Call— Back in js—-------------------------------->>>>

// Example function that accepts a callback

function fetchData(callback) {

// Simulating an asynchronous operation (e.g., fetching data from a server)

setTimeout(function() {

const data = "Hello, World!";

// Call the callback function and pass the data as an argument

callback(data);

}, 2000); // 2 seconds delay

}

// Define a callback function

function handleData(data) {

console.log("Received data:", data);

}

// Call the fetchData function and pass the handleData callback

fetchData(handleData);

In the above code, we have a function called fetchData that simulates an asynchronous operation. It accepts a callback function as an argument. Inside the fetchData function, we use setTimeout to simulate the delay and then call the callback function with the fetched data.

Next, we define a callback function called handleData that receives the data as an argument and logs it to the console.

Finally, we call the fetchData function and pass the handleData function as a callback. When the asynchronous operation completes, the handleData callback is invoked with the fetched data.

This is a basic example of how callbacks work in JavaScript. Callbacks are commonly used in asynchronous operations, such as handling API responses, event handling, and more.

Promise–

⇒ Sorting in js

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.sort();

Apple,Banana,Mango,Orange

const points = [40, 100, 1, 5, 25, 10];

points.sort((a,b)=>{return a - b});

points.sort((a,b)=> a - b);// without return statement

console.log(points)

======================================================================================================================================

—----->>my\_string.charAt(4);

—------->this approach, we will use the [split()](https://www.geeksforgeeks.org/javascript-string-prototype-split-function/) method in order to convert the string into an array first.

* We will apply the sort() method on that converted array in order to sort the characters alphabetically.
* After sorting the characters alphabetically, we will convert our array back into the string itself using the method called [join()](https://www.geeksforgeeks.org/javascript-array-join-method/).

let sortString = (stringg) => {

return stringg.split("").sort().join("");

};

console.log("Sorted String: ");

console.log(sortString("qwertyuiop"));

============================================================

—---->Object.keys()

* Object.values()
* Object.entries()

⇒const population = {

male: 4,

female: 93,

others: 10

};

let genders = Object.keys(population);

console.log(genders); // ["male","female","others"]

let genders = Object.keys(population);

⇒function iterateObject() {

let exampleObj = {

book: "Sherlock Holmes",

author: "Arthur Conan Doyle",

genre: "Mystery"

};

Object.entries(exampleObj).map(entry => {

let key = entry[0];

let value = entry[1];

console.log(key, value);

});

}

iterateObject();

genders.forEach((gender) => console.log(gender));

This will return:

"male"

"female"

"Others"

========================================================================

========================================================================

const array = [1, 2, 3, 4, 5];

for (const value of array) {

console.log(value);

}

Op = 1 2 3 4 5

const array = [1, 2, 3, 4, 5];

for (const index in array) {

console.log(index);

}

Op = 0 1 2 3 4

==================================================================

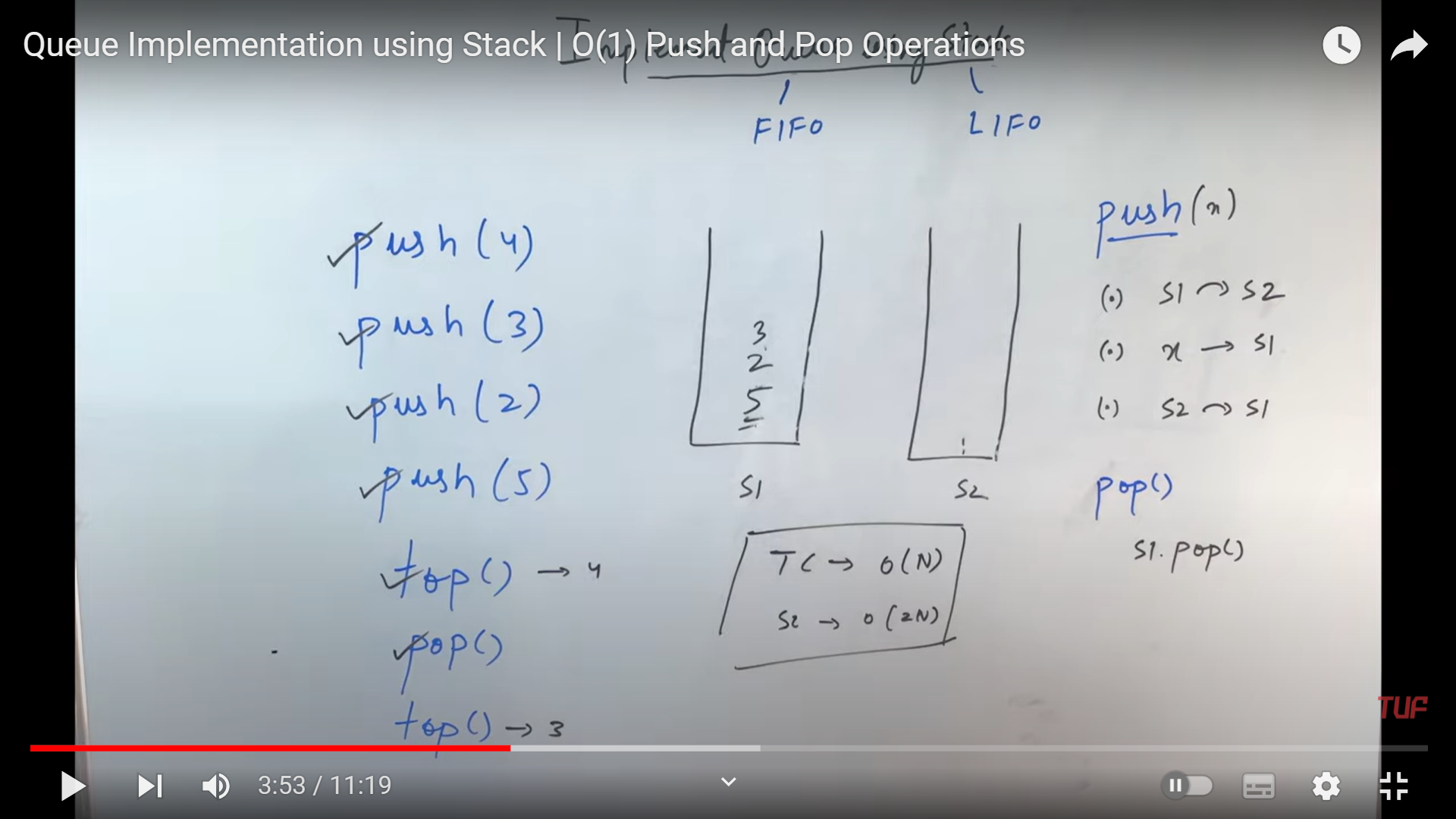
let str = "Hello, World!";

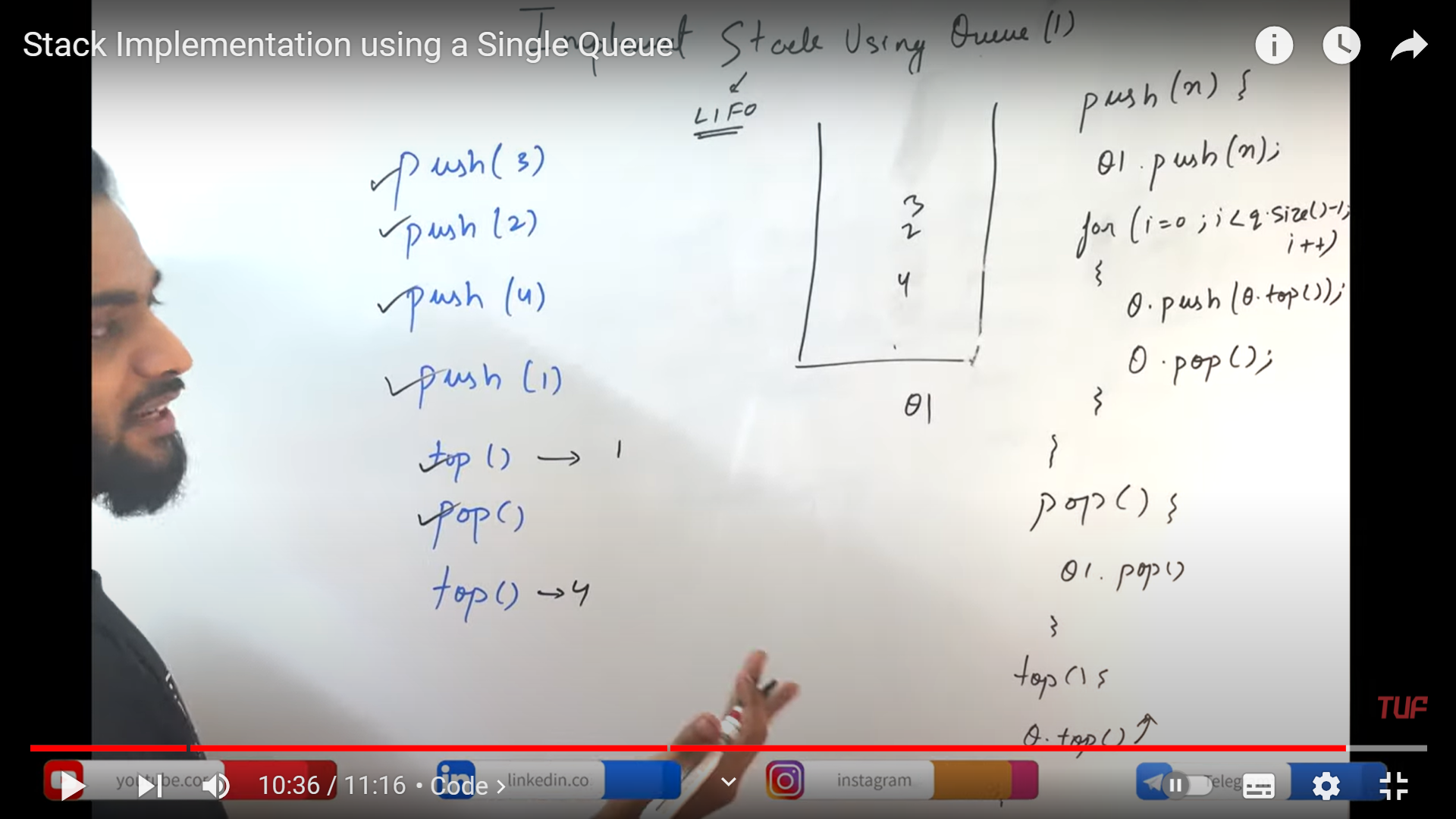
let lowercaseStr = str.toLowerCase(); // returns new string (immutable)

console.log(lowercaseStr);

—----------->return str.replace(/[^a-z]/g, '');

STACK—---------->





Stack(S for single) form queue.

================================================================================================================================================

// Create a stack object

const stack = [];

// Push elements onto the stack

stack.push(10);

stack.push(20);

stack.push(30);

// Peek the top element

console.log(stack[stack.length - 1]); // Output: 30

// Pop elements from the stack

console.log(stack.pop()); // Output: 30

console.log(stack.pop()); // Output: 20

// Check if the stack is empty

console.log(stack.length === 0); // Output: false

// Get the size of the stack

console.log(stack.length); // Output: 1

// Clear the stack

stack.length = 0

The code let c = []; in JavaScript initializes a variable c as an empty array. Arrays in JavaScript can be thought of as a data structure that has both stack-like and queue-like

Queue Operations:

* Enqueue (Queue Enqueue Operation): To add an element to the end of the array (similar to adding to the back of a queue), you can use the push method.

c.push(item);

* Dequeue (Queue Dequeue Operation): To remove and return the first element from the array (similar to removing from the front of a queue), you can use the shift method.

⇒let removedItem = c.shift();

⇒To get the front element in the queue use

obj[0];

========================================================================

Css flex is used to bring the two divs which are on different lines together and is used in the parent div only, is applicable to the first generation of child only, not on the 2nd or third generation of childs.

====================================================================

Fetch is a way to send a request from frontend to backend.