What are sets?

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

- Another way we can structure our data is with a set. Similar to linked lists, hash maps, and other data structures, a set is an abstract data type. It's an idea that can be implemented in many different ways. In the simplest of terms, a set is a collection of unique items. The order of these items doesn't matter, but none of the elements are duplicated. So why would we want to keep track of data in this way? Well, sets are a way of grouping things with a common property. For example, you might create a set that represents types of clothing you can wear, or a set with three primary colors. You could also create a set that represents the names of the fingers we have. Think pinky, ring, middle, and index. Unlike the other data structures we've seen, here we don't care about the order of the elements, and often, we don't even want to retrieve a piece of data. That's why we don't have an index, or key, or anything specific to look up the value. Here, we care about membership. With sets, we are usually testing if a piece of data is a member in that set. We care about if the set contains a given number, character, or string, and in implementation, this structure will be optimized for this specific operation. So implementation, let's jump into that. As we've seen before, certain data structures are great for lookup or search, and others are not. In an array, we might have to do a linear search of the entire array. If we want to find a specific object in a linked list, we have to start at the first node and traverse through the list, and even with an associative array, we can only get directly to the object if we know what its specific key is. Now behind the scenes, sets are actually using the same idea of hash tables most of the time. Instead of hashing a key to store a separate value object, when you're using a set, the key is the value. We don't have two pieces of information, we're just adding one. So a set works by taking an object, hashing it, and then using the generated index to store the object itself. Then, to check membership, to find out if we already know about a particular object, we just repeat the process and see if this object is already stored. We only care about membership, if the object is already in the set. In checking to see if an object is in the set, we already need to have the object itself, which is why we never use sets for retrieval of data.