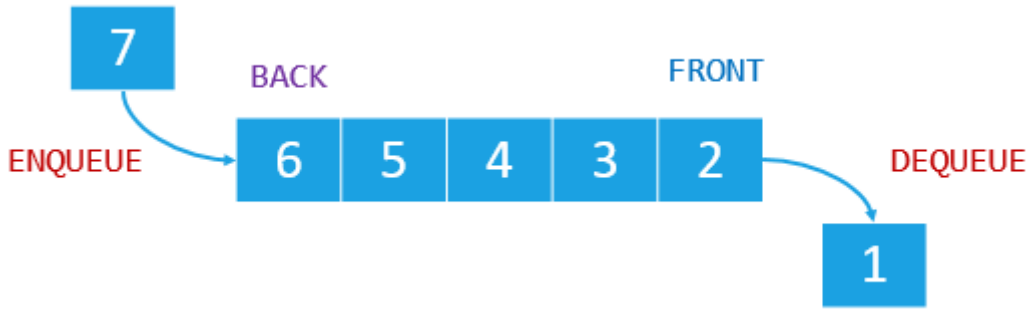
Queue Data Sctructure in JavaScript 24th Day

A queue is an ordered list of elements where an element is inserted at the end of the queue and is removed from the front of the queue.

Unlike a stack, which works based on the last–in, first–out (LIFO) principle, a queue works based on the first–in, first–out (FIFO) principle.

A queue has two main operations involving inserting a new element and removing an existing element.

The insertion operation is called enqueue, and the removal operation is dequeue. The enqueue operation inserts an element at the end of the queue, whereas the dequeue operation removes an element from the front of a queue.



The above diagram illustrate a queue.

Another important operation of a queue is getting the element at the front called peek. Different from the dequeue operation, the peek operation just returns the element at the front without modifying the queue.

The name queue comes from the analogy to a queue of customers at a bank. The customer who comes first will be served first, and the one who comes later is queued at the end of the queue and will be served later.

Applications of the Queue.

a). Serving request on a single shared resource, eg – PointerEvent, CPU task scheduling, etc.

b). In real life, call center phone system(people should wait in and hold until the service representative is free).

c). Handling of interrupts in real–time systems.

Implementing a JavaScript queue using an array.

You can use an array as a queue by using two methods of the Array type:

– Add an element at the end of the array using the push()method. This method is equivalent to the enqueue operation.

– Remove an element from the front of an array using the shift() method. It is the same as the dequeue operation.

Follow the following steps to implement a queue:

```
function Queue() {
this.elements = [];
 }
```

– The above snippets are the constructor of the queue, The `Queue()` constructor function uses an array to store its elements.

The enqueue() method adds an element at the end of the queue. We use the push() method of the array object to insert the new element at the end of the queue.

```
Queue.prototype.enqueue = function (e) {
this.elements.push(e);
 };
```

The dequeue() method removes an element from the front of the queue. In the dequeue() method, we use the shift() method of the array to remove an element at the front of the queue.

```
// remove an element from the front of the queue
 Queue.prototype.dequeue = function () {
return this.elements.shift();
};
```

The isEmpty() method checks if a queue is empty by checking if the length property of the array is zero.

```
// check if the queue is empty
Queue.prototype.isEmpty = function () {
return this.elements.length == 0;
};
```

The peek() method accesses the element at the front of the queue without modifying it.

```
// get the element at the front of the queue
Queue.prototype.peek = function () {
 return !this.isEmpty() ? this.elements[0] : undefined;
};
```

To query the length of a queue, we develop the length() method:

```
Queue.prototype.length = function() {
 return this.elements.length;
 }
```

To create a new queue from the Queue() constructor function, you use the new keyword as follows:

```
let q = new Queue();
```

Enqueue – the follow example will enqueue numbers from 1 to 7.

```
for (let i = 1; i <= 7; i++) {
 q.enqueue(i);
}
```

peek() to get the number infront of a queue the peek method is used:

```
// get the current item at the front of the queue
console.log(q.peek()); // 1
```

As we said earlier also, to get the length of our queue we use length()

```
// get the current length of queue
 console.log(q.length()); // 7
```

To remove the element at the front of the queue, you use the dequeue() method as follows:

```
/ dequeue all elements
while (!q.isEmpty()) {
console.log(q.dequeue());
 }
```

What will be the output of the above code? rememner we added 1 –7.

I hope this give you a good understanding of queue Data Structures In javaScript.

Practice questions.

a). Define the queue data structure.

b). List some applications of queue data structure.

c). What are the drawbacks of array implementation of Queue?

D). What are the scenarios in which an element can be inserted into the circular queue?

e). What is a dequeue?

f). What is the minimum number of queues that can be used to implement a priority queue?

g). Implement stack using a queue

h). Reverse first k elements of a queue

I). Generate binary numbers from to n using a queue

Best wishes,

Lux Tech Academy.

training@luxtechacademy.com