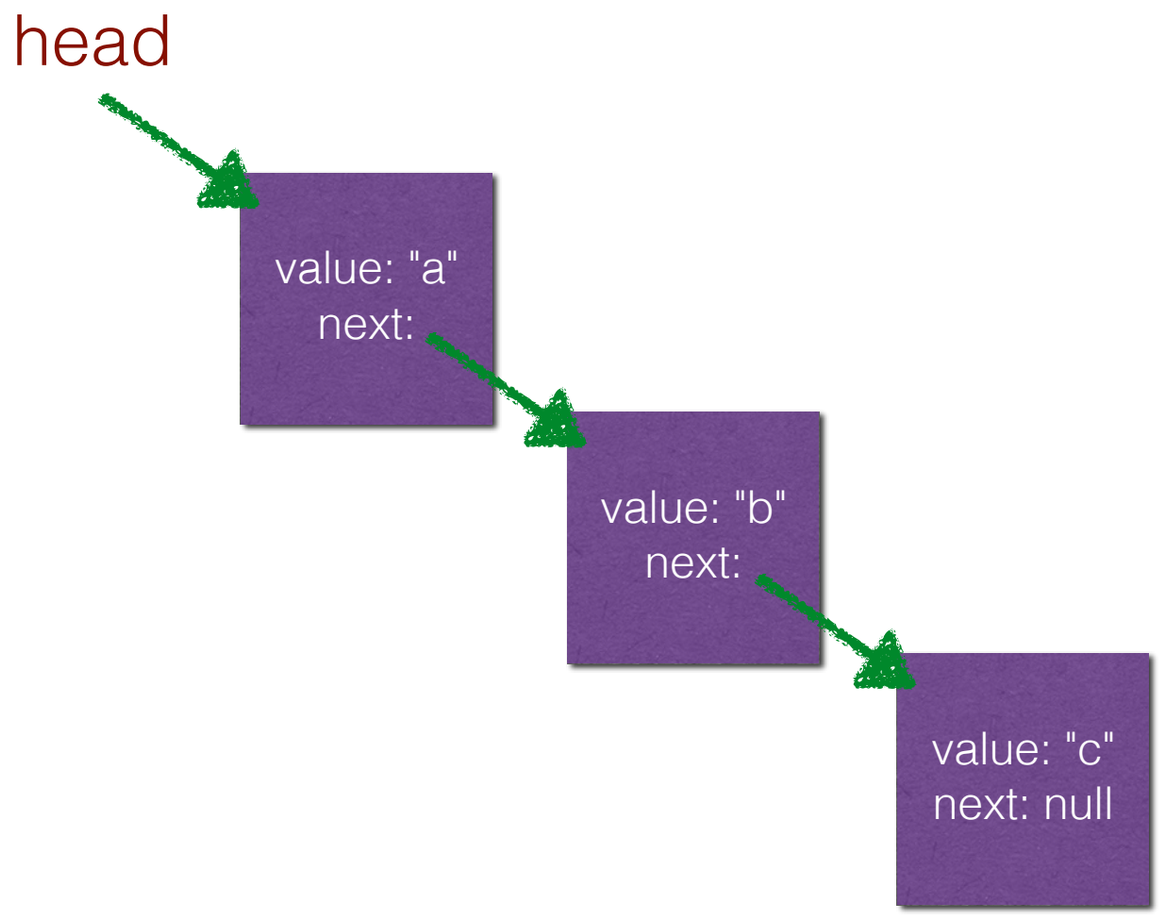
Linked List

data structure of ordered items,
each item pointing to the next in the series.

a linked list is a series of list-nodes (the purple boxes below). each node has a value (the item actually in the list) and next (the next node in the linked list).



head is a variable that points to the first list-node. the last node has null as it's "next" value.

```
var head = null;

var nodeA = {
  value: "a",
  next: null
};

var nodeB = {
  value: "b",
  next: null
};

var nodeC = {
  value: "c",
  next: null
};

head = nodeA;
nodeA.next = nodeB;
nodeB.next = nodeC;

console.log(head);
```

the items in this list are:
"a", "b" and "c"

the analogous array is:
["a", "b", "c"]

Try Me

iterate over a linked list. (assume the structure of the previous slide). print out the value of each item.

```
var currentNode = head;
while(currentNode){
  console.log(currentNode.value);
  currentNode = currentNode.next;
}
```

Try Me

add "d" to the end of the list.

```
var current = head;
while(current && current.next){
   current = current.next;
}
current.next = {
   value: "d",
   next: null
}

console.log(head);
```

Try Me

write a function that takes an array of items and returns the first node in a linked list composed of those same items.

```
function createList(items) {
  if(items.length < 1) return null;

  var head = {
    value: items[0],
    next: null
  };

  var current = head;
  for(var i = 1; i < items.length; i++) {
    current.next = {
      value: items[i],
    };
    current = current.next;
  }
  current.next = null;

  return head;
}
```

constructor time

```
function LinkedList(items) {

  function ListNode(item, nextNode){
    this.value = item;
    this.next = nextNode;
  }

  var length = items.length;
  this.head = length ? new ListNode(items[length-1], null) : null;

  for(var i = items.length - 2; i >= 0; i--) {
    head = new ListNode(items[i], head);
  }

}

LinkedList.prototype.at = function (index) {
  var current = this.head;
  for(var i = 0; i<index && current; i++){
    current = current.next;
  }
  return current && current.value;
}
```

Try Me

implement append for the LinkedList

```
LinkedList.prototype.append = function (item) {
  // add item to the end of the list
}
```

Try Me

implement prepend for a linked list

```
LinkedList.prototype.prepend = function (item) {
  // add item to the beginning of the list
}
```

Try Me

implement includes for a linked list

```
LinkedList.prototype.includes = function (item) {
  // return true if the list includes the given item
}
```

Try Me

implement forEach for LinkedList

```
LinkedList.prototype.forEach = function (callback) {
  // invoke callback on each item in the list
}
```

Try Me

implement removeAt for a linked list

```
LinkedList.prototype.removeAt = function (index) {
  // remove the item at index and return it
}
```

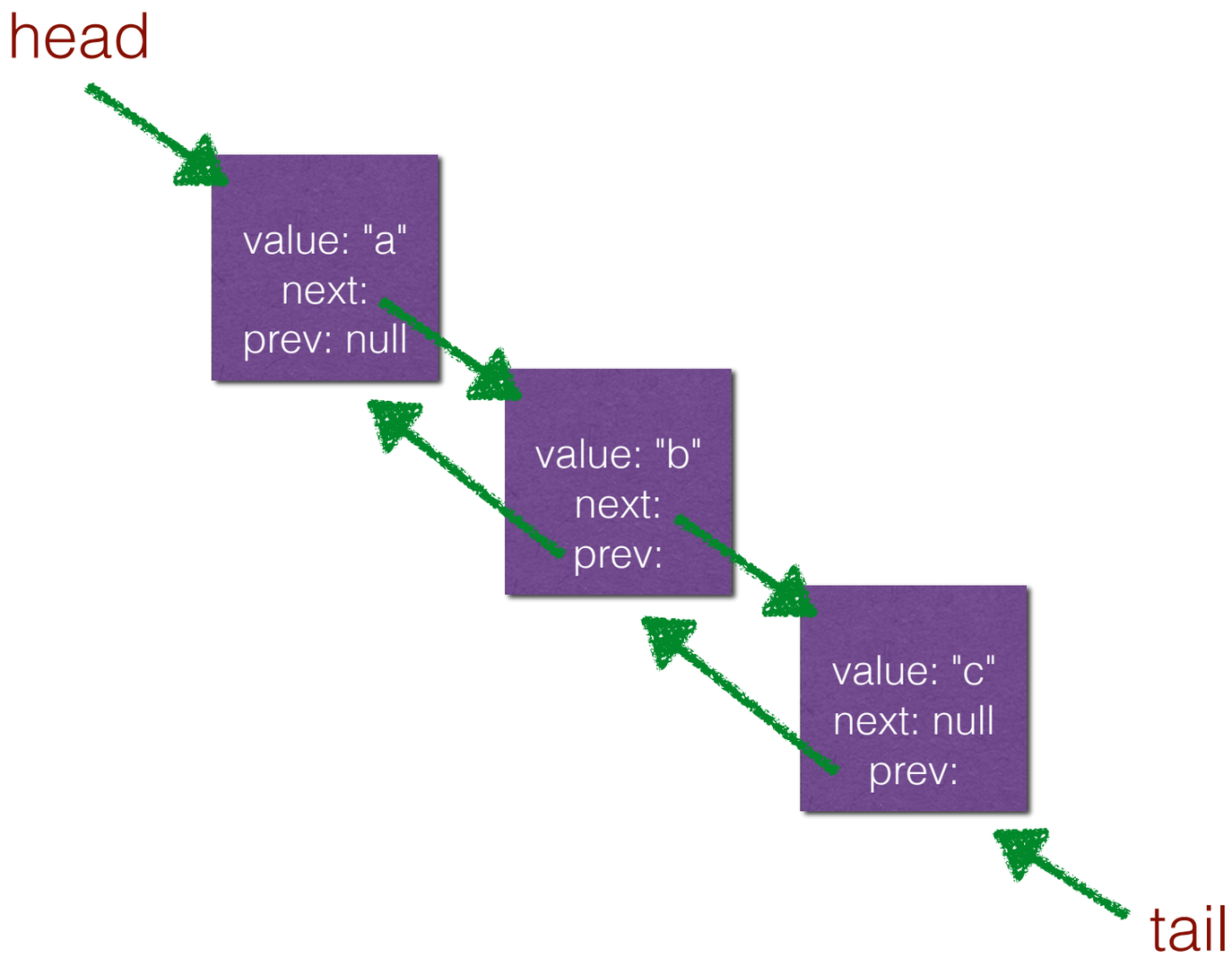
Try Me

use a linked list to implement a stack

```
function Stack () {
  var linkedList = new LinkedList();
  // implement push, pop, peek
}
```

Doubly Linked List

each node has a previous property as well.


head
value: "a"
next:
prev: null
value: "b"
next:
prev:
value: "c"
next: null
prev:
tail

Try Me

write a constructor for a doubly linked list.