* - In the real world, as in programming, we often deal with lists or arrays of objects.
* This could be because we need to group a bunch of objects together or because we need to apply some sort of order to a bunch of objects or a whole array of other reasons.
* When I think of an array, I always picture a rack like this one with slots for each object to stand on.
* I can then add objects to the array, retrieve objects from the array, delete, reorganize or reposition objects and change the objects within the array while they are inside the array.
* In programming, we use arrays a lot to store, retrieve, and process data.
* And there are a whole array of functions and methods to help us work with and manipulate these arrays to add or remove objects, find objects, reorder objects, et cetera.
* In JavaScript, these methods have names like pop and push and shift that at first glance seem a bit strange, at least when you think of them as code, but when you see them in action in real life, they actually make a lot of sense.
* Here, let me show you.
* All right, I have an array of five items, starting here on the left side in slot zero, in programming the first lot of an array has to index number zero because of math reasons, and ending right now in the fifth object in slot number four.
* Now, let's say I want to ***grab the last object off the array.***
* Rather than finding out how long the array is and asking for the correct slot number, I use a method called **pop** to literally pop the object out of the array and return it.
* If I then wanted ***to add the object back onto the end of the array***, I push it back onto the array using a method called **push**.
* Now these two operations are relatively simple because all I'm doing is removing or adding something to the end of the array.
* So the only thing that changes is the total array length.
* To take the first item out of the array, I need to **shift** the other items to fill in the empty space.
* I do this with the method called **shift**.
* If I want to **add an item into the first slot of the array**, I effectively reverse this shift, so the method is called **unshift**.
* and yes, this is a bit strange.
* But now that you see what's happening inside the array, the items shifting to take in new positions, the methods names make sense.
* There are many more methods with names like **reverse, sort, slice, join**, et cetera.
* And in almost every case, visualizing a physical array like this one in front of me helps explain what each of these methods do.
* In this chapter, we'll explore arrays in detail to see how they work, how to work with them, and what all of these different methods do.