Multidimensional arrays

Selecting transcript lines in this section will navigate to timestamp in the video

- Since arrays are container objects and data structures, they don't have to just hold primitive types. We can actually have containers within containers. If we go back to our movie theater example, we can actually represent all the seats in a movie theater with something called a multidimensional array. Before, our array only worked for one row of seats, but with a multidimensional array, we can add the dimension of a column. This means that our multidimensional array actually holds arrays inside of it. Keeping this in mind, we can actually apply this idea to other examples. Say you're planning a dinner where guests have to choose between three options for their appetizer and three options for the main course. To store this information in code, we can use a multidimensional array. We'll have one array that stores our appetizers and another array that stores our main courses. This is just like our movie theater example in that our appetizers are like one row and the main courses are another row. Essentially, we will be grouping our appetizers and main courses into two separate arrays, but a container holding both arrays will still maintain the connection between the courses of the meals. Just like with regular arrays, we can access each item by index, but the indices work a little differently. For multidimensional arrays, we can use two numbers, a pair, to describe where a specific item lives. In this case, the first row will represent our appetizers, and the second row will represent our main dishes. This means that salad, which should live in the first row at the first slot, would be at index zero, zero. The first number in the pair represents what row it is in, and the second number in the pair represents what column it's in. Moving along, soup is at index zero, one because it is in the first row, appetizers, and the second item in this row. Cheese plate is at zero, two, making it the last appetizer in this first row. For our main dishes, they will be located in the second row. Chicken is at index one, zero because it is the first item in the second row. Salmon will be at one, one, and lasagna is at one, two. The number of rows in this array is just the length of the actual array. It's the number of courses the meal will have. As for the number of columns, we know that each course or row has the same number of choices. That means we can access any of the interior arrays, get the length of it, and that is how many columns we will have. So, why would you ever need to use these? Well, sometimes you might need to iterate through the entire array, and those values can come in handy when working with for-loops.