4promise.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>

    <script>
        //technical suneja
        //async js programming
        //callback promise async and awaits

        ////explanation this program jiski timing kar rahte hai wo hi pahle chalata hai sirf set time out ke sath
        // const datas=[
        //     {
        //         name:"Ajay", Profession:"software doveloper"
        //     },
        //     {
        //         name:"anuj", Profession:"software doveloper"
        //     }
        // ]
        //  function getDatas(){
        //     setTimeout(()=>{
        //         let output=" "
        //       datas.forEach((data,index)=>{
        // output+= `<li>${data.name}</li>`
        //       })
        //       document.body.innerHTML=output;;;;;;;;;
        //     },3000)
        //  }

        //  const createData=(newdata)=>{
        //     setTimeout(()=>{
        //         datas.push(newdata)
        //     },2000)
        //  }

        //  getDatas()
        // createData({name:"shubh",Profession:"web doveloper"})


        //explanation this program jiski timing kar rahte hai wo hi pahle chalata hai sirf set time out ke sath




        //calback
        // const datas=[
```

```
//     {
//         name:"Ajay", Profession:"software doveloper"
//     },
//     {
//         name:"anuj", Profession:"software doveloper"
//     }
// ]

// function getData(){
//  setTimeout(()=>{
//     output="";
//     datas.forEach((data,index)=>{
//         output+=`<li>${data.name}</li>`
//     })
//     document.body.innerHTML=output;
//  },1000)
// }

// function createData(newdata,calback){
//     setTimeout(()=>{
//         datas.push(newdata)
//         calback()
//     },3000)
// }
// // getData()
// createData({name:"shubh",Profession:"web doveloper"},getData)




//promise
// const datas=[
//     {
//         name:"Ajay", Profession:"software doveloper"
//     },
//     {
//         name:"anuj", Profession:"software doveloper"
//     }
// ]

// function getData(){
//  setTimeout(()=>{
//     output="";
//     datas.forEach((data,index)=>{
//         output+=`<li>${data.name}</li>`
//     })
//     document.body.innerHTML=output;
//  },1000)
// }


// function createData(newdata){
//     return new Promise((resolve,reject)=>{
// setInterval(()=>{
//     datas.push(newdata);
//     let error=true;
//     if(!error){
//         resolve()
```

```
        //     }else{
        //         reject("kuch sahi nahi hai....")
        //     }
        // },2000)
        //     })
        // }


        // createData({name:"shubh",Profession:"web doveloper"}).then(getData).catch(err=>console.log(err))







        //async and await



        //master async  sherians coding school

        // sync matlab ek ke bad dusra hoga jab tak ek command complete na ho, dusra nahi hoga

        //task a -5
        //task b -2
        //task c -15
        //task d -1
        //task e -4
        // total time 27 sec
        // console.log("hey1")
        // console.log("hey2")
        // console.log("hey3")


        // async matlab saare kaam ek sath shuru kardo jiska answer pahle aajaye uska jawaab dedena


        // kaise pata chalata hai ki hum sync code likh rahe ya async?

        // async ki pahchan
        // setInterval
        // setTimeout
        // Promise
        // fatch
        // axios
        // XMLHttpRequest







        // async js hai kya?
        // kai bar aapko final code depended hota hai kisi aur ke server par, is case main hame pata nahi hota ki ans uske server se kab laut kar aayega to hum kya nahi kr sakte is writing sync code, isse nipatane ke liye hum log async code likh deta taaki blocking na hoo and jab bhi answer aaye humara answer ke respect mai  chalne wala code chal jaye

        // async code ka main motive hota hai ki un case main jisme hume pata nahi code ka answer ke
```

```
respect main koi particular code chala dena

        // example set time out
        // callback function humesha async code mein asnwer ane par chalta hai
        // console.log("hey1")
        // setTimeout(function(){
        // console.log("hey2")
        // },2000)



        // js is not asynchronous
        // async ka matlab do kam ek sath karna

        // jo bhi main stack par hota hai and jo bhi side stack par hota hai wo behind the scenes prossing kar sakte hai  aur jab uski processing complete ho use main stack mai la kar chalaya ja sakta hai

        // main stack  execution stack

        // event loop importent for interview


        // console.log("hey1")
        // console.log("hey2")
        // setTimeout(function(){

        //     console.log("hey3")
        // },0)
        // console.log("hey4")


        // single threading and multi threading
        // js single threading hai

        // ye requist bhejete hai
        // callback
        // fetch
        // axios
        // setTimeOut
        // setInterval

        // jab complete ho toh iska ans yaha hai
        // promise
        //then catch
        // try and catch
        // async await



        // callback =>function
        // callback hamesha ek function hota ye sirf tab chalta hai jab async code ja complete hojataa hai

        // example
        // fetch('www.indiawalesir.com').then()
        // var ans= new Promise((res,rej)=>{
        //     if(true){
        //         return res()
```

```
        //     }
        //     else{
        //         return rej()
        //     }
        // })
        // ans.then(function(){
        // console.log("resolve ho gya")
        // }).catch(function(){
        //     console.log("reject ho gay")
        // })



        // example
        // user will ask  for a number between 0 se 9 and if the number is below 5 resolve if not reject

        // var ans=new Promise((res,rej)=>{
        //        var n=Math.floor(Math.random()*10);
        //        console.log(n)
        //        if(n<5){
        //         return res()
        //        }else{
        //         return rej()
        //        }
        // })




        // ans.then(function(){
        //     console.log("below")
        // }).catch(function(){
        //     console.log("abobe")
        // })


        // what is promise
        // a promise is an object that returns a value which you hope to receive in the future but not now

        // why we need promise
        // javascript always synchronous and single threaded language. it means javascript never waits for code or function result when they take some time js direct excute next code


        // example 1  problem
        // let data=1;
        // console.log(data);
        // data=2;
        // setTimeout(()=>{
        // console.log("timer"+data)  //output 3
        // },2000)

        // data=3;
        // console.log("last"+data)


        // promise example
```

```
278         // let data=new Promise((resolve,reject)=>{
279         //  setTimeout(()=>{
280         //     if(true){
281         
282         //         resolve("done has been executed")
283         //     }else{
284         
285         //         reject("code has been rejected")
286         //     }
287         //  },2000)
288         // })
289         // data.then((item)=>{
290         //     console.log(item)
291         // }).catch((err)=>{
292         //     console.log(err)
293         // })
294         
295         // fetch ke andar promise wala code apne aap likha hota hai
296         //
297         // let data=fetch("http://127.0.0.1:5500/data.json");
298         // data.then((item)=>{
299         //     return item.json() //json ko resolve karne ke liye
300         // }).then((result)=>{
301         //     console.log("2nd out",result)
302         // })
303         
304         
305         
306         // promise chaining
307         
308         // let data=new Promise((res,rej)=>{
309         //     if(true){
310         //         res(10);
311         //     }else{
312         //         rej("reject ho gya")
313         //     }
314         // })
315         // data.then((item)=>{
316         //     console.log(item);
317         //     return item*10;
318         // }).then((item)=>{
319         //     console.log(item)
320         // }).catch((err)=>{
321         //     console.log(err)
322         // })
323         
324         
325         
326         
327         
328         // finally  keyword
329         // finnal block hamesha chalega chahe reject ho ya resolve ho
330         
331         // let data=new Promise((resolve,reject)=>{
332         //     setTimeout(()=>{
333         //         resolve(10);
334         //     },2000)
335         // })
```

```
        // data.finally((item)=>{
        //     console.log("this is finnal block")
        // }).then((item)=>{
        //      console.log("this is resolve")
        // })



        // throw new error chalne par hamese catch block mai hi jayeha

        // let data=new Promise((resolve,reject)=>{
        //     setTimeout(()=>{
        //          resolve("done");
        //      },2000)

        // })

        // data.then((item)=>{
        //     throw new Error("data issue")
        // console.log("then block",item);
        // }).catch((err)=>{
        //     console.log("catch erroe",err)
        // })



        // promise.all
```

```
        // anuj bhaiya
        //  in Javascript a promise is a good way to handle asynchronous operation it is used to fined out if the asynchronous operation is successfully completed or not.
```

```
        // a promise may have three state
        // pending   fullfilled   reject


        // creating a promise
        // to create a promise we use the Promise() constructor

        // the promise constructor takes a function as an argument .
        // the function also accepts two function resolve() and reject() if the promise return successfully the resolve function is called and if an error occurs the reject function is called


        // using a promise

        // then() method
        // the then method is used with the callbacks when the promise is succrssfully fullfiled or resolve

        // the catch()  method
        // the catch method is used with the callback when the promise is rejected or if an error occurs.

        // the finally() method
        // the finally method gets executed when the promise is either resolved successfully or rejected



        // const ticket=new Promise(function (resolve,reject){
        //     const isBoarded=true;
        //     if(isBoarded){
        //         resolve("your are not in the flight")
        //     }else{
        //         reject("your flight has been cancelled")
        //     }
        // })

        // ticket.then((data)=>{
        //     console.log("wohoo",data)
        // }).catch((data)=>{
        //     console.log("o no",data)
        // }).finally(()=>{
        //     console.log(" i will always be executed")
        // })




        // example 2
        // function getCheese() {
        //     return new Promise((reslove, reject) => {
        //         setTimeout(() => {
        //             const cheese = "🧀";
        //             reslove(cheese);
        //         }, 2000);
        //     })
        // }
        // function makeDough(cheese) {
        //     return new Promise((reslove, reject) => {
```

```
448         //         setTimeout(() => {
449 
450         //             const dough = cheese + "🫓"
451         //             reslove(dough);
452         //         }, 2000);
453         //     })
454         // }
455         // function bakePizza(dough) {
456         //     return new Promise((reslove, reject) => {
457         //         setTimeout(() => {
458 
459         //             const pizza = dough + "🍕"
460         //             reslove(pizza);
461         //         }, 2000);
462         //     })
463         // }
464 
465 
466 
467 
468         // getCheese()
469         //     .then((cheese) => {
470         //         console.log("here is your cheese", cheese)
471         //         return makeDough(cheese)
472         //     }).then((dough) => {
473         //         console.log("here is your dough", dough);
474         //         return bakePizza(dough)
475         //     }).then((pizza) => {
476         //         console.log("here is your pizza", pizza);
477         //     }).catch((data) => {
478         //         console.log("some thing error")
479         //     })
480 
481 
482 // example 3 using async await
483 // function getCheese() {
484 //             return new Promise((reslove, reject) => {
485 //                 setTimeout(() => {
486 //                     const cheese = "🧀";
487 //                     reslove(cheese);
488 //                 }, 2000);
489 //             })
490 //         }
491 //         function makeDough(cheese) {
492 //             return new Promise((reslove, reject) => {
493 //                 setTimeout(() => {
494 
495 //                     const dough = cheese + "🫓"
496 //                     reslove(dough);
497 //                 }, 2000);
498 //             })
499 //         }
500 //         function bakePizza(dough) {
501 //             return new Promise((reslove, reject) => {
502 //                 setTimeout(() => {
503 
504 //                     const pizza = dough + "🍕"
505 //                     reslove(pizza);
```

```
//                 }, 2000);
//             })
//         }

// async function orderPizza(){
//     try{
//         const cheese=await getCheese();
//         console.log("here is your cheese",cheese);
//         const dough=await makeDough(cheese);
//         console.log("here is your dough",dough);
//         const pizza=await bakePizza(dough);
//         console.log("here is your pizza",pizza);
//     }catch(err){
//         console.log("error happen",err);
//     }
//     console.log("ended")

// }

// orderPizza()
```

```
    </script>
</body>

</html>
```