Introduction to tree data structures

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

- Before jumping into tree data structures, recall linked lists. With a linked list, we had an ordered list of nodes that we could traverse forward or backward. Each node contains some data, an integer, string or something else and pointers to the next node and previous node if this was a doubly linked list. Like a linked list, a tree data structure contains nodes but it's really a collection of nodes. The main difference is that the nodes in the tree might be linked to one, two or more nodes. While we could represent this visually in all sorts of confusing ways, the best visual representation is a tree structure. Now again, how these objects are physically allocated in memory is unimportant. This is a logical data structure we're talking about. It's an idea, just like a linked list, there's always a specific starting node in a tree data structure. We call it the root node and when drawing the structure on paper, we usually draw it out like this. Now this root node can contain data as well as links to other nodes. In C++ these links would be pointers. In most other languages, they'll be object references. These next nodes, these linked nodes we call child nodes. The root is the parent of both of these child nodes and the root itself has no pair. However, each of these child nodes can also be linked to other nodes therefore becoming the parent to the newly linked nodes and so on and so forth. So, this idea of parent and child nodes continues down the tree. This means a node can be a parent and a child at the same time. Taking a step back, we can think of a linked list as a very specialized type of tree structure where each node has only one link and that link is to a child node. Now bringing in some more terminology here, we also call child nodes with the same parent siblings. And if we have a node with no children, we say it's a leaf or a leaf node. For a tree, each node can have many children, just one or none at all. We put constraints on the exact structure of the tree and take these constraints to optimize a certain operation. For example, let's say we have the constraint where there is a maximum of two child nodes for any given node. Those child nodes can have children but no more than two each. This definition of a tree is pretty broad but we can add more constraints that give us different functionality.