Before proceeding, in your command-line interface (CLI), change to the downloaded folder called `palindrome` using `cd` plus space and then drag and drop the folder.

Open `palindrome.js`, and you will see four functions, of which one is incomplete. You will also see a variable called `message`, which is currently an empty string. Before moving on to the tasks *put in your message to post on Twitter*.

Your post will appear on the profile *@ps_or_cs*. There should be a link to this profile on learn.gold in Week 2.

The first function `putIntoStack` will store the characters in a `string` argument in a stack, which is implemented with a function in the following way:

```javascript
var stack = {
    arr:[],

    push:function(o){
       this.arr.push(o);
    },

    peek:function() {
       return this.arr[this.arr.length-1];
    },

    pop:function(){
       if (this.arr.length == 0) {
         return "Stack underflow!";
       } else {
         return this.arr.pop();
       }
    }

    isEmpty:function(){
       return this.arr.length == 0;
    }
  };
```
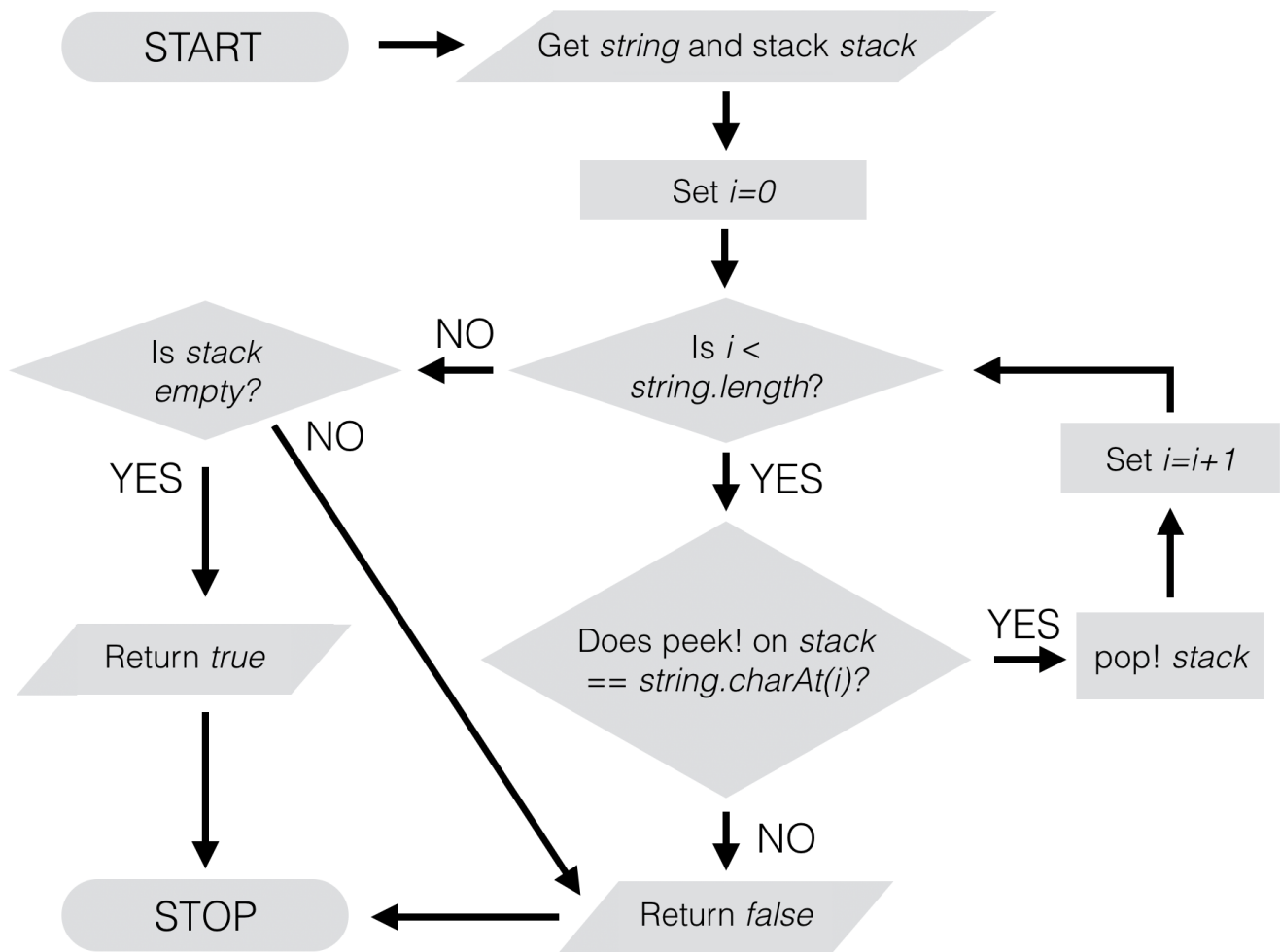
Here you can see the data in the stack is stored in a JavaScript array, and the methods can alter this array using the operations allowed by a stack.

Your Task

Now go to the function `compareStack`. Do not delete the lines of code already inside the function. Your task is to complete the function `compareStack`, which will implement the following algorithm:

Flowchart

START → Get *string* and stack *stack*

↓

Set *i=0*

↓

Is *i* < string.length?

— NO → **Is *stack empty?***
- YES → Return *true* → **STOP**
- NO → Return *false* → **STOP**

— YES ↓

Does peek! on *stack* == string.charAt(i)?

- YES → pop! *stack* → Set *i=i+1* → (back to Is *i* < string.length?)
- NO ↓ Return *false* → **STOP**

When you think you have completed both tasks, test them with:

```
npm test
```

You will find out whether you have posted to Twitter or if there is a problem.