**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.

**Truths Behind JWT token**

1. When you do log in, send 2 tokens (Access token, Refresh token) in response to the client.

2. The access token will have less expiry time and Refresh will have long expiry time.

3. The client (Front end) will store refresh token in an httponly cookie and access token in local storage.

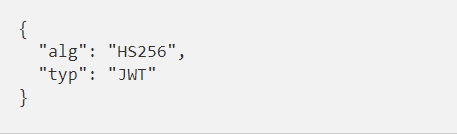
4. The client will use an access token for calling APIs. But when it expires, you call auth server API to get the new token (refresh token is automatically added to http request since it's stored in cookies).

5. Your auth server will have an API exposed which will accept refresh token and checks for its validity and return a new access token.

6. Once the refresh token is expired, the User will be logged out.

**A JWT token consists of 3 parts separated by a . being each one of them**:

**Header:** with the type (JWT) and type of coding



**Payload:** It is where the user’s information will be found that will allow the server to discern whether or not it can access the requested resource



**Signature:** The signature function will be applied to the other two token fields to obtain the check field

**Types of token**

There are many types of token, although in authentication with JWT the most typical are access token and refresh token.

**Access token**: It contains all the information the server needs to know if the user / device can access the resource you are requesting or not. They are usually expired tokens with a short validity period.

**Refresh token**: The refresh token is used to generate a new access token. Typically, if the access token has an expiration date, once it expires, the user would have to authenticate again to obtain an access token. With refresh token, this step can be skipped and with a request to the API get a new access token that allows the user to continue accessing the application resources. It is stored in browser cookie side.

Ref:- <https://stackoverflow.com/questions/27726066/jwt-refresh-token-flow>

https://www.geeksforgeeks.org/jwt-authentication-with-refresh-tokens/?ref=ml\_lbp