QUIZ JAVASCRIPT

A stack is a useful data structure to maintain the arrival of guests at a party. Complete this script to add the guest names to a `Stack` instance and print out the guest who arrived last (without removing it). Use the basic methods of the `Stack` class.

```
const Stack = require('./Stack.js');
const names = ['Harry', 'Lisa', 'Miles','James', 'Eve' ];

const namestack = new Stack( names.length );

for (let i = 0; i < names.length; i++) {

  namestack. push (names[i]);

}

console.log('Who arrived last? Answer: ', namestack. peek ())
```

You got it!

How do we add and remove items from a `Stack` data structure?

We add an item anywere to a `Stack` with an `insert` operation based on its location inside the stack. We remove an item anywhere from a `Stack` with a `remove` operation based on its location.

We add an item to a `Stack` by an `insert` operation replacing the current first item of the stack. We remove an item from the `Stack` by a `remove` operation which removes the first item off the stack.

We add an item to a `Stack` with an `append` operation to the end of the stack. We remove an item from a `Stack` with a `pop` operation from the top of the stack.

We add an item to a `Stack` by the `push` operation which only adds to the top of the stack. We remove an item from a `Stack` by the `pop` operation which only removes the top item.

Correct!! These are the only two operations supported by the `Stack` to add and remove an item from it.

The Javascript `Stack` class that we implement needs to support three operations. Which of these is NOT one of the three?

Get Bottom

👏 Correct! The `Stack` can only push, pop, and peek.

Peek

Pop

Push

What classes does the JavaScript `Stack` class use internally?

`LinkedList`, `Item`

`Array`, `Node`

`LinkedList`, `Node`

👏 Excellent! You've been working with these helper classes.

`LinkedList`, `Array`

The `Stack` data structure supports 3 basic operations: `push`, `pop`, and `peek`. In the following Javascript `.push()` implementation, fill in the code with the correct condition.

```
push(value) {
  if ( this.size < this.maxSize ) {
    this.stack.addToHead(value);
    this.size++;
  } else {
    throw new Error('Stack is full');
  }
}
```

👏 You got it!

The `Stack` data structure supports 3 basic operations: `push`, `pop`, and `peek`. In the following Javascript `.peek()` implementation, fill in the code with the correct condition.

```
peek() {
    if (   !this.isEmpty()   ) {
      return this.stack.head.data;
    } else {
      return null;
    }
  }
```

You got it!

The `Stack` data structure supports 3 basic operations: `push`, `pop`, and `peek`. In the following Javascript `.pop()` implementation, fill in the code with the correct condition.

```
pop() {
    if (   this.size !== 0   ) {
      const value = this.stack.removeHead();
      this.size--;
      return value;
    } else {
      throw new Error('Stack is empty');
    }
  }
```

You got it!