I would either previously delete it with remove child, or I never actually had an event for allowing a drop, just preventing it. In the end, using the MDN return page, with a little help from Google (a little meaning around 15 articles from CSS tricks, Code Burst and more) and incredible patience that waned countless times, defining prevDrop as the event target, which would end up

being defined as drop-zone, I declared the drop-zone as the parent. Drop-zone then read whether or not something was inside of it when attempting to drop an image inside of the drop-zone. If there wasn't anything, it skips the return false and allows the drop. If there is something in the drop-zone, it would know through .length > 0 and prevent default would never fire, preventing dropping inside of that drop zone.

Full Conclusion

Obviously there are still problems. If there were multiple puzzles, it would likely reset all pieces instead of just one puzzles, you could drop pieces from one puzzle to a completely different one, so on. But as it stands, the puzzle works as intended. The pieces are to the left, you can drag one piece onto the board and not stack pieces. And when you have finished the puzzle, or struggled and want to try a new one, you can select one of the other puzzles and the pieces will reset back to the left, clearing the board.