**Sync and Async in Js ?**

**Sync in js**

Synchronous means the code runs in a particular sequence of instructions given in the program. each instruction waits for previous instruction to complete its execution.

In other words

Meaning of synchronous programming is suppose you write a multiple line of instruction in your program in a sequence . in that case all lines of code run one by one and second lines of code waits for the previous line when previous line of code is executed now control of execution jump to the next line.

Note : JavaScript is also synchronous programming language or single threaded

**Async in js**

Due to synchronous programming sometimes important instruction get blocked due to some previous instruction , which causes a delay in the UI.

 Asynchronous programming allow us to execute next instruction immediately and dose not block the flow of code

Other Word

Suppose we write a 5 line of instruction in program. And unfortunately 3rd  line instruction take some time for execution. In that case  Asynchronous programming allow us to execute next instruction immediately so all line of instructions run one by one and also 3rd line of instruction execute parley . when it execute print whatever written in 3rd instruction.

setTimeout() execute a function according to user . means When the user want to execute that function.we have to pass the time inside the setTimeout() function

*<*script*>*

      console.log("Hello Anurag"); *//Line 1 execute*

      console.log("Hello Anurag"); *//Line 2 execute*

      setTimeout(()=>{   *// Line 3 takes some time for executing in that case*

*Execution of control goes to next line but line 3 execute parallel*

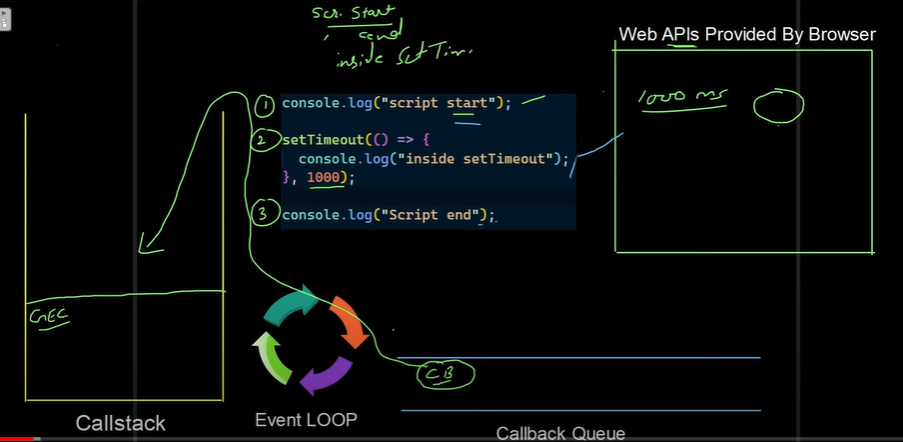
     console.log("Take time to execute the code");

        }, 4000)

      Console.log("Hello Anurag");*//Line 4 execute after 2*

     console.log("Hello Anurag");*// Line 5 execute*

*</*script*>*



**Callback**

A function which is passed inside the another function as an argument. that is called callback.

*<*script*>*

        functionSum(a,b){

            console.log(a+b)

        }

         functionCal(x,y,Sumcallback){

            Sumcallback(x,y)

         }

         Cal(10,20,Sum);

*</*script*>*

**Callback hell (it is problem in js .)**

Nested callbacks stacked below one another forming a pyramid structure

Nesting of callback . means it create a pyramid structure. That is very difficult to understand

// Suppose we create a function A1 and we pass another function A2 inside A1 as an argument and also pass another function A3 inside A2 as an argument again we pass another function A4 inside A3 as an argument. So basically

we are doing Callback nesting . we know that this type of problem is very difficult to understand so this type of problem is called Callback hell.

How callback hell generates:

Whenever we use callback like nesting and deeper nesting in that case callback hell problem may be generated .

*<*script*>*

    function getData(dataId, getNextData){

        setTimeout(()=>{

            console.log("data",dataId);

            if(getNextData){

                getNextData();

            }

        } , 2000);

    }

*// Callback hell*

  getData(1,()=>{

          console.log("Getting data 2...");

          getData(2, ()=>{

            console.log("Getting data 3...");

            getData(3, ()=>{

                console.log("Getting data 4...");

               getData(4);

            });

          });

    });

*</*script*>*

**Promises in JavaScript**

Callback hell is a problem in JavaScript so Solving for this type of problem we use promises in js.

Promises is becally spacell object in javacript which represent the eventual completion or falure of a asynchronous operation in JavaScript

It has three state

Starting is panding state and other two are resolve and reject based on the operation which is performed through async programming

Class(Use this class to create a new promise and inside the promise we pass a function with two handlers )

Let promise = new promise( (Resolve, reject) =>{

}

Function with two handlers(and these handlers is also a function or callback)

Promise is a object in JavaScript and it has three state.

1Pending state

2. Resolve state

3 Reject state

A JavaScript Promise object can be

1Pending state : The result is Undefined .

2. Resolve state : The result is value(Fulfilled)

3 Reject state : The result is an error object

**How to use Promise**

promises in js is for eventual completion of task(Means given task is perform eventually but we cant say that given task will be resolve or reject ) it is an object in js.

//for creating promise object we write

new Promise((res,reject) =>{

…………………..

……………………

})

We have two function then() and catch()of promise object. inside the function we can pass both the function if.

* If promise is full fill then we want to perform some task in that case we use .then() method
* If promise is Reject then we want to perform some task in that case we use .catch() method And both the function execute when our promise is full fill or reject.

We pass parameter inside then() and catch() method that is res and err it is use to access resolve or reject value // res= result

Promise.then((res) =>{ // res parameter is used to access resolve value

………………….

…………………

});

Promise.then((res) =>{ // err parameter is used to access reject value

………………….

…………………

});

Promise chaining is a technique used to handle multiple asynchronous operations in a sequence. When a promise is returned inside a .then() handler, it allows chaining additional .then() handlers to deal with subsequent asynchronous tasks.

 Function getData(dataId){

        Return newPromise((resolve,reject)=>{

            setTimeout(()=>{

            console.log("data",dataId);

            resolve("Success");

            } , 2000);

})

    }

*// Promise changing*

    getData(1)

    .then((res)=>{

        returngetData(2);

    })

    .then((res)=>{

        returngetData(3);

    })

    .then((res)=>{

       console.log("Success");

    })

*</*script*>*

*</*body*>*

*</*html*>*

**Async-Await in JavaScript**

**Async keyword**

It is a keyword in JavaScript and it is use to make a simple and Understandable Asynchronous programming (We can do that same thing with the help of promise changing and callback hell but it is best way )

* Async keyword is use with any function for making normal to Async function.
* Async function returns a promise

**Await keyword**

It use to  pauses the execution of its surrounding async function until the promise is settled.

(Suppose we call a function and we know this function return a promise . if i write Await in

front of that function in that case Await Wait for promises with pauses the execution of its surrounding async function)

* Await keyword is only use with Async function.

**Async-Await in JavaScript**

function getData(dataId){

    return new Promise((resolve,reject) =>{

        setTimeout(()=>{

              console.log(" Data",dataId);

              resolve("2000"); *// Susseccful in srrver*

        },3000)

    });

   }

   async function getAllData(){

    console.log("Getting data 1......");

    await getData(1);

    console.log("Getting data 2......");

    await getData(2);

    console.log("Getting data 3......");

    await getData(3);

    console.log("Getting data 4......");

    await getData(4);

   }

*</*script*>*