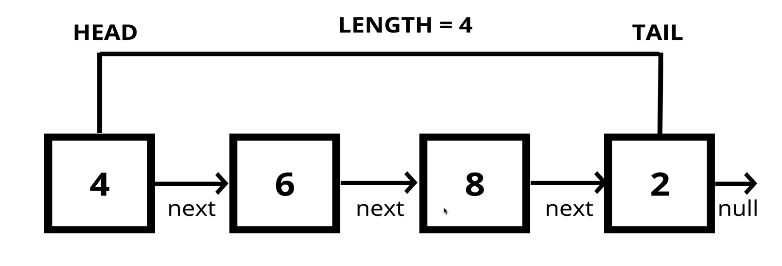
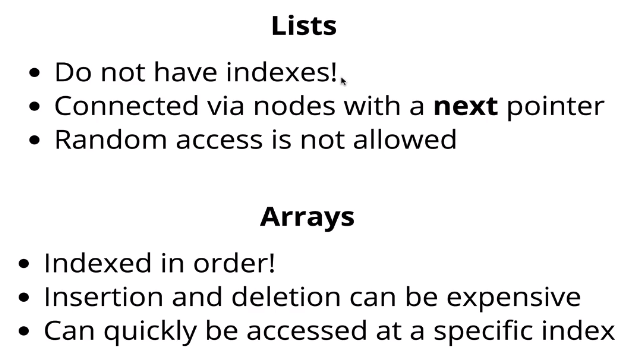
JavaScript Algorithms and Data Structures Masterclass

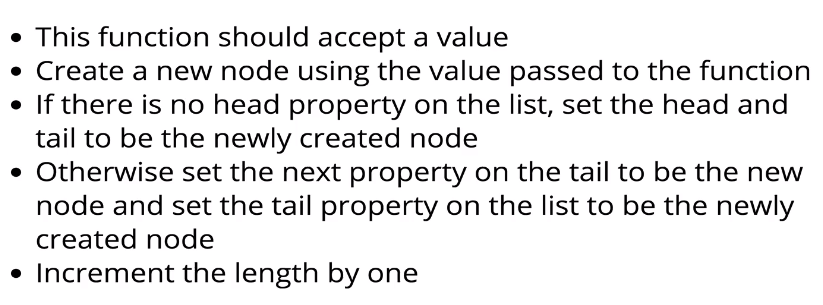
# Section 19: Singly Linked Lists

## What is a Linked List?

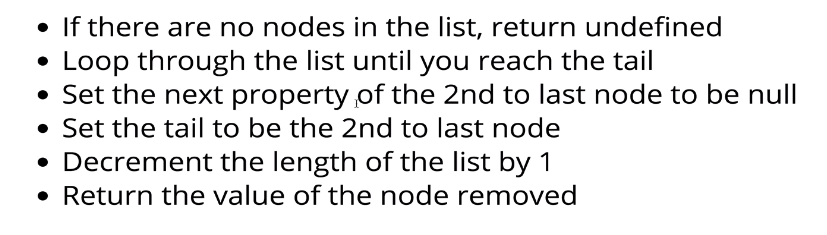
* DS that contains a head, tail, and length property
  + It consists of ‘nodes’
    - Each node has a value and a pointer to the next node
* **Diagram**:
  + 
  + Nodes are only connected to one node in 1 direction
* **Arrays vs. Linked Lists**
  + 

## Methods!

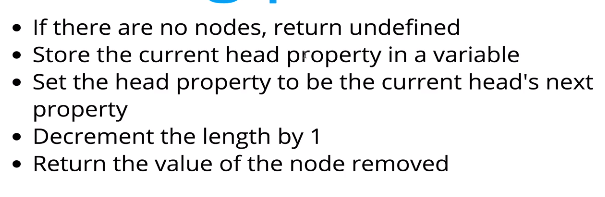
### Pushing

* Adding a new node at the **end** of the Linked List
  + Psuedo-code:
    - 

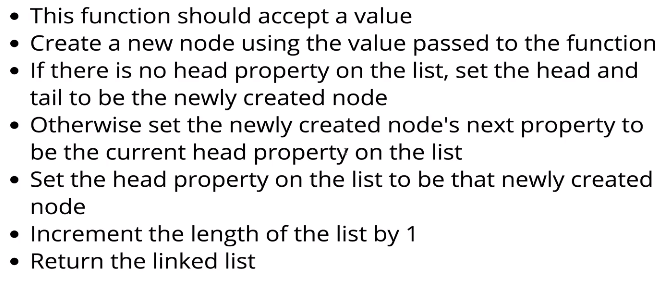
### Popping

* Remove a node from **end** of the Linked List
  + **Psuedo-code**:
    - 

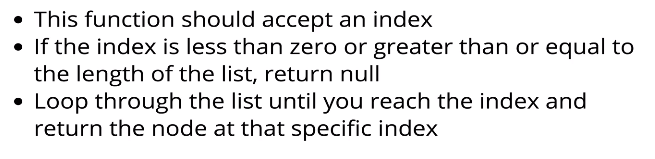
### Shift

* Remove a new node from the **beginning** of the Linked List
  + **Psuedo-code**:
    - 

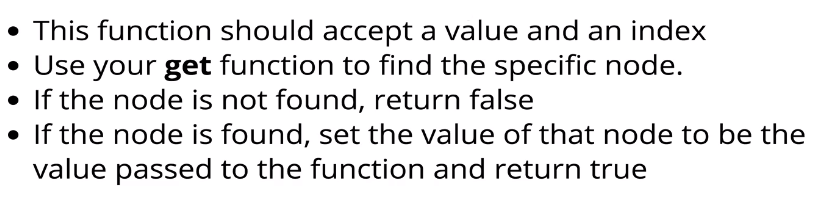
### Unshift()

* Add a new node to the beginning of the Linked List
  + **Psuedo-code**:
    - 

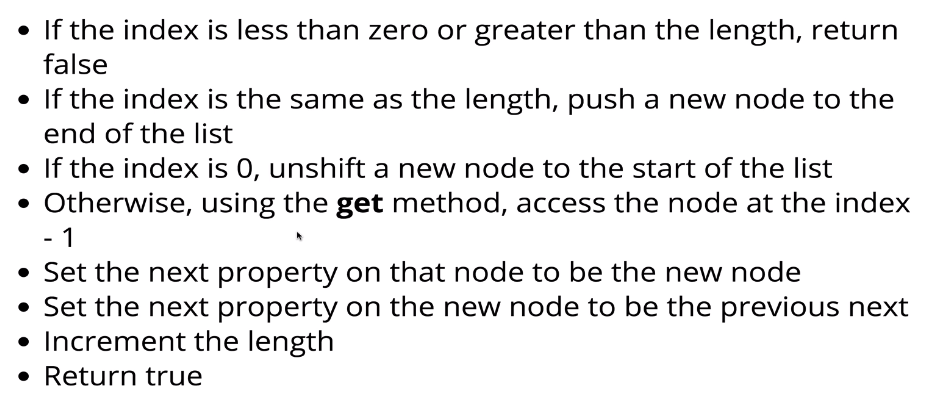
### Get()

* Retrieve a node by its position in the Linked List
  + **Psuedo-code**:
    - 

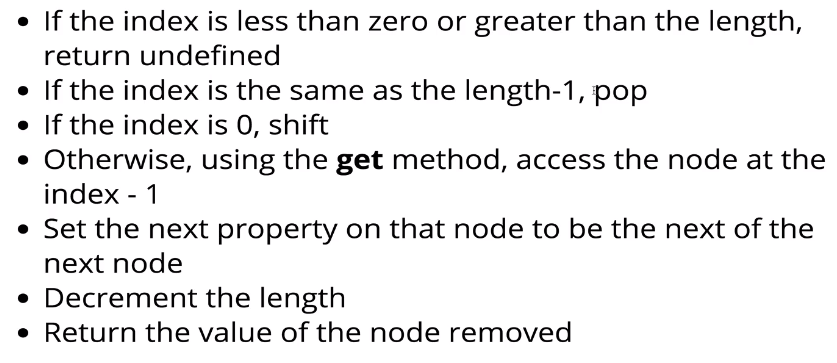
### Set()

* Changing/updating the **value** of a node based on its position in the Linked List
  + **Psuedo-code**:
    - 

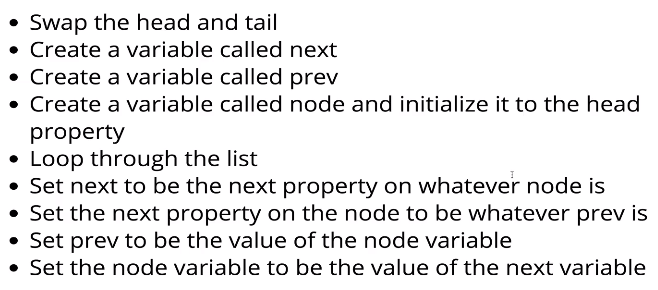
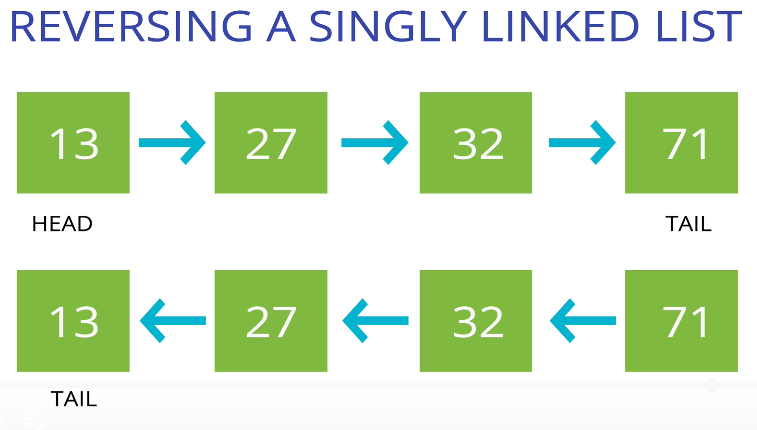
### Insert()

* Add a new node to the Linked List at a **specific** position (**ex**. Given index2, you must plug in the new node as the .next of the node before)
  + **Psuedo-code**:
    - 

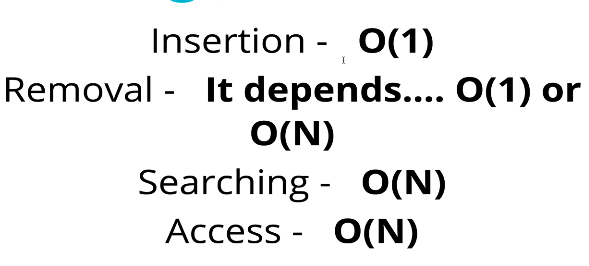
### Remove()

* Delete a node at a **specific** index
  + Psuedo-code:
    - 

### Reverse()

* Reverse the Linked List **in place** (as you traverse the SLL, you are reversing)
  + **Psuedo-code**:
    - 
  + **Diagram**:
    - 

### Big O for Singly Linked Lists

* 
  + Removal depends on where you remove:
    - Head = O(1)
    - Tail = O(n)
* 