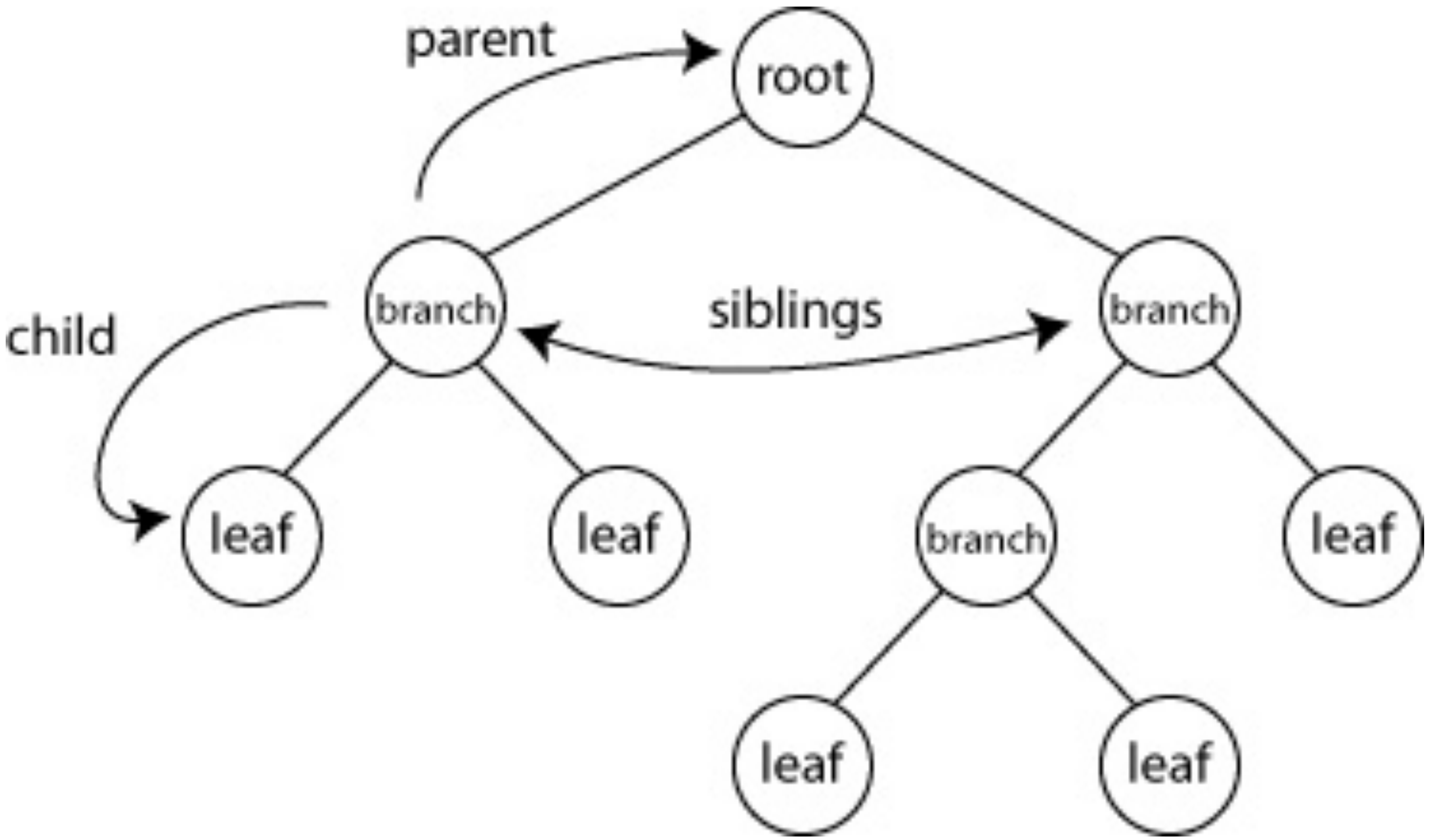
Tree

data structure composed of nodes. each node has a value, and a set of (zero or more) nodes that it references.



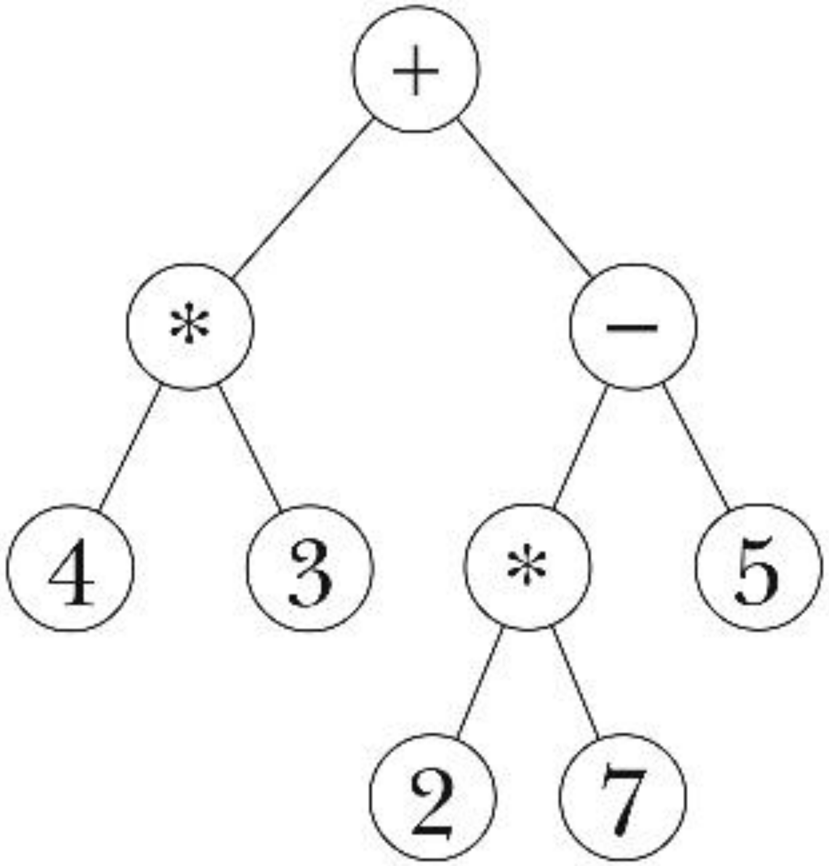
http://www.jslab.dk/gfx/trees1.jpg

Recursive Data Structure

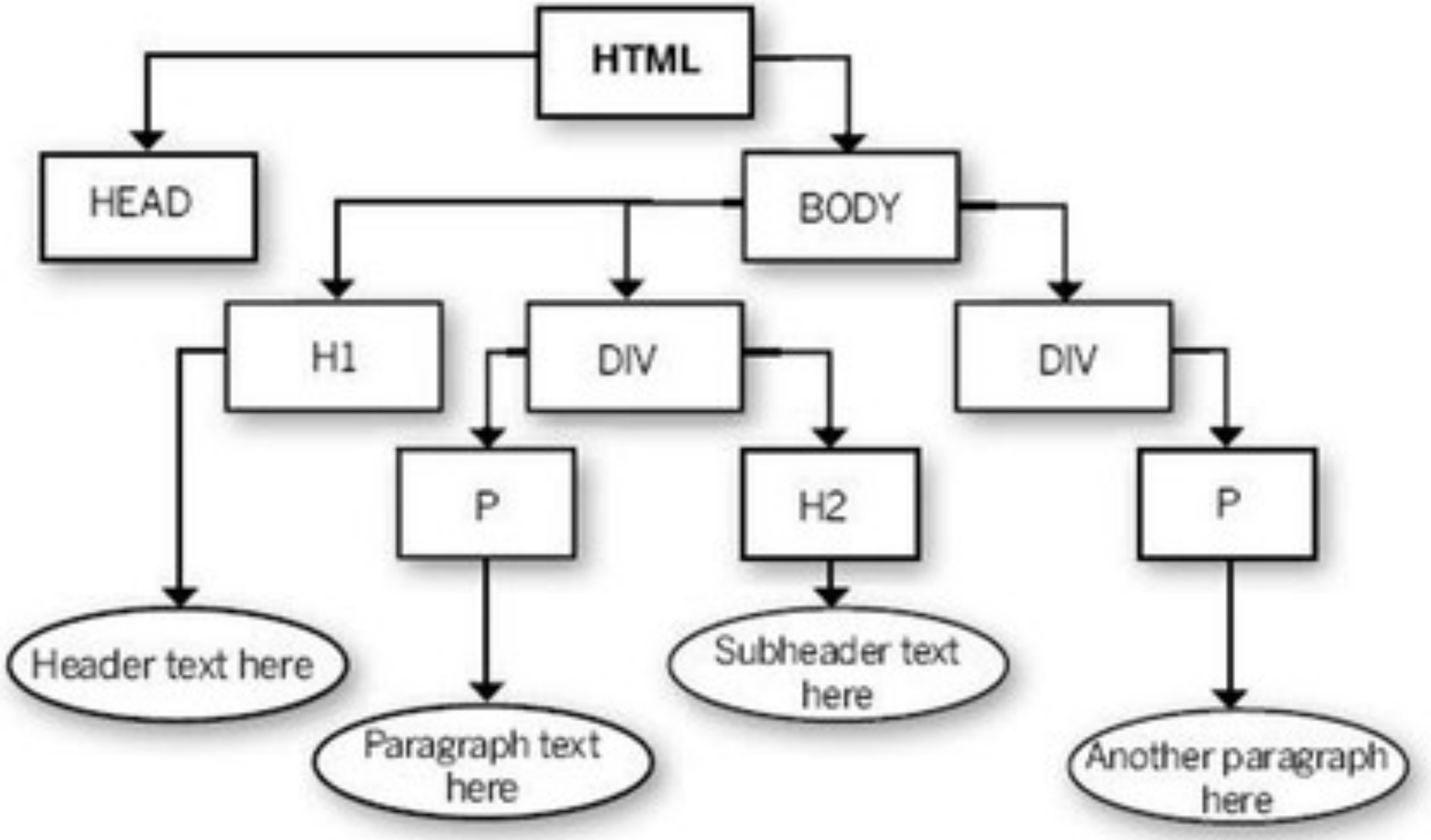
one that's composed of smaller versions of the same type of data structure

Uses

Parsing

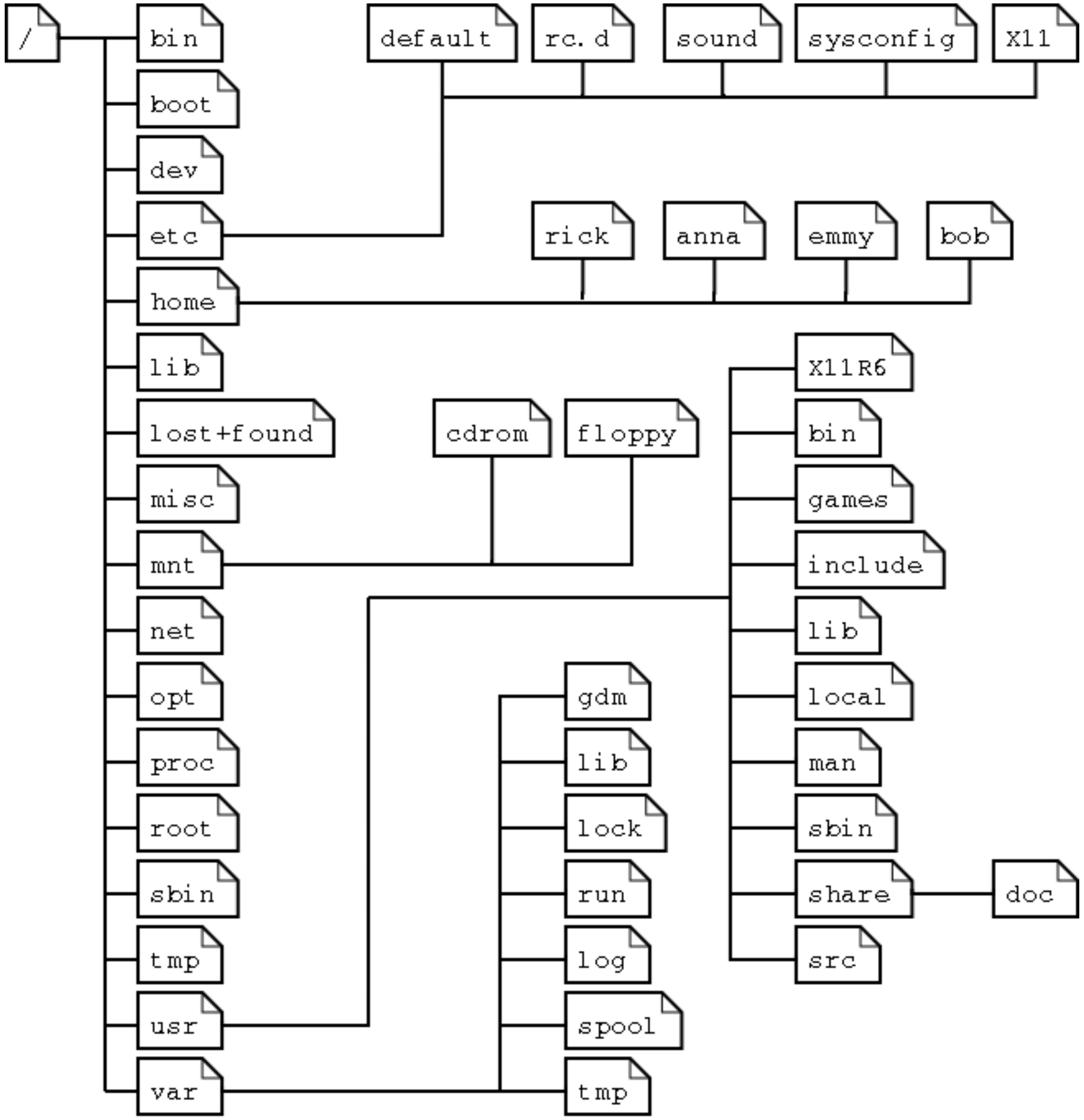


http://i.stack.imgur.com/Gg7FD.jpg



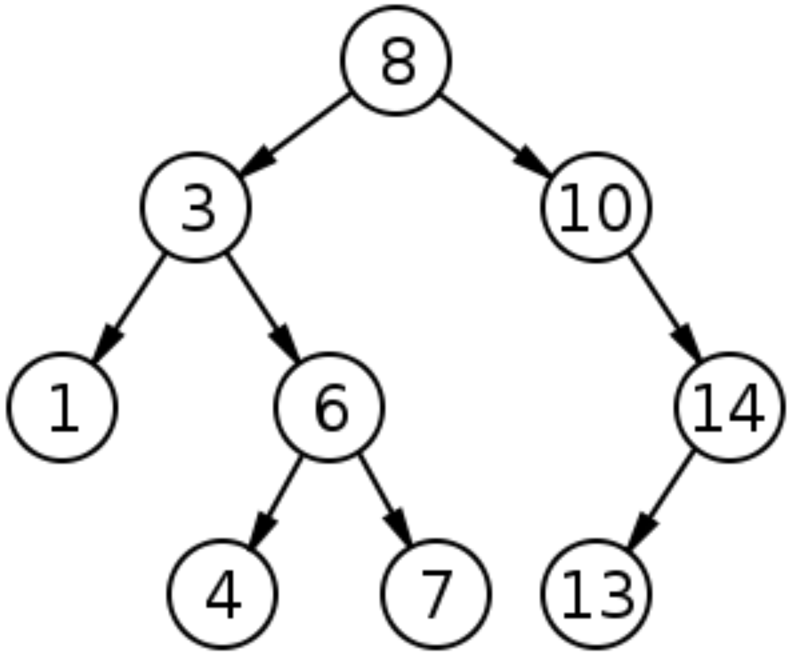
http://cdn0.mos.techradar.futurecdn.net/Review%20images/Linux%20Format/Issue%20118/DOM%20tree%20inline2-420-90.jpg

File System



http://tldp.org/LDP/intro-linux/html/images/FS-layout.png

Search



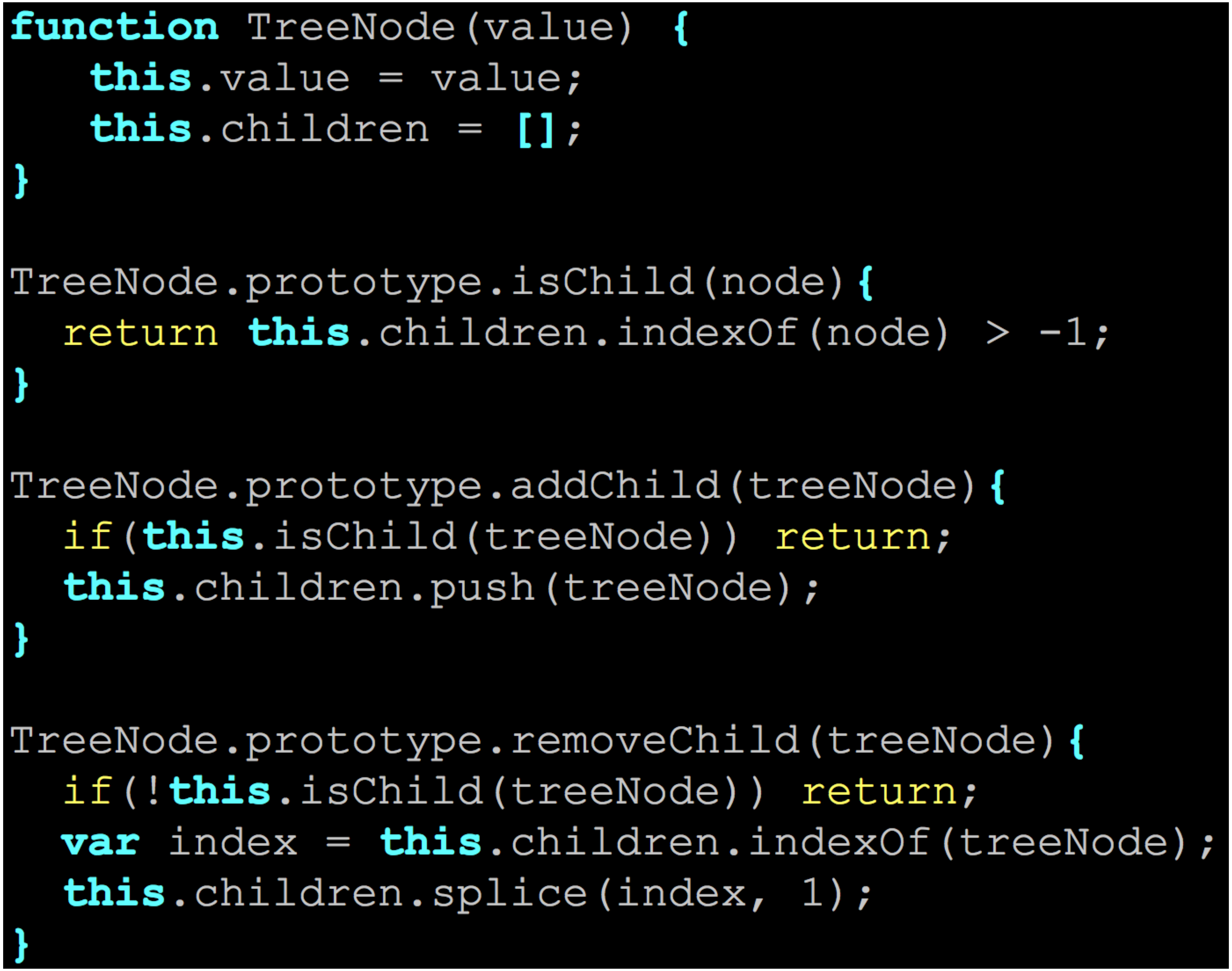
http://programminggeeks.com/wp-content/uploads/2014/01/nodes-in-binary-search-tree.png

Tree Node

holds a value and references to child nodes.

Try Me

define a constructor called TreeNode



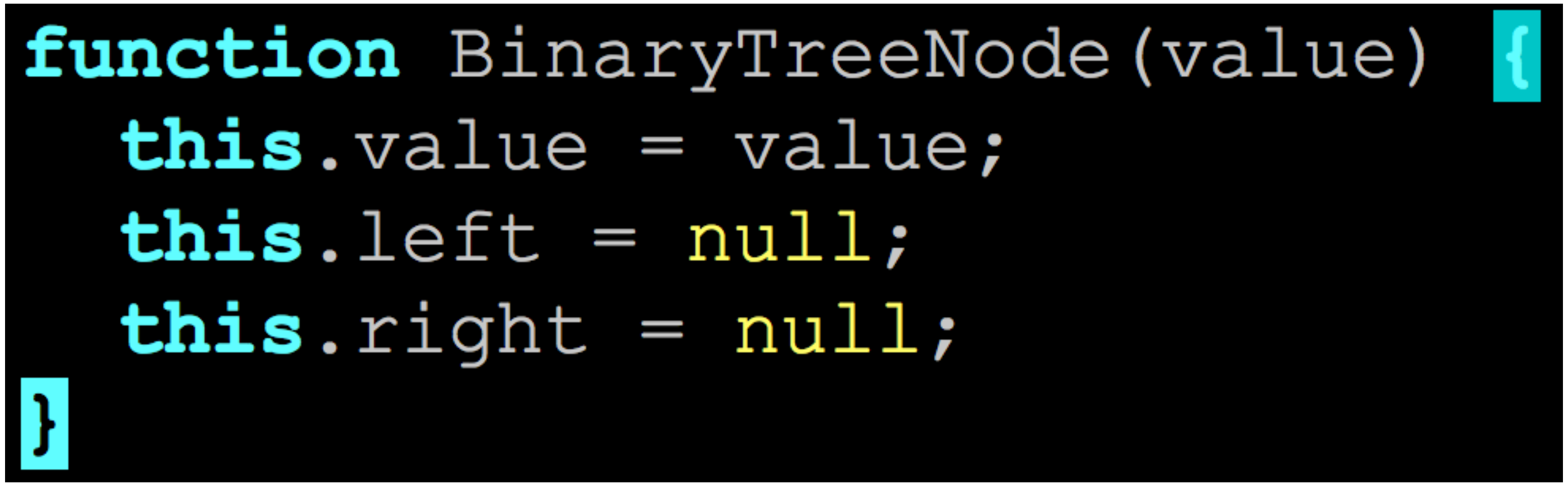
```
function TreeNode(value) {
   this.value = value;
   this.children = [];
}

TreeNode.prototype.isChild(node){
  return this.children.indexOf(node) > -1;
}

TreeNode.prototype.addChild(treeNode){
  if(this.isChild(treeNode)) return;
  this.children.push(treeNode);
}

TreeNode.prototype.removeChild(treeNode){
  if(!this.isChild(treeNode)) return;
  var index = this.children.indexOf(treeNode);
  this.children.splice(index, 1);
}
```

Binary Tree Node

Each node has at most two children. They are referred to as *left* and *right*.

Try Me

define a constructor called BinaryTreeNode



```
function BinaryTreeNode(value) {
  this.value = value;
  this.left = null;
  this.right = null;
}
```

Height

distance (number of edges) from the root to the furthest-away leaf.

Try Me

what are the largest and smallest possible heights for a binary tree with n nodes?

Tree Traversal

iterating through every element in a tree

Depth First

visit a node's children before its siblings.

Breadth First

visit a node's siblings before its children

Binary Search Tree

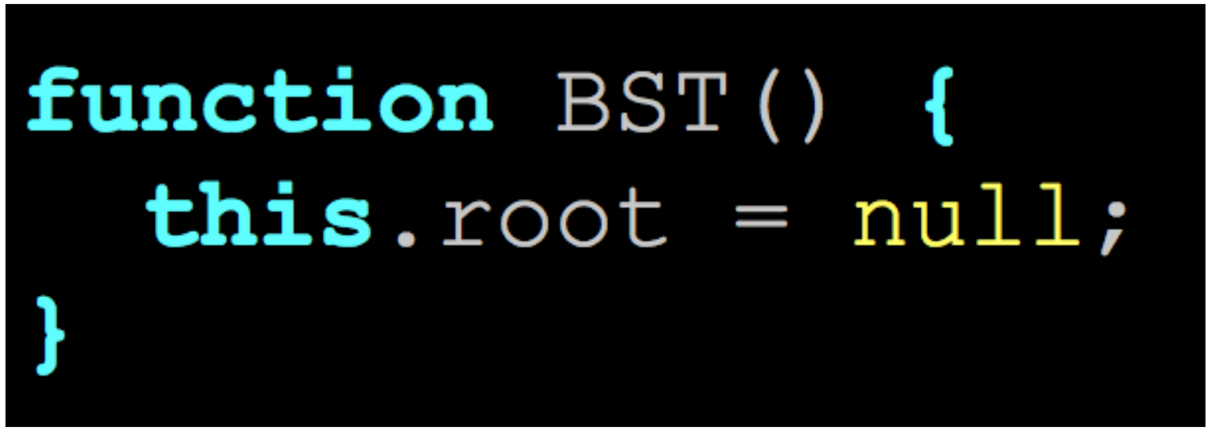
left children are smaller than their parents. right children are larger than their parents.

BST Interface

insert, delete, contains, max, min

Try Me

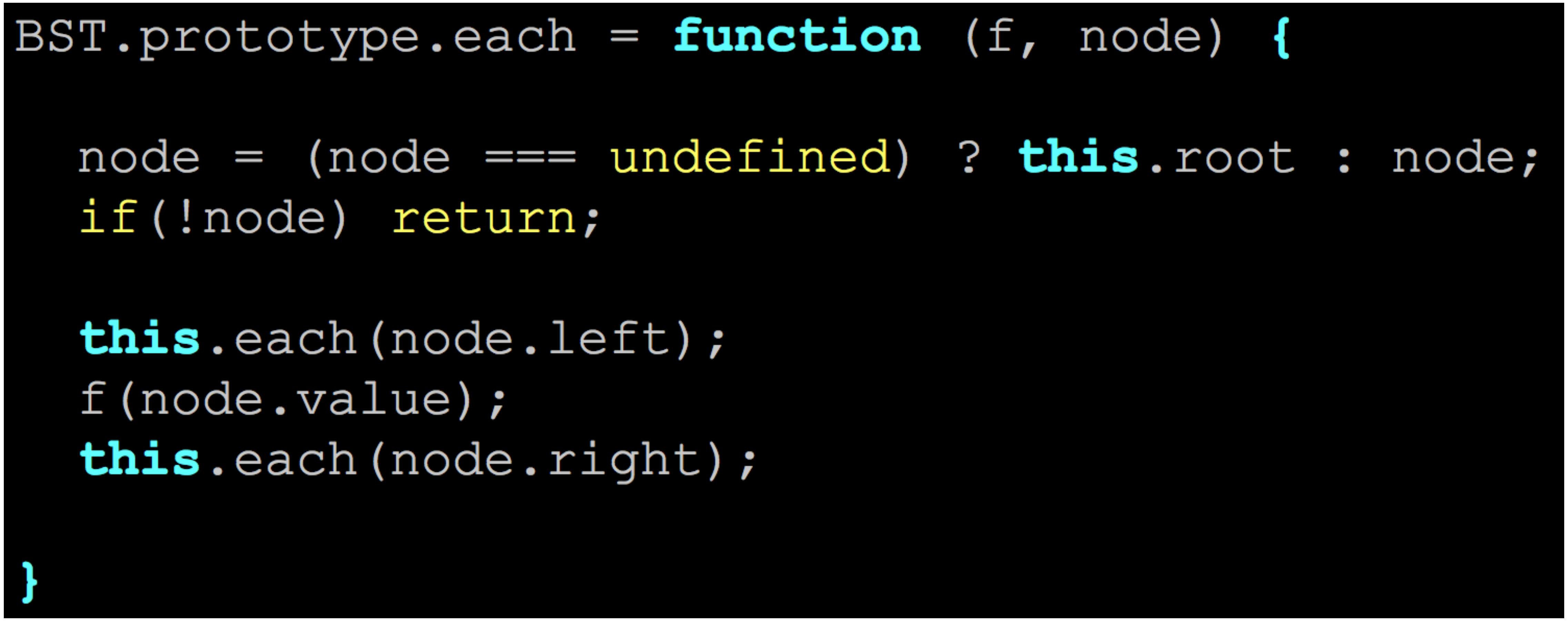
create a BST constructor



```
function BST() {
  this.root = null;
}
```

Try Me

Give the BST constructor a method called each. Each takes a function and applies that function to each element in the BST, in order of the smallest to the largest values.



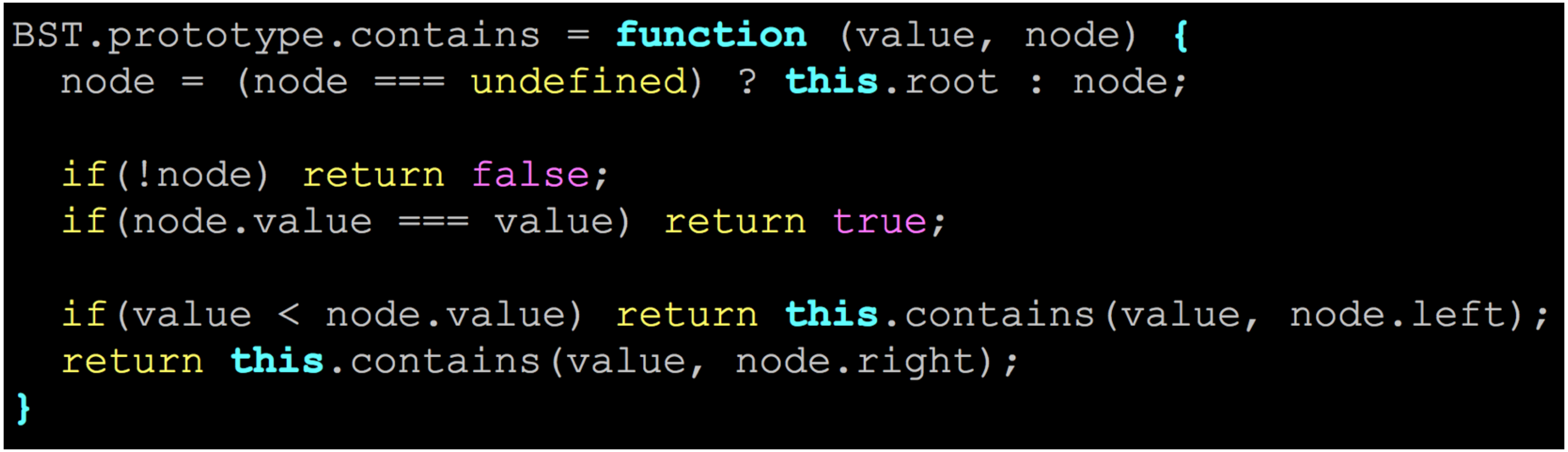
```
BST.prototype.each = function (f, node) {

  node = (node === undefined) ? this.root : node;
  if(!node) return;

  this.each(node.left);
  f(node.value);
  this.each(node.right);

}
```

Searching a BST

if x is less than the current element, go left.
if it's more, go right.

Try Me

Give the BST constructor a method called contains. Contains takes a value and returns true if that value is in the tree, false if it isn't.



```
BST.prototype.contains = function (value, node) {
  node = (node === undefined) ? this.root : node;

  if(!node) return false;
  if(node.value === value) return true;

  if(value < node.value) return this.contains(value, node.left);
  return this.contains(value, node.right);
}
```

More Practice

Write a function that takes a binary tree node. Return true if its corresponding sub-tree is a BST. Otherwise, return false.

http://codercareer.blogspot.com/p/binary-tree-interview-questions.html

Given a BST tree and a value x. Write a function closest(tree, x) that returns the value in tree that's closest to the value x.

http://codercareer.blogspot.com/p/binary-tree-interview-questions.html