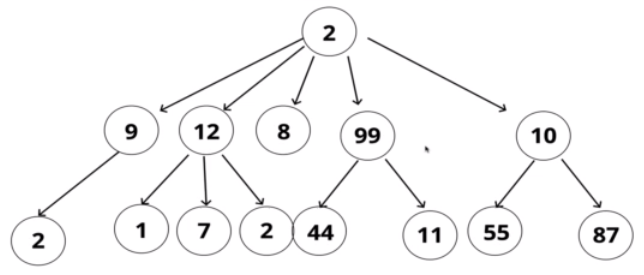
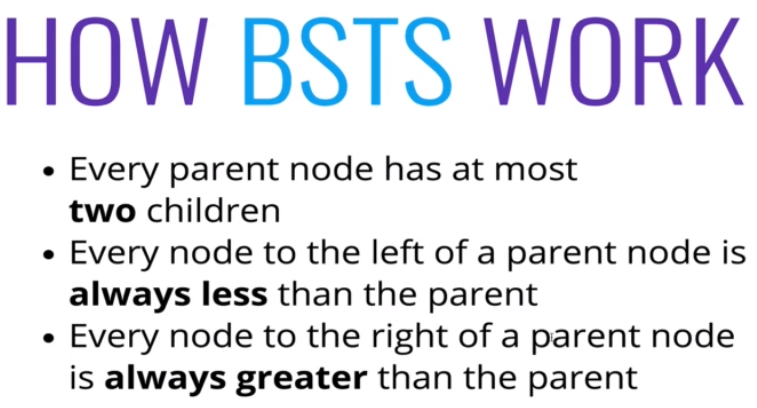
JavaScript Algorithms and Data Structures Masterclass

# Section 22: Binary Search Trees (BST)

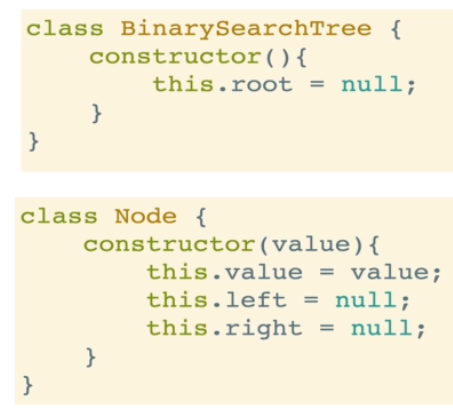
## What is a Tree?

* A tree is a DS that consists of nodes in: **parent / child** relationship
  + Nodes can only point downwards
    - **Diagram**
      * 
* **Comparison of Lists and Trees**
  + Lists are LINEAR
  + Trees are NON-LINEAR
* **Terminology**:
  + *Root* – **Top** node in a tree
  + *Child* – node directly **connected** to another node (move away from the root)
  + *Parent* – node which has **following** node connected to it
  + *Siblings* – group of nodes w/ **same** parent
  + *Leaf* – node with **no** children
  + *Edge* – the **connects** between nodes
* **Use Cases**
  + HTML DOM (JSON code is a tree)
  + Network Routing
  + Abstract Syntax Tree
  + Artificial Intelligence
  + Folder structure in OS

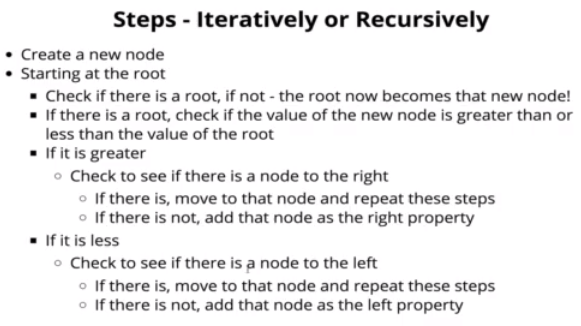
## What is a Binary Tree?

* Each node can only have up to 2 children max. In this tree type
* BSTs primarily consist of nodes with values that are comparable (ex. Numbers)
  + **Special property**
    - Every value to the left of the root/parent node is less than it
    - Every value to the right of the root/parent node is more than it
  + 
* Searching through BST
  + Compare right/left, then cut the tree in half

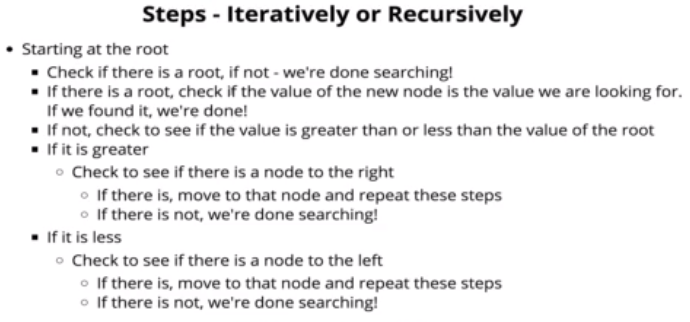
## Implementation

* BST Class
  + 

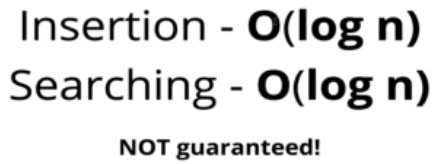
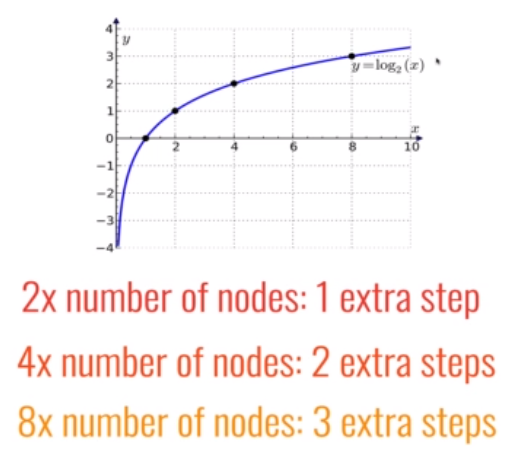
### Insert()

* Add a new node to the BST in the correct location
  + **Psuedo-code**:
    - 

### Find()

* Search if a value is in a BST
  + **Psuedo-code**:
    - 

## Big O of BST

* **Best and Average Case**:
  + 
    - \*\*\*Log (base 2)
    - ONLY applies to BST (sorted tree)
    - 
* **Worst Case**: *This is O(n) time in a binary tree which models a SLL*
  + 