Operations on linked lists

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

- Just like arrays, there are specific operations we can use to add, access, delete and search for items in a linked list. To add an item, it's easiest to insert at the back or front of the list. When inserting at the front, we can initialize the new node with the appropriate data and set the new node's next pointer to point to the first node. When inserting at the back of the list, we take the last node of the list and set its next pointer to our new node. This new node will have its associated pointer pointing to nothing since it's the last item in the list. To access an item, we don't have to have an index like we do with an array. This means we have to follow the pointers until we find the item we want to access. So, if we want to access the third item in the list, we would need to grab the first item, follow its pointer to the second item, grab the second item and then follow its pointer to the final item we want to access. You might be thinking, this sounsds very similar to searching and that's because it is. When searching for data, we follow the same method. We have to traverse through the entire list to find anode with a specific value or find out that the data does not even exist in the list. This is also true for deleting an item. To delete an item, we first have to find the item and then update the next pointer of the node preceding and following that node. We can also insert into any place within the list. Let's say we wanted to add a train car somewhere else in the list. We'd have to follow the pointers to that specific place and then update the pointers so that the previous train car points to our new train car and our new train car points to the next element. These operations are the foundations of the linked list data structure.