Stacks (Implementation)

create a stack, add and delete data, and print the stack (Reading time: under 2 minutes)

In a stack, we should at least be able to **push** and **pop** values, or **nodes**. But first, we need to create the stack.

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

The initial value of the stack is an empty array: there are no nodes in the stack unless we push them!

To push a node, we can use the built-in push method that JavaScript provides.

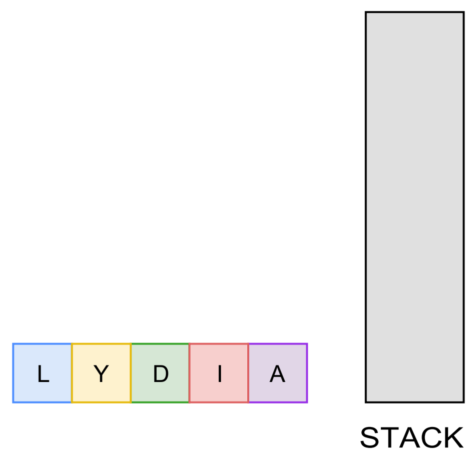
```
  push(data) {
    this.stack.push(data);
  }
}
let myStack = new Stack(); //create item of type stack
//push elements to a stack
myStack.push(2);
myStack.push(3);
myStack.push(9);
myStack.push(1);
//print stack
console.log("Your stack is:\n"+myStack.printStack()); //printStack discussed later in this le
```

The same goes for popping a node. However this time we don't need to provide any data, as the pop method always pops the last item in the list:

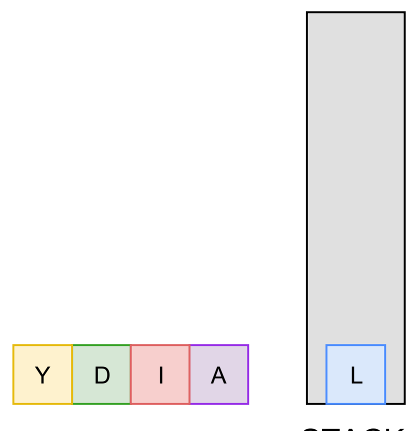
```
pop() { //pop takes no arguments and automatically removes the top elements
  return this.stack.pop();
}
}
```

```
let myStack = new Stack(); //create item of type stack
//push elements to a stack

myStack.push(2);
myStack.push(3);
myStack.push(9);
myStack.push(1);

//print stack
console.log("Your stack is:\n"+myStack.printStack()); //printStack discussed later in this le

//pop items from the stack
myStack.pop();
//print stack
console.log("Your stack after popping is:\n"+myStack.printStack()); //printStack discussed la
```

These are the two most important methods of a stack, as we're able to add values and pop them off again. Let's say that we want to reverse the word "Lydia":

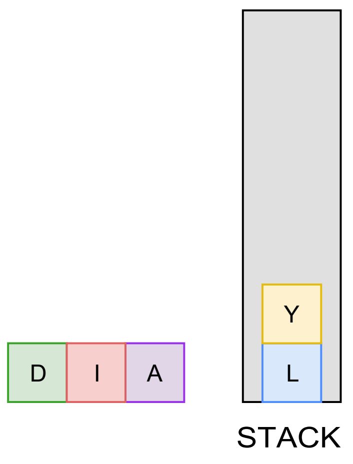


1 of 11



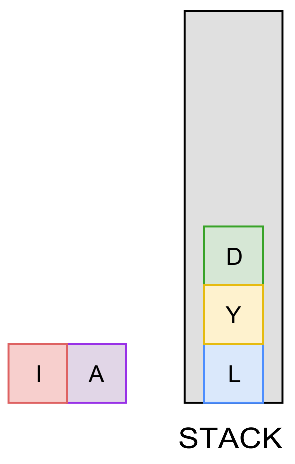
push

2 of 11



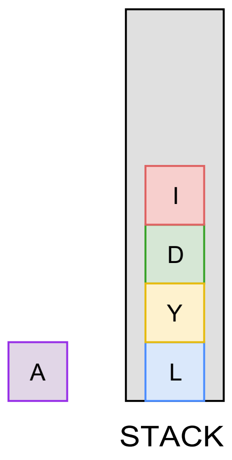
push

3 of 11



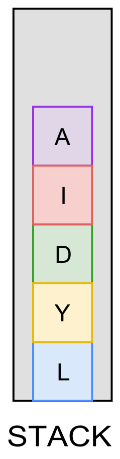
push

4 of 11



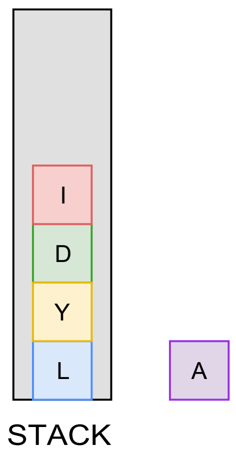
push

5 of 11



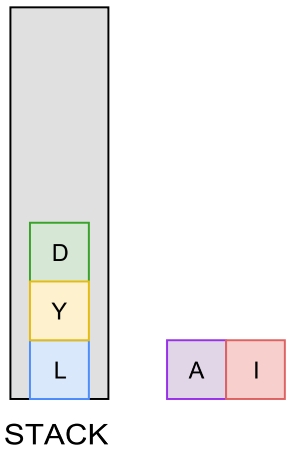
push

6 of 11



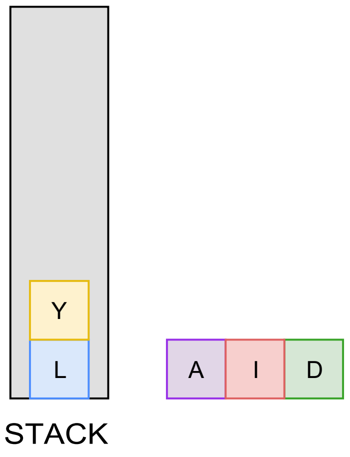
pop

7 of 11



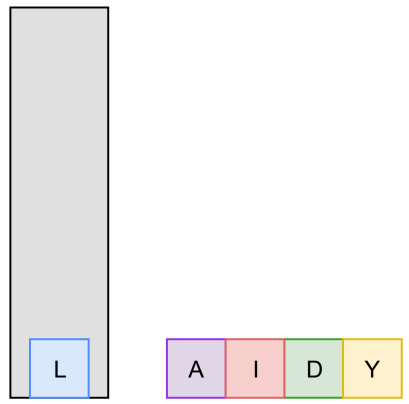
pop

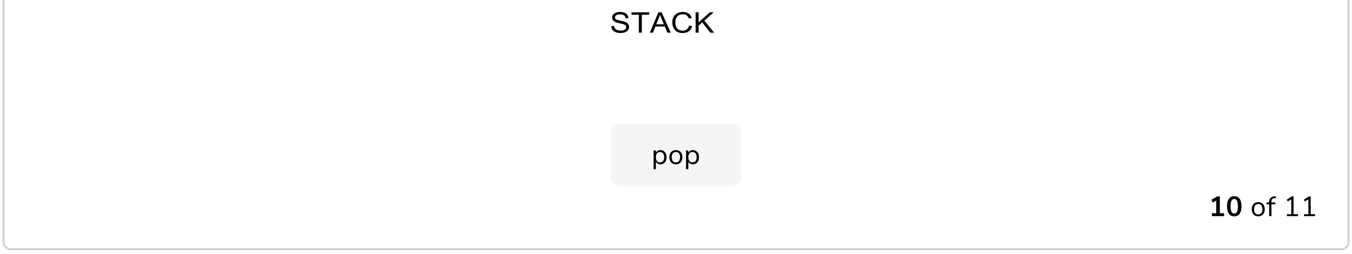
8 of 11

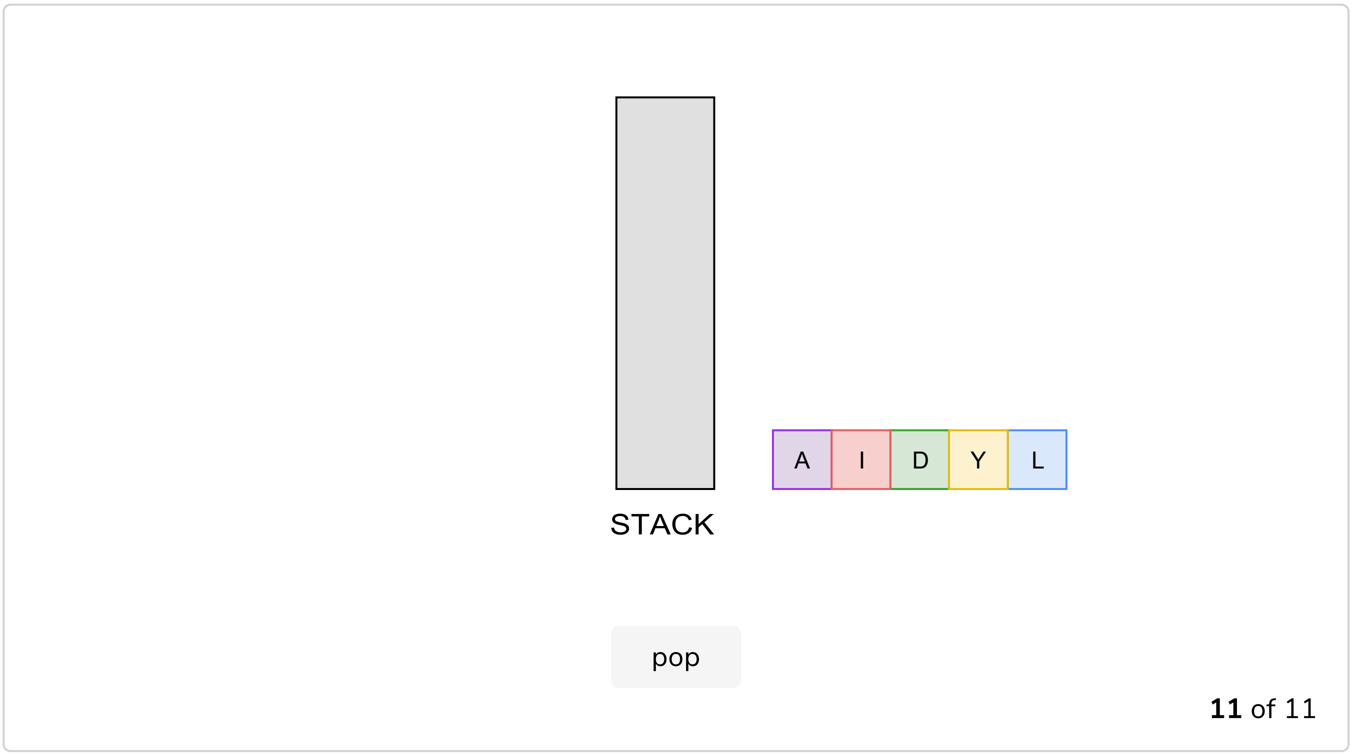


pop

9 of 11







In the above example, we first make an array of the string “Lydia”. The values get pushed to the stack, and the value on top of the stack gets popped off first. This results in the word reversed!

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

```
printStack()
  {
    var str = "";
    for (var i = 0; i < this.stack.length; i++)
        str += this.stack[i] + "\n";
    return str;
  }//print function ends here
}

let myStack = new Stack(); //create item of type stack
//push elements to a stack
myStack.push(2);
myStack.push(3);
myStack.push(9);
myStack.push(1);
//print stack
console.log("Your stack is:\n"+myStack.printStack());
```

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