**CALL BACK HELL**

getData(function(x){

console.log(x);

getMoreData(x, function(y){

console.log(y);

getSomeMoreData(y, function(z){

console.log(z);

});

});

});

Ref: <https://blog.bitsrc.io/understanding-promises-in-javascript-c5248de9ff8f>

**WHAT IS CALL BACK FUNCTION:-**

A JavaScript callback is a function which is to be executed after another function has finished execution. A more formal definition would be - Any function that is passed as an argument to another function so that it can be executed in that other function is called as a callback function.

A callback is a function that isn’t immediately executed but is instead passed to another function as a parameter.

When the receiving function completes its task, it calls the callback function to continue the execution of the program1.

Essentially, callbacks allow you to handle asynchronous operations and control the flow of your code.

**REST PARAMETER:-**

We can use rest parameter in inside object also. We can do rest parameter to merge the objects.

Ref:- https://www.javascripttutorial.net/javascript-object-spread/

function show(...args) {

let sum = 0;

for (let i of args) {

sum += i; }

console.log("Sum = "+sum);

}

show(10, 20, 30);

**? After Variable:-**

In JavaScript, the ? after the type of a variable indicates that the variable is optional. This means that the variable can be assigned a value of the specified type, or it can be left undefined.

For example, the following code declares a variable called name that is optional:

JavaScript

name?: string;

This variable can be assigned a string value, such as "John Doe", or it can be left undefined.

**CORE MODULES IN NODE JS**

1. Event Emitter
2. Stream
3. FS
4. Net
5. Global Objects

**GLOBAL OBJECT:-**

1. Buffer
2. Console
3. Process
4. Global

micro task :-

--------------

1. take a long time.

2. Event loop gives higher priority to MicroTask Queue.

2. Eg Promises, Process.nextTick, I/O operation

Macro Task:

-----------------

1. setTimeout, setInterval,

For example the code shown below is

console.log("Start");

setTimeout(function() {

console.log("Timeout");

}, 0);

Promise.resolve().then(function() {

console.log("Promise"); // microTask!

});

console.log("End");

The out put is start --> End --> Promise --> Timeout

**queueMicrotask :-**

It convert the synchronous task into asynchronous

**What is the difference between promise and async await?**

The promise involves chaining . then and . catch methods, whereas Async Await uses a try-catch block that looks more like synchronous code.