Some languages will have Data Structures built into the language itself, such as Java, and C++. However, these Data Structures can be implemented in to almost any language, such as JavaScript.

All Data Structures will have the following operations:

* Insertion
* Deletion
* Access
* Searching
* Traversal
* Sorting

**Arrays**

* Data is stored sequentially

Dynamic Arrays vs Static Arrays

Static arrays have a set size, whereas dynamic arrays have no fixed size, and will double in size once the size has been filled.

**Stack Data Structure**

* Last in first out data structure (LIFO). The last element inserted into the structure is the first to be removed.
* Vertical tower structure

**DS Operations**

* push() : add an element to the top of the stack
* pop() : remove an element from the top of the stack and return it
* isEmpty() : return Boolean value if stack is indeed empty
* isFull() : return Boolean value if the stack is indeed full
* peek() : return the value of the element on the top of the stack without removing it

**O Analysis**

* the push and pop operations take constant time O(1)

**Real Life Implementations**

* Reverse a word (put all the letters from the string into a stack, and pop them out.)
* In compilers
* Browser Back Button

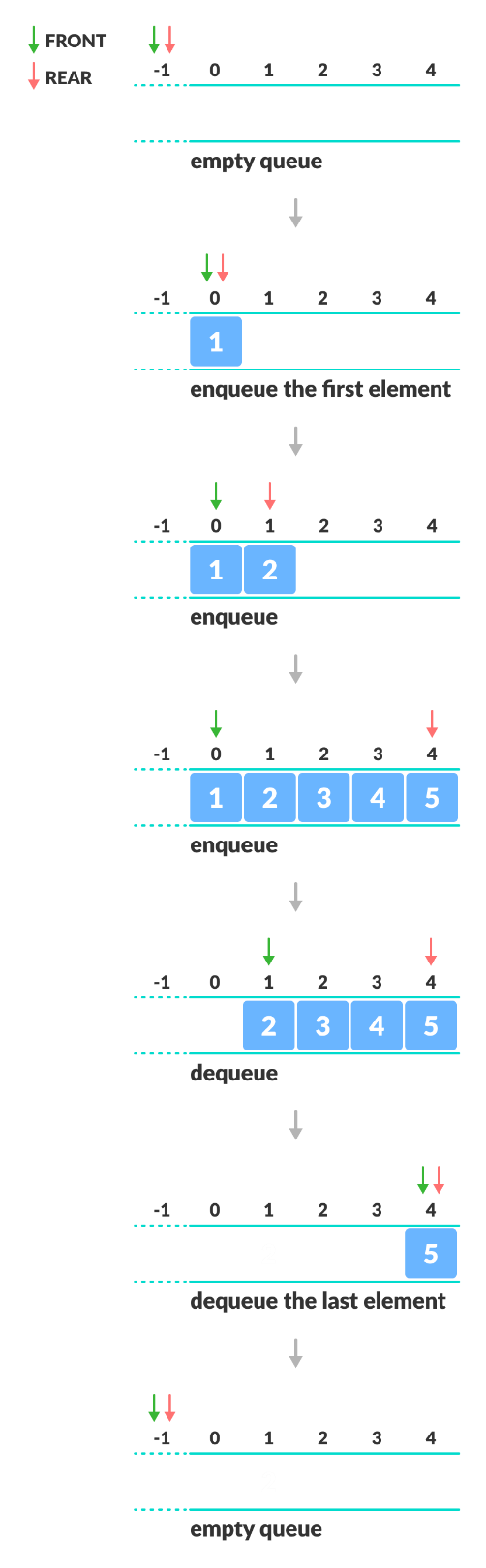
**Queue Data Structure**

* First in First out (FIFO)
* A collection designed for holding elements prior to processing
* Linear Data Structure

Diagram

Description automatically generated

**DS Operations**

* enqueue() : add an item to the back of the queue
* dequeue() : remove an item from the queue
* peek() : gets the element at the front of the queue without removing it
* isFull() : check if the queue is full
* isEmpty() : check if the queue is empty

**Real Life Implementations**

* CPU Scheduling
* Handling of interrupts in real time systems
* Call Centre phone systems use Queues to hold people calling them in order

**Big O Analysis**

* Enqueue or dequeue operations use O(1)

**Priority Queue**

* Each element is associated with a priority value
* Elements are served based on their priority; higher priority is served first
* IF, elements have the same priority, they are served according to their order in the queue

Priority Queue vs Normal Queue

In a queue, the **first-in-first-out rule** is implemented whereas, in a priority queue, the values are removed **on the basis of priority**. The element with the highest priority is removed first.