JavaScript

callbacks, Promise & Async Await

CodeMaster Noyon

JS callback

Synchronous means the code runs in a particular sequence of instructions given in the program.

- Line 1 execute
- Line 2 execute
- Line 3 execute

Asynchronous code execution allows to execute next instructions immediately and doesn't block the flow.

- Line 1 execute
- Line 2 takes time, Line 3 execute
- Line 2 execute

CodeMaster Noyon

JS callback

Functions that are used as parameters to another function is called callback function.

```
const display = (result)=>{
    console.log(result);
}

const sum = (a, b, callback)=>{
    let result = a+b;
    callback(result);
};

sum(10, 20, display);
```

CodeMaster Noyon

JS callback Hell

Callback Hell occurs by nested callback.

```
function getData (id, getNextData){
    console.log('Data: ', id);
    if(getNextData){
        getNextData()
    }
}

getData(1, ()=>{
    getData(2, ()=>{
        getData(3, ()=>{
            getData(4)
        })
    })
})
```

```
// Data:  1
// Data:  2
// Data:  3
// Data:  4
```

CodeMaster Noyon

JS Promise

For improving callback hell we can use promise. Promise object has three state:

- Pending `// when there is no final result, promise successful or reject`
- Fulfilled `// when is promise successfully done`
- Rejected `// when is promise reject`

```
// creaating new promise
let newPromise = new Promise((resolve, reject)=>{
    if(true){
        resolve()
    }else{
        reject()
    }
})
```

CodeMaster Noyon

JS Promise chain

```
function getData(dataID, getNextData){
    return new Promise((resolve, reject)=>{
        setTimeout(() => {
            console.log('Data: ', dataID);
            resolve('Success')
        }, 2000);
    })
• }
•
```

```
// promise chain
getData(1)
.then((res)=>{
     return getData(2)
})
.then((res)=>{
    return getData(3)
})
.then((res)=>{
    console.log(res)
})
```

CodeMaster Noyon

JS Async-Await

To improving promise chain we can use Async-Await:

```
function api (id) {
    return new Promise((resolve, reject)=>{
        setTimeout(() => {
            console.log('Data fetching: ', id)
            resolve('Data...')
        }, 2000);
    })
}
```

```
// async-await
async function getInfo(){
    await api(1);// 1st call
    await api(2);// 2nd call
    await api(3);// 3rd call
}
getInfo();
```

CodeMaster Noyon