Advance Node js:

callback pattern:

ashynchorouns task should execute sequencially

promisify is a function that convert callback function into promises

async will use for asynchronous function callbacks

await will use for waiting for the output of that statement

promise.all used for executing all parllelly-> the output will wait untill all promises will resolved

promise.race will used for executing all parllelly->the out put will wait at least one promise fullfilled.

```
//ex1.js
function delay(seconds, callback) {
    setTimeout(callback, seconds * 1000);
}
console.log("delay started");
delay(3, () => {
    console.log("three seconds");
    delay(1, () => {
        console.log("four seconds");
        delay(1,()=>{
            console.log("five seconds");
        })
    })
})
```

```
//ex2.js
delay = function(seconds){
    return new Promise(function(resolves,reject){
        setTimeout(resolves,seconds*1000);
    })
}
delay(1).then(function(){
    console.log("one second");
});


var delay1 = (seconds)=>new Promise((resolves,reject)=>{
    setTimeout(resolves,seconds*1000);
})
```

```
delay1(1).then(()=>{
    console.log("next loop second");
})


/*function delay(seconds,callback){
    setTimeout(callback,seconds*1000);
}

delay(1,()=>{
    console.log("one second");
});*/
console.log("process");
```

```
//ex3.js
var delay = (seconds)=>new Promise((resolves,reject)=>{
    setTimeout(()=>{
        resolves("first call")
    },seconds*1000)
});
delay(1)
.then(console.log)
.then(()=>42)
.then((value)=>console.log(`the value is ${value}`));
console.log("started");
```

```
//ex4.js
var delay = (seconds)=>new Promise((resolves,reject)=>{
    if(seconds>3){
        reject(new Error(`${seconds} seconds too hign to load!`));
    }
    setTimeout(()=>{
        resolves("the first call");
    },seconds*1000);
});
delay(4)
.then(console.log)
.then(()=>42)
.then((value)=>console.log(`the value is ${value}`))
.catch((error)=>console.log(error.message));
console.log("started");
```

```
//promise-all.js
```

```
var fs = require("fs");
var {promisify} = require("util");
var writeFile = promisify(fs.writeFile);
var readdir = promisify(fs.readdir);
var unlink = promisify(fs.unlink);

var delay = (seconds)=>new Promise((resolves,reject)=>{
    setTimeout(resolves,seconds*1000);
});
Promise.all([
    //writeFile("readme.md","sample me file"),
   // writeFile("readme.txt","some txt file")
   unlink("readme.md"),
   delay(5),
   unlink("readme.txt")
]).then(()=>readdir(__dirname))
.then(console.log)
```

```
//promise.race.js
var fs = require("fs");
var { promisify } = require("util");
var writeFile = promisify(fs.writeFile);
var readdir = promisify(fs.readdir);
var unlink = promisify(fs.unlink);

var delay = (seconds)=>new Promise((resolves,reject)=>{
    setTimeout(resolves,seconds*1000);
});

Promise.race([
    delay(5),delay(1),delay(10)
]).then(()=>readdir(__dirname))
.then(console.log);
```

```
//promisify.js
var fs = require('fs');
var { promisify } = require('util');
var writeFile = promisify(fs.writeFile);
writeFile('sample.txt',"this is sample file")
.then(()=>{
    console.log("file created successfully");
})
.catch((error)=>{
    console.log(error.message);
```

```
})
```

```
//promisify1.js
var { promisify } = require('util');
var delay = (seconds,callback)=>{
    if(seconds>3){
        callback("error occured");
    }
    else{
        setTimeout(()=>{
            callback(null,"messages displayed");
        })
    }
}

var delayseconds = promisify(delay);
delayseconds(4)
.then(console.log)
.catch((error)=>{
    console.log(error);
})


/*
delay(4,(error,message)=>{
    if(error){
        console.log(error);
    }
    else{
        console.log(message);
    }
})
*/
```

```
//sequence_exe.js
var fs = require("fs");
var { promisify } = require("util");
var writeFile = promisify(fs.writeFile);
var unlink = promisify(fs.unlink)
var beep = ()=>process.stdout.write('\x07');
var delay = (seconds)=>new Promise((resolves)=>{
    setTimeout(resolves,seconds*1000);
})
var deSeq =()=> Promise.resolve()
```

```
.then(()=>console.log("waiting"))
.then(()=>delay(1))
.then(()=>writeFile("sample.txt","this is sample file"))
.then(()=>console.log("file created"))
.then(()=>beep)
.then(()=>console.log("waiting"))
.then(()=>delay(1))
.then(()=>unlink("sample.txt"))
.then(()=>beep)
.then(()=>console.log("file removed"))
.catch((error)=>console.log(error))

deSeq();
```

```
//async-await.js
var fs = require("fs");
var { promisify } = require("util");
var writeFile = promisify(fs.writeFile);
var unlink = promisify(fs.unlink);

var delay = (seconds)=>new Promise((resolves)=>{
    setTimeout(resolves,seconds*1000);
});
const deSeq = async()=>{
    console.log("starting");
    await delay(1);
    console.log("start again");
    try{
        await writeFile("file.txt","some sample code");
    console.log("waiting");
    }catch(error){
        console.log(error);
    }
    await delay(1);
    console.log("waiting");
    await delay(5);
    console.log("waiting");

    await unlink("file.txt");
    console.log("file removed");
    console.log("ended");
    return Promise.resolve;
}
deSeq()
```

```
.then();
```

```
//async-await2.js
var fs = require("fs");
var { promisify } = require("util");
var writeFile = promisify(fs.writeFile);
var unlink = promisify(fs.unlink);
var readdir = promisify(fs.readdir);

var delay = (seconds)=>new Promise((resolves)=>{
    setTimeout(resolves,seconds*1000);
});
async function start(){
    var files = await readdir(__dirname);
    console.log(files);
}
start();
```

```
//callapplybind.html
<script>
    var obj = {"name":"chandra","address":"plvd"}
    var fun1 = function(a,b,c){
        console.log("welcome"+this.name+"to"+a+"and"+b+","+this.address);
    }
    //fun1.call(obj,"HYD","INNDIA");
    //var args = ["HYD","INDIA","kdp"];
    //fun1.apply(obj,args);
    var bindvar = fun1.bind(obj);
    bindvar("HYD","AP","cuddapah");
</script>
```