Queues (Implementation)

create a queue, add and delete items, and print the queue (Reading time: 3 minutes)

The queue will be an array, just like we did with the stack.

```
class Queue {
  constructor() {
    this.queue = [];
  }
}
```

Again, we initialize an empty array as our queue. To enqueue a node, we use the same built-in method that JavaScript provides.

```
  enqueue(data) {
    this.queue.push(data);
  }
}

//create an object of type queue
let myQueue = new Queue();

//insert items into the queue
myQueue.enqueue(2);
myQueue.enqueue(3);
myQueue.enqueue(9);
myQueue.enqueue(1);

//print the queue (I'll discuss this function in detail later in this lesson)
console.log("Your queue is:\n"+myQueue.printQueue());
```

To dequeue, you use shift instead of pop! This returns the first index of the array, which means that the first value added to the array will also be removed first. The first-in-first-out principle!

```
  dequeue() { //does not need an argumen because it deletes the elements that was inserted fi
    return this.queue.shift();
  }
```

```
}

//create an object of type queue
let myQueue = new Queue();

//insert items into the queue
myQueue.enqueue(2);
myQueue.enqueue(3);
myQueue.enqueue(9);
myQueue.enqueue(1);
//print the queue (I'll discuss this function in detail later in this lesson)
console.log("Your queue is:\n"+myQueue.printQueue());

myQueue.dequeue();
console.log("Your queue after deletion is:\n"+myQueue.printQueue());
```

Another important queue function is print. You can view the code here:

```
printQueue()
{
    var str = "";
    for (var i = 0; i < this.queue.length; i++)
        str += this.queue[i] + " ";
    return str;
}
```

In the next lesson, I will talk about the time complexity of various queue functions.