Example#1

let obj = {

    name : "ram"

}

function show() {

    console.log(this.name);

}

show.call(obj);

Example#2

let obj = {

    name : "ram"

}

function show(x) {

    console.log(this.name, x);

}

show.call(obj,10);

Example#3

let obj = {

    name : "ram"

}

let arr1 = [300, 400];

function show(arg1, arg2) {

    console.log(this.name, arg1, arg2);

}

show.apply(obj, arr1);

Example#4

let obj = {

    name : "ram"

}

function show(x) {

   console.log(this.name, x);

}

let new\_loc = show.bind(obj);

new\_loc(10);

call back function

Example#4

function show1(arg1) {

    console.log(arg1());

}

show1 ( function show2() {

    return "welcome..";

})

Example#5

function show1(arg1) {

    console.log(arg1());

}

show1 (()=>{ return "hello"})

Example#6

function show1(arg1) {

    arg1(10, 20);

}

show1 ((x, y )=>{

    console.log(x,y);

})

Example#7

function show1(x, arg1) {

    console.log(x);

    arg1(10, 20);

}

show1 (10, (x, y )=>{

    console.log(x,y);

})

Example#8

function add(res1,  callback) {

    callback(res1+ 20);

}

function sub(res, callback){

    callback(res+3)

}

add(10, (result)=>{

    sub(result - 5, (x)=>{

        console.log(x);

    })

});

Call back hell

Promises

* Proper communication between producer and consumer is called promises.
* Producer will create the promises.
* Consumer will consume the process.
* Promises have 2 state
  + Resolve
  + Reject
* We will create the promises by using Promises class.
* We will consume the promises in 2 ways:
  + then()
  + async and await
* Promises are special java script objects.

Example#1

let promise1 = new Promise((resolve, reject)=>{

    resolve("welcome to promises");

});

promise1.then(

    (successRes)=>{

        console.log("success:"+successRes);

    },

    (errorRes)=>{

        console.log("Error:"+errorRes)

    }

);

Example#2

let promise1 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("welcome to promises");

    }, 5000);

});

promise1.then(

    (successRes)=>{

        console.log("success: "+successRes);

    },

    (errorRes)=>{

        console.log("Error: "+errorRes)

    }

);

Example#3

let promise1 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise1...");

    }, 0);

});

let promise2 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise2...");

    }, 5000);

});

let promise3 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise3...");

    }, 10000);

});

promise1.then(

    (successRes)=>{

        console.log("success: "+successRes);

    },

    (errorRes)=>{

        console.log("Error: "+errorRes)

    }

);

promise2.then(

    (successRes)=>{

        console.log("success: "+successRes);

    },

    (errorRes)=>{

        console.log("Error: "+errorRes)

    }

);

promise3.then(

    (successRes)=>{

        console.log("success: "+successRes);

    },

    (errorRes)=>{

        console.log("Error: "+errorRes)

    }

);

Example#4

let promise1 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise1...");

    }, 0);

});

let promise2 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise2...");

    }, 5000);

});

let promise3 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise3...");

    }, 10000);

});

Promise.all([promise1, promise2, promise3]).then(

    (successRes) => {

        console.log(successRes);

    },

    (errorRes) => {

        console.log(errorRes);

    }

);

Example#5

let promise1 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise1...");

    }, 0);

});

let promise2 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        reject("promise2...");

    }, 5000);

});

let promise3 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise3...");

    }, 10000);

});

Promise.all([promise1, promise2, promise3]).then(

    (successRes) => {

        console.log(successRes);

    },

    (errorRes) => {

        console.log(errorRes);

    }

);

Example#6

let promise1 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise1...");

    }, 0);

});

let promise2 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise2...");

    }, 5000);

});

let promise3 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise3...");

    }, 10000);

});

Promise.race([promise1, promise2, promise3]).then(

    (successRes) => {

        console.log(successRes);

    },

    (errorRes) => {

        console.log(errorRes);

    }

);

Example#7

let promise1 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise1...");

    }, 0);

});

let promise2 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        reject("promise2...");

    }, 5000);

});

let promise3 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise3...");

    }, 10000);

});

Promise.allSettled([promise1, promise2, promise3]).then(

    (successRes) => {

        console.log(successRes);

    },

    (errorRes) => {

        console.log(errorRes);

    }

);

**asyn and await**

Example#1

let promise1 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise1...");

    }, 0);

});

let promise2 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise2...");

    }, 5000);

});

let promise3 = new Promise((resolve, reject)=>{

    setTimeout(()=>{

        resolve("promise3...");

    }, 10000);

});

async function consume() {

    let res1 = await promise1;

    let res2 = await promise2;

    let res3 = await promise3;

    console.log(res1, res2, res3);

}

consume();

Example#2

function add(num, callback) {

    return callback(num+10);

}

function sub(num, callback) {

    return callback(num-5);

}

function mul(num, callback) {

    return callback(num\*5);

}

function div(num, callback) {

    return callback(num/2);

}

add(5, (addRes) => {

   sub(addRes, (subRes)=>{

        mul(subRes, (mulRes)=>{

            div(mulRes, (divRes)=>{

                console.log(divRes);

            })

        })

   })

});

Example#3

function add(num) {

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

        resolve(num+5);

    });

}

function sub(num) {

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

        resolve(num-5);

    });

}

function mul(num) {

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

        resolve(num \* 5);

    });

}

function div(num) {

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

        resolve(div / 2);

    });

}

async function consume() {

    let addRes = await add(5);

    let subRes = await add(addRes);

    let mulRes = await add(subRes);

    let divRes = await add(mulRes);

    console.log(divRes);

}

consume();