# Crushing Bugs

There are two bugs in the in-class build files:
1. You can drop more than one puzzle piece into a drop zone - there should only be one puzzle piece at a time.
2. The pieces don't restart and stay in the drop zones when you change the puzzle.

Solutions:
1. Check if there is already a child. If the 'children' property of the drop-zone element has a length greater than 0. If it does, then there is already a puzzle piece in the drop zone and returns early. If the length is 0, then there is no puzzle piece in the drop zone and the function proceeds to add the new puzzle piece to the drop zone.
2. Loop through all the puzzle pieces and use removeChild( ) which removes them from their current parent node

**Notes:**

### JS Element Children Property
- The 'children' property of 'Element' returns a live 'HTMLCollection' of child elements given the element.
- The 'HTMLCollection' returned by 'children' is read-only and does not include non-element nodes such as text nodes.
- The 'children' property can be used to access a specific child element or manipulate child elements of an elements
- 'childNodes' returns all child nodes, including non-element nodes like text
- If the element has no element children, then children is an empty list with a length of 0.

**https://developer.mozilla.org/en-US/docs/Web/API/ParentNode/children**

### Checking if element has children using JS
- Children of an HTML element are the elements that are nested inside the parent element
- There are three methods of checking if an element has children in JavaScript:
  - Using the 'childNodes' property to check the number of child nodes
  - Using the 'firstChild' property to check if the first child node exists
  - Using the 'hasChildNodes( )' method to check if the element has any child nodes

**https://www.geeksforgeeks.org/how-to-check-if-an-element-has-any-children-in-javascript/**

### Remove Child Node
- The 'removeChild( )' method is a built-in method that allows you to remove a child node from its parent node
- How to use the method:
  - First, select the parent node that contains the child you want to remove
  - Then, you can call the 'removeChild( )' method on the parent node
- if the child node is successfully removed from the parent node, 'removeChild( )' returns the removed node
- if the child node is not found or can't be removed, 'removeChild( )' returns 'null'

**https://developer.mozilla.org/en-US/docs/Web/API/Node/removeChild**

### Function Exit Techniques
- in JavaScript, you can exit a function in several ways
  - Use 'return' statement, which returns a value and exits the function

    - If you don't get a return value, you can use 'return;' to exit the function without returning anything
  - throw an error using 'throw new Error('error message');' - this will stop the function's execution and jump to the nearest catch block (if there is one)
  - Use 'break' or 'continue' statements to exit a loop within a function
    - 'break' will exit the loop entirely
    - 'continue' will exit the current iteration and move on to the next one
  - in some cases you might use a 'finally' block to ensure that certain code runs regardless of how the function exits
- Handle errors appropriately in your code to prevent unexpected exits from your function

**https://flaviocopes.com/how-to-exit-a-function-javascript/**

## Function Exiting in JavaScript

- in JavaScript, you can exit a function using the 'return' statement
- The 'return' statement can be used to return a value from a function or to exit the function early
- If you want to exit a function early without returning a value, you can use 'return' with no arguments
- You can use 'return' with a value to exit the function and return a value at the same time
- If you have nested functions, you can use 'return' to exit the inner function and the outer function at the same time
- In some cases, you might 'throw' an exception to exit a function