Generators

```
//ES6 Generators
//Generators special type of iterator er kaj value generate kora. Se ekta ekta kore value generate kore.
//Iterator o ekta ekta kore kaj kore
//Generators function k star(*) diya likte hoi
//Generator ke return kora jabe na yield korte hoi yield o return er moto e.
//Generator function ke call korle ami ekta iterator pabo.


//Yield & return er syntextic use
function * generator(){//generator function //Star k bibino vabe dewa jai
    yield 1;//value produce korsi
    yield 2;
    return "I am finished"//return kore dile last step dore nibe // So jokhon ami chai user k r value dibo na ami return
 kore dite pari she khane
    yield 3;
    yield 4;
}
let iterator = generator();//generator k call korlam akhon amra iterator pabo.
console.log(iterator.next());//prothom yield print hower por she next yield shomporke jane e na she next yield call e
korbe na mane she 1 generate korbe pause hoiya jabe. Then abr .next() call hole print hobe
console.log(iterator.next());
console.log(iterator.next());
console.log(iterator.next());
console.log(iterator.next());
//generator e ami amr iccha moto value produce kore er por abr amr iccha moto value pause rakhe rakhe certain timey ami
bar kore niya ashte parbo amk value kothaw store kore rakte hobe na.
//So yield value generate kore & return close kore dai or thamiya dei









//Syntaxes of Generator Functions

//How to use star in constractor function
const myGenerator = function*() {}

//Araw function k generator akare likha jai na

//Generator Functions in object literal
const myObject = {
    *myGenerator()

}









//Make Object Iterable
//Object er khatre amra for of use korte pari na karon object iterable na So object k jodi amra iterable banate pari
baniya tar betor jodi amra Symbol.iterator er implementation korte pari tahole object er khatre spread for of shob e use
 korte parbo.
//So akhon amra ai rokom er ekta example dekbo er por shetake amra convert korbo generator a
//Iterator er upgrade function hosse genertor


let object = {
    value1: 1,
    value2: 2,
    value3: 3,
    value4: 4,
    value5: 5,
};//ai obj er moddo diya amr for of, spread operator implement korte pari na


for(let element of object);
```

```
console.log([...object]);//error


console.log({...object});//ai vabe amra spread korte o pari but shaita onno implementation iterator er implementation na
 mane iterator use kore nai she khane.








//Iterator use kore object k iterator banabo....lets do
let object = {
    value1: 1,
    value2: 2,
    value3: 3,
    value4: 4,
    value5: 5,
};
// console.log(Object.entries(object));//[["value1", 1], ["value2", 2], ["value3", 3] , ["value4", 4], ["value5", 5]]


//Object.prototype er modda Symbol.iterator dukiya dibo karon she khane to iterator nai
Object.prototype[Symbol.iterator] = function() {//akhon iterator function er moto e
   const entries = Object.entries(this);//Object.entries object k array te convert kore dai
    let count = entries.length;//koita element ache jante hobe to next implenent korer jonno so entries.length
    let index = 0; //protita index theke element bar kore niya ashte hobe tai index er o ekta track rakte hobe so index
= 0.
   //bar bar call korer shomoi count, index value gula k memory te dore rakbe
    return {
        next() {
            if(count > 0) {
                let resut = {done: false, value: entries[index][1]};
//index= value1, 1=1  that mean ["value1", 1] k indicate korse
                count --;
                index++;//result er porey ++ -- kora lagbey ta result er por ++ -- korsi
                return resut;
            }
            return {done: true};//count 0 er shoman hole done true kore dibe

        },

    };
};
//Akhon for of use korle kaj korbe
for(let element of object){
    console.log(element);//object k iterable baniya fellam
    //Output: 1,2,3,4,5 ...line by line
}
console.log([...object]);//spread operator o kaj korse
//Output:(5) [1, 2, 3, 4, 5]









//Aita generator diya korbo akhon
let object = {
    value1: 1,
    value2: 2,
    value3: 3,
};
// console.log(Object.entries(object));//[["value1", 1], ["value2", 2], ["value3", 3] , ["value4", 4], ["value5", 5]]
function *generator(){//generator function
    const entries = Object.entries(object);
    for(let element of entries){
        yield element[1];//["value1", 1] aita pabo
    }//kono function a return na dile she by default nije e return hoiya jai
}
```

```
//generator function iterator return kore
//its so simple/easy amk kono protocol use korte hosse na
//programmer er life easy korte e generator, amk manully bole dite hosse na

const iterator = generator(object);//call korlam function k
// console.log(iterator.next());
// console.log(iterator.next());
// console.log(iterator.next());
// console.log(iterator.next());

console.log([...iterator]);//spread operator er khatre o kaj korse
//output: (3) [1, 2, 3]






//Example 2
//Range function (iterator tutorial er Range function k generator a convert)

function *range(start, end, step) {
    let current = start;

    while (current <= end) {
        yield current;//output 1 dilo pause hoiya ase er por ami onnano kaj korte parbo
        console.log("Did yoju execute?");//last output er por kisui ashbe na ai khane
        current += step;//1 print kore 1+2 korlo akhon
    }

}

var iterator = range(1, 1000, 2);//call korlam
console.log(iterator.next());//output: {value: 1, done: false}
console.log(iterator.next());//output: {value: 3, done: false}

//generaator use kore indirectly ami iterator k call korsi amk onek code likte hosse na generator e kore disse






//Generator control flow
//example
function *generator(a, b) {
    let k = yield a + b;
    let m = yield a + b + k;

    yield a + b + k + m;
}

var gen = generator(10, 20);
console.log(gen.next());//yield a + b = 30 kore bariya ashse k er kase jai ni //output:{value: 30, done: false}
console.log(gen.next(50));//thik je khane sesh hoiya chilo er por let k = 50 boshbe er por m = 10+20+50=80
//output:{value: 80, done: false}
console.log(gen.next(100));//output: {value: 180, done: false}  //let m =100, yiels 10+20+50+100=180
console.log(gen.next());// output: {value: undefined, done: true}
//ai kahne amk async way te o korte hosse na ami sync way te e kaj korte passi easly, so ai kahane stack full hosse na
bar bar call kosse na







//async-await o generator use korse
//generator async-await er jonmo dhata
```

```
//higher level Abstraction to Generators(async-await)
//async-await tutorials example k use korbo


//shei example ta....jake use korbo...
const takeOrder = (customer, callback) => {//callback pass korer jonno callback recive korte hobe
    console.log(`take order for ${customer}`);
    callback(customer);//callback pass kore dibe
}

const processOrder = (customer, callback) => {
        console.log(`Processing order for ${customer}`);

        setTimeout(() => {
            console.log(`order processed for ${customer}
            `);
            callback(customer);
        }, 3000);

    };

    const completeOrder = (customer) => {
        console.log(`Complete order for ${customer}`);
    }
    //tackOrder k call korlam parameter pass korsi customer & callback pass korsi
    takeOrder('customer 1', (customer) => {
        processOrder(customer, (customer) => {
            completeOrder(customer);
        })

    });






//Now convert example....

const takeOrder = (customer) => {
    console.log(`take order for ${customer}`);

}

const processOrder = (customer) => {
        console.log(`Processing order for ${customer}`);

        setTimeout(() => {
            console.log(`order processed for ${customer}`);
        }, 3000);

    };
const completeOrder = (customer) => {
    setTimeout(() => {
        console.log(`Complete order for ${customer}`);
    }, 1000);

};


function *solution(customer) {
    yield takeOrder(customer);//callback to r korte hobe na ai khane
    yield processOrder(customer);
    yield completeOrder(customer);
}

//je vabe iterator bar kore niya ashi
const gen = solution("karim");//next kori nai output ashe na ai khane
gen.next();
gen.next();
gen.next();










//So ami ekta ekta kore valu passi but sequentially jinish gula korte parsi na so call back hell er solution chilo promise
//amra kisu return kori na so yield kisu dite parse na so pottek jaiga tkeke promise return kore dibo
//Now convert example....
//async generator bujsi..So iterator er o ekta async version thakte hobe  So 2018 js amder [Symbol.asyncIterator]() diyase so aita e async generator use kore
//generator backend a iterator e use kore
const takeOrder = (customer) => {
    return new Promise((resolve) => {
        setTimeout(() => {
       // console.log(`Order take order for ${customer}`);//console.log o kora jai but relove e korbo promise jahutu
       resolve(`Order take order for ${customer}`);
        }, 1000);

    });
    }

const processOrder = (customer) => {
    return new Promise((resolve) => {
        setTimeout(() => {
            resolve(`order processed for ${customer}`);
        }, 3000);

    });
    };
const completeOrder = (customer) => {
    return new Promise((resolve) => {
        setTimeout(() => {
            resolve(`Complete order for ${customer}`);
        }, 1000);

    });
    };


function *solution(customer) {
    yield takeOrder(customer);//callback to r korte hobe na ai khane
    yield processOrder(customer);
    yield completeOrder(customer);
}




//aija amra yield korsi so amder to ekta await dorkar chilo
async function *solution(customer) {
    yield await takeOrder(customer);//callback to r korte hobe na ai khane
    yield await processOrder(customer);
    yield await completeOrder(customer);
}

const gen = solution("karim");




// gen.next();//console a ai kaj ta korle valo vabe buja jai 1st a proime disse then value disse
gen.next().then((value) => console.log(value));//aita pura object
//output:{value: "Order take order for karim", done: false}
gen.next().then(({value}) => console.log(value));//object er shudu value bar korbo ai vabe dorkar porle
//output:order processed for karim
gen.next().then(({value}) => console.log(value));
```

```
//then er por value te output ta passi






//promise gulo loop kore ek shate pete chile? lets do it...
const promise = [gen.next(),gen.next(),gen.next()];//ai 3 tai promise async code
//aita k for diya iterate korte parbo na karon for sync so amra for of use korbo
for(let p of promise) {//aita kaj korbe na karon const promise aita to async jinish

console.log(p);
}



//So await boshi dibo
const promise = [gen.next(),gen.next(),gen.next()];
for await(let p of promise) {
    console.log(p);
}







//async chara to await hoina so ekta async IIfi function er modda niya nibo
const promise = [gen.next(),gen.next(),gen.next()];
(async function(){
    for await(let p of promise) {//async er js for await diyase
        console.log(p);
    }

})();













//Async Iterators
//Symbol.async iterator ektu details dekbo
//age amra custom iterable baniyasi akhon Custom async itrable banabo
const myAsyncIterator = {
   async *[Symbol.asyncIterator]() {//aita k ekta generator function baniya falsi jabe manually onek code na korte hoi
        yield "hello";
        yield "async";

    },
};
//ekta async banalam so aita te for of spread korte parer kotha so akhon tai korbo
(async () => {
    for await( let x of myAsyncIterator){
    console.log(x);
    }
})();





// Summery........
// 1. Generator er kisu special behavior ache she iterator er shomoshto jinish use korte pare & pura state remember korte pare
// 2. Generator sync async both khatre kaj kore & yield korer por kisu kaj kore baire theke value dite pari thik ja
jaigai ager yield rakhe ashsi, shei value k abr use korte parci
```

```
// 3.ja kahne pause hoiya chilo shai jaga theke e suro korte parsi abr
// that mein......ja value generate kore, pause behavior ase, baire thaka value nite pare, she ekta iterator return kore
 ja amra yield kore kore next next value gula pate pari
```