# Data Structure – Stacks

Stacks are similar to arrays but it only adds and removes its elements in a LIFO way.

A stack is a data structure based on the principle Last In First Out. stack is container to hold nodes and has two operations — push and pop. The push operation is to add nodes into the stack and pop operation is to delete nodes from the stack and returns the top most node.

## Why use stack instead of arrays?

* Most methods on the Array Prototype hava a time complexity of O(n), whereas for Stacks they are mostly O(1).
* Arrays uses more space.

## Code Implementation

// Creates a stack

var Stack= function() {

this.count = 0;

this.storage = {};

}

// Adds a value onto the end of the stack

Stack.prototype.push = function(value) {

this.storage[this.count] = value;

this.count++;

}

//Removes and returns a value from the end of the stack

Stack.prototype.pull = function(value) {

if (this.count === 0) {

returns undefined;

}

var result = this.stroage[this.count];

delete this.storage[this.count];

this.count--;

return result;

}

// Returns the length of the stack

Stack.prototype.size = function() {

return this.size;

}

# Data Structures – Queue

## What is a queue?

A queue is like a line at a restaurant. Data that is entered first is removed first.

// Creates the queue

var Queue = function() {

this.storage = {};

//Keep track of the order of the new elements added in

this.count = 0;

// The index of the element entered first

this.lowestCount = 0;

}

Queue.prototype.enqueue = function(value) {

this.storage[this.count] = value;

this.count++;

}

// Removes a value from the beginning of the chain

Queue.prototype.dequeue = function () {

// Checks to see if queue is empty

if (this.count – this.lowestCount === 0) {

return undefined;

}

var result = this.storage[this.lowestCount]

delete this.storage[this.lowestCount];

this.lowestCount -- ;

return result;

}

// Returns the length of the queue

Queue.prototype.size = function() {

return this.count – this.lowestCount;

}