The stack is a data structure that follows Last In First Out (LIFO) principle. The element that is added at last is accessed at first. This is like stacking your books on top of each other. The book that you put at last comes first.

Implementation of Stack in Javascript

```
class Stack {
    constructor() {
        this.items = [];
    }

    // add element to the stack
    add(element) {
        return this.items.push(element);
    }

    // remove element from the stack
    remove() {
        if(this.items.length > 0) {
            return this.items.pop();
        }
    }

    // view the last element
    peek() {
        return this.items[this.items.length - 1];
    }

    // check if the stack is empty
    isEmpty(){
       return this.items.length == 0;
    }

    // the size of the stack
    size(){
        return this.items.length;
    }

    // empty the stack
    clear(){
        this.items = [];
    }
}

let stack = new Stack();
stack.add(1);
stack.add(2);
stack.add(4);
stack.add(8);
console.log(stack.items);
```

```
stack.remove();
console.log(stack.items);

console.log(stack.peek());

console.log(stack.isEmpty());

console.log(stack.size());

stack.clear();
console.log(stack.items);
```

Output

```
Output

[1, 2, 4, 8]
[1, 2, 4]
4
false
3
[]
```

Run the above code in a javascript compiler & play around with the push and pop functions.

In the above program, the Stack class is created to implement the stack data structure. The class methods like add(), remove(), peek(), isEmpty(), size(), clear() are implemented.

An object stack is created using a new operator and various methods are accessed through the object.

Here, initially this.items is an empty array. The push() method adds an element to this.items. The pop() method removes the last element from this.items. The length property gives the length of this.items.