WEB講義 参考プリントNo.1

・setTimeout/setInterval

Ex1

setTimeout(()=>{

    console.log("hello world")

},1000)

setInterval(()=>{

    console.log("hello world(繰り返し)")

})

(1000ミリ秒=１秒)

setTimeout(関数,ミリ秒)でミリ秒後に関数を実行する。

setInterval(関数,ミリ秒)でミリ秒ごとに実行する。

・async/await 非同期処理

Ex2 promise

const promise = new Promise((resolve, reject) => {

    setTimeout(() => {

        resolve("hello world")

    }, 1000)

})

promise.then(data => {

    console.log("log:" + data)

}).catch(e=>{

    console.log("エラーが出ました。"+e)

})

>>> "log:hello world"

>>> "hello world" (一秒後)

Promiseとは非同期の処理をするためにある。

Resolveは解決、rejectはエラーを出すための関数。

今回ではこのpromiseはPromise<string>型になる。stringを後で返す、という意味の関数。

Thenはこの後に実行する。Catchはエラーが出たときの処理。

Ex3 １秒待つコード

async function main() {

    console.log("1")

    await new Promise((resolve) => {

        setTimeout(resolve, 1000)

    })

    console.log("2")

}

main()

>> "1"

>> "2"(１秒後)

Promiseを待機した後に実行するには、asyncを関数の前につけ、非同期関数にする必要がある。

Ex4 非同期関数

async function getStringInAMinute(){

    await new Promise((resolve) => {

        setTimeout(resolve, 1000)

    })

    return "hello world"

}

const result= await getStringInAMinute()

非同期関数にreturnをつけて返り値をつけるようにすると、Promiseと同様にawaitを用いることができる。(今回はawaitが非同期関数の中で実行されていないため、エラーが出る。Ex3にようにmain非同期関数内で実行すること。)

Ex5 axiosでpost,get通信をする

const axios=require("axios")

async function main(){

    const response=axios.get("https://google.com")

    // const response=axios.post("https://google.com")

    // postも実行できる。

    const data=response.data

    console.log(data)

}

main()

axiosというパッケージを用いてコードを実行する。

post通信: データをURLに渡して、データをもらう。

get通信: 単にURLからデータを得る。普段サイトをみるときのhtmlはget通信で取得されている。