DAY - 4

setTimeInterval

- The setInterval() method calls a function at specified intervals (in milliseconds).
- The setInterval() method continues calling the function until clearInterval() is called, or the window is closed.
- 1 second = 1000 milliseconds.

```
let count=0;

const countInterval= setInterval(count1,3000)

function count1(){

        if(count===4){

                clearInterval(countInterval);

         }

         console.log("Nammatha");

         count++;

};
```

setTimeout

- The setTimeout() method calls a function after a number of milliseconds.
- 1 second = 1000 milliseconds.
- If you need repeated executions, use setInterval() instead.
- Use the clearTimeout() method to prevent the function from starting.

```
setTimeout(greeting,3000);

function greeting(){
   console.log("Hello2");
}
```

```
const myTimeout = setTimeout(myGreeting, 5000);

function myStopFunction() {

  clearTimeout(myTimeout);

}
```

Exception handling-

Try, throw, Catch

Try - we can write a code that expects errors to come.

Throw - throwing the error

Catch - catch the throw error and displayed

Promises:-

Instead of callbacks we use promises or method chaining.

```
function walkDog(){

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

        setTimeout(()=>{

            const dogwalk = true;

            if(dogwalk){

                resolve("Dog is walking");

            }else{

                reject("Dog is not walking");

            }

        },2000)

    })

}



function cleanKitchen(){

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

        setTimeout(()=>{
            const clean = true;

            if(clean){

                resolve("Kitchen is Cleaned")

            }else{

                reject("Kitchen is not cleaned");

            }

        },2000)

    })

}



function trash(){

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

        setTimeout(()=>{

            const trashww = true;

            if(trashww){

                resolve("Trash is though");

            }else{

                reject("Trash is filled");

            }
        },2000)

    })

}



walkDog().then(value=>{console.log(value);

     return cleanKitchen()}) .then(value=>{console.log(value);

        return trash()}).then(value=>{console.log(value)}).

        catch(error=>console.error())
```

Hands on programming:-

```
// SetTimeOut

// setTimeout ( function , milliseconds, parameters1 , parameter2 ... )


setTimeout(greeting,3000);


function greeting(){

   console.log("Hello2");

}
```

```
setTimeout(greeting1,3000, 'Hiii');


function greeting1(x){

    console.log(x);

}

/*---------------------------------*/


setTimeout(()=>{

    console.log('Naveen');

},3000);


//-------------------------------------------

// setInterval ( function , milliseconds, parameters1 , parameter2 ...
)

// let count=0;

// const countInterval= setInterval(count1,3000)

// function count1(){

//      if(count===4){

//          clearInterval(countInterval);

//      }

//      console.log("Nammatha");
//      count++;

// };

//
//======================================================================



let p = new Promise((resolve, reject) => {

    let n = 1+21;

    if(n==3){

        resolve("pass");

    }else{

        reject("failed");

    }

})



p.then((message)=>{

    console.log('Then is in ' + message);

}).catch((message)=>{

    console.log("catch "+ message);

})



//=================================================================
// try{

//      let n = prompt('Enter a number')

//      if(n=='')

//          throw("Should not be empty")

//      if(isNaN (n))

//          throw("Enter a number");

//      console.log(n*2);

// }

// catch(error){

//      console.log(error)

// }



/*========================*/



function walkDog(){

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

        setTimeout(()=>{

            const dogwalk = true;

            if(dogwalk){

                resolve("Dog is walking");

            }else{
                reject("Dog is not walking");

            }

        },2000)

    })

}



function cleanKitchen(){

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

        setTimeout(()=>{

            const clean = true;

            if(clean){

                resolve("Kitchen is Cleaned")

            }else{

                reject("Kitchen is not cleaned");

            }

        },2000)

    })

}



function trash(){

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

        setTimeout(()=>{

            const trashww = true;

            if(trashww){

                resolve("Trash is though");

            }else{

                reject("Trash is filled");

            }

        },2000)

    })

}



walkDog().then(value=>{console.log(value);

return cleanKitchen()}) .then(value=>{console.log(value);

return trash()}).then(value=>{console.log(value)}).

catch(error=>console.error());
```