

Learn how a Stack Works

You are probably familiar with stack of books on your table. You have likely used the undo feature of a text editor. You are also probably used to hitting the back button on your phone to go back to the previous view in your app.

You know what they all have in common? They all store the data in a way so that you can traverse backwards.





The topmost book in the stack was the one that was put there last. If you remove that book from your stack's top, you would expose the book that was put there before the last book and so on.

If you think about it, in all the above examples, you are getting *Last-In-First-Out* type of service. We will try to mimic this with our code.

This data storage scheme is called a *Stack*. In particular, we would have to implement the `push()` method that pushes JavaScript objects at the top of the stack; and `pop()` method, that removes the JavaScript object that's at the top of the stack at the current moment.

Here we have a stack of homework assignments represented as an array: `"BIO12"` is at the base, and `"PSY44"` is at the top of the stack.

Modify the given array and treat it like a `stack` using the JavaScript methods mentioned above. Remove the top element `"PSY44"` from the stack. Then add `"CS50"` to be the new top element of the stack.

	<code>homeworkStack</code> should only contain 4 elements.
	The last element in <code>homeworkStack</code> should be "CS50".
	<code>homeworkStack</code> should not contain "PSY44".
	The initial declaration of the <code>homeworkStack</code> should not be changed.