ES6

Modules

What is module?

An ES6 module is a file containing JS code

- Encapsulates a topic/logic
- Can have dependencies (using other modules)
- Can have dependants (used by other modules)

sample module

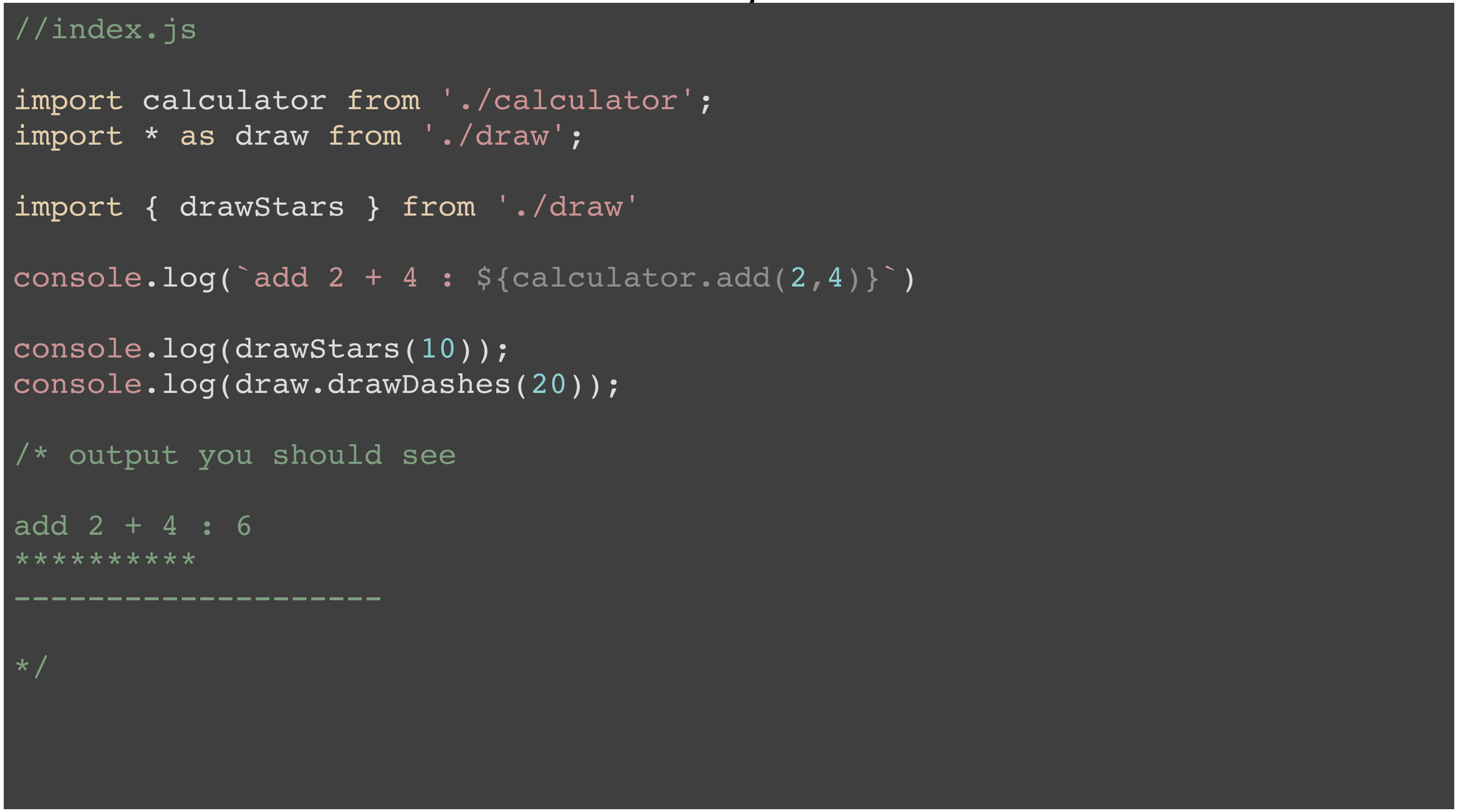
```
//calculator.js

class Calculator {

    constructor(){}

    static add(a,b){
        return a + b;
    }

    static substract(a,b)
    {
        return a - b;
    }

}

module.exports=Calculator;
```

sample module

```
//draw.js

export function drawStars(n){
   let stars="";
   for(let i=0;i<n;i++)
   {
      stars+="*";
   }
   return stars;
}

export function drawDashes(n)
{
   let dashes="";
   for(let i=0;i<n;i++)
   {
      dashes+="*";
   }
   return dashes
}
```

How can I use module?

index.js



```
//index.js

import calculator from './calculator';
import * as draw from './draw';

import { drawStars } from './draw'

console.log(`add 2 + 4 : ${calculator.add(2,4)}`)

console.log(drawStars(10));
console.log(draw.drawDashes(20));

/* output you should see

add 2 + 4 : 6
**********
--------------------

*/
```

How can I use modules in nodeJS?

Install babel

Node is still not ready for full ES6. However it can work with babel transpiler

```
//First we'll install babel-cli.

$ npm install --save-dev babel-cli


//Install some presets.

$ npm install --save-dev babel-preset-es2015 babel-preset-stage-2


//Update package.json

 "scripts": {
+    "start": "babel-node index.js --presets es2015,stage-2"
  }
```

How can I use modules in nodeJS?

```
//Now you can use npm start to run under babel.

$ npm start

> training@1.0.0 start /Users/varmabhupatiraju/Documents/Projects/training
> babel-node index.js --presets es2015,stage-2

add 2 + 4 : 6
**********
--------------------
```

Promises

What are they?

Promises are special ES6 objects used for deferred and asynchronous computations.

They are used to replace the need for callbacks, and can, in turn, reduce the need for "callback hell".

What are they?

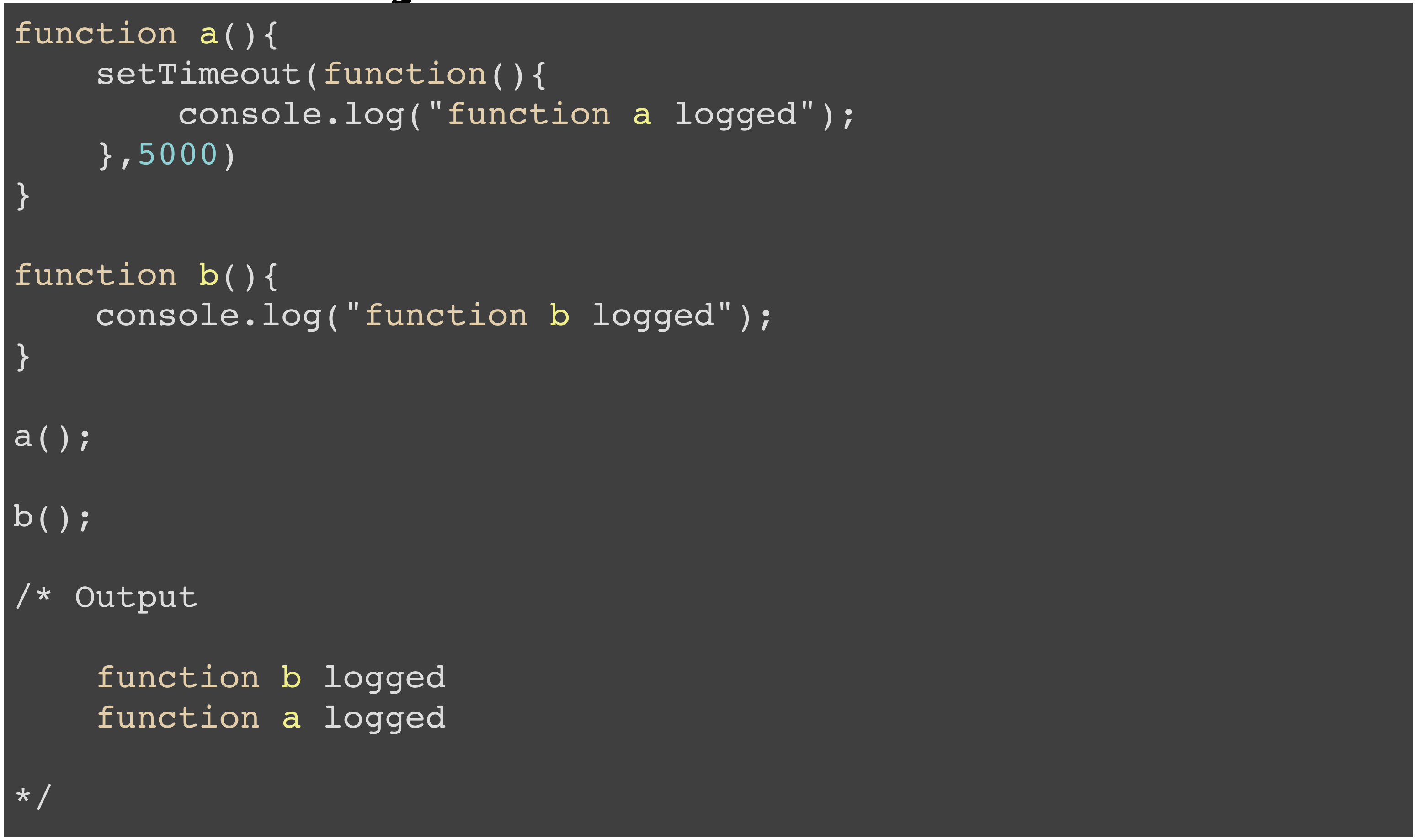
We can't use jQuery ajax or xmlHTTPRequest in Node.js.

They are only meant for browser where DOM exists.

Why we need Promises?

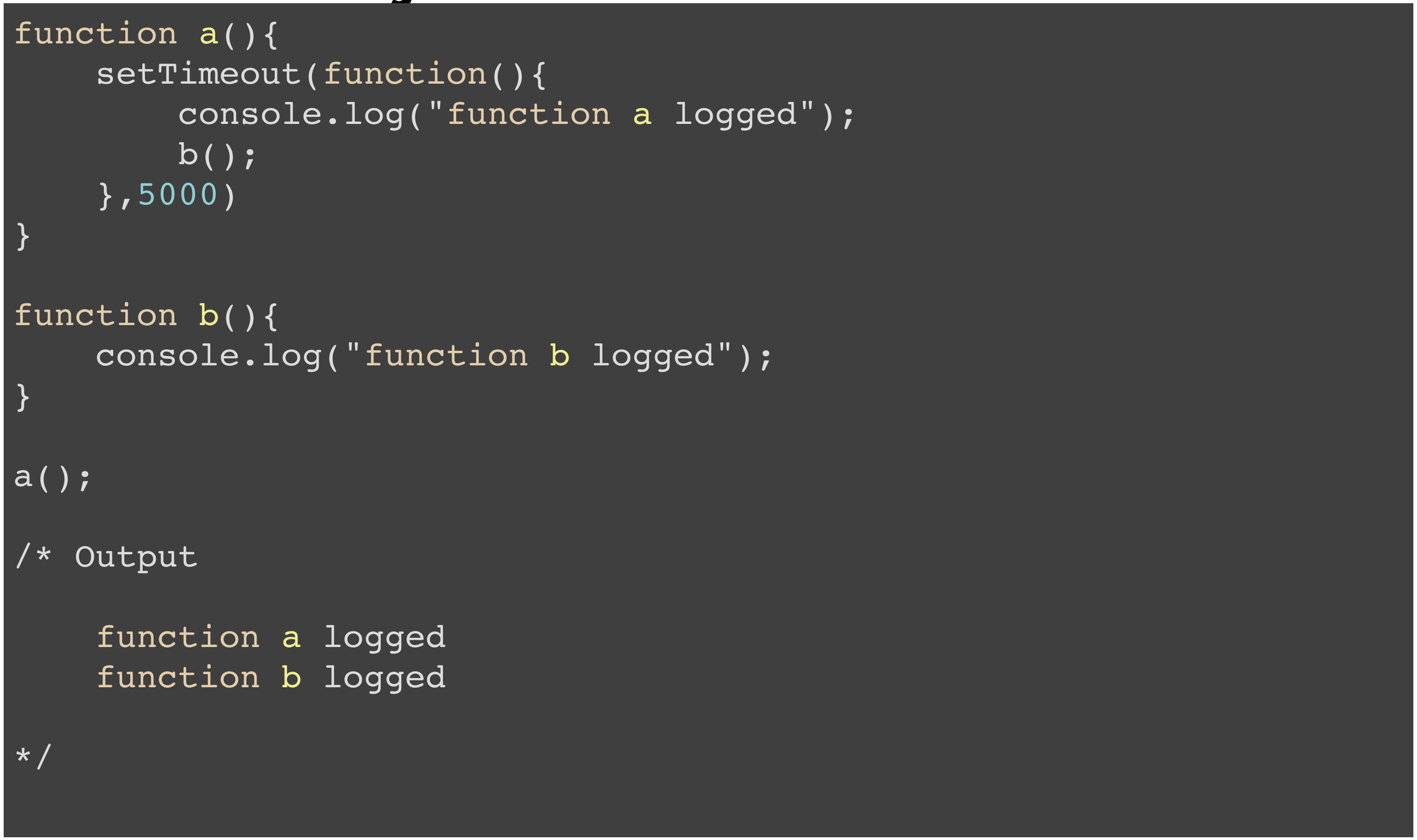
```
function a(){
    setTimeout(function(){
        console.log("function a logged");
    },5000)
}

function b(){
    console.log("function b logged");
}

a();

b();

//Guess output
```

Why we need Promises?



```
function a(){
    setTimeout(function(){
        console.log("function a logged");
    },5000)
}

function b(){
    console.log("function b logged");
}

a();

b();

/* Output

    function b logged
    function a logged

*/
```

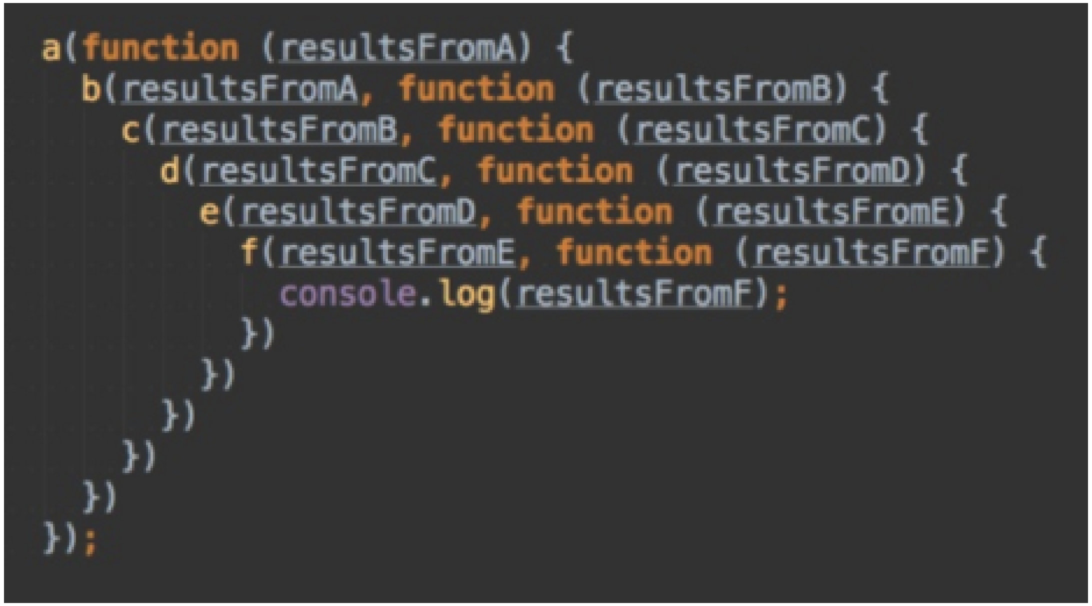
Why we need Promises?



```
function a(){
    setTimeout(function(){
        console.log("function a logged");
        b();
    },5000)
}

function b(){
    console.log("function b logged");
}

a();

/* Output

    function a logged
    function b logged

*/
```

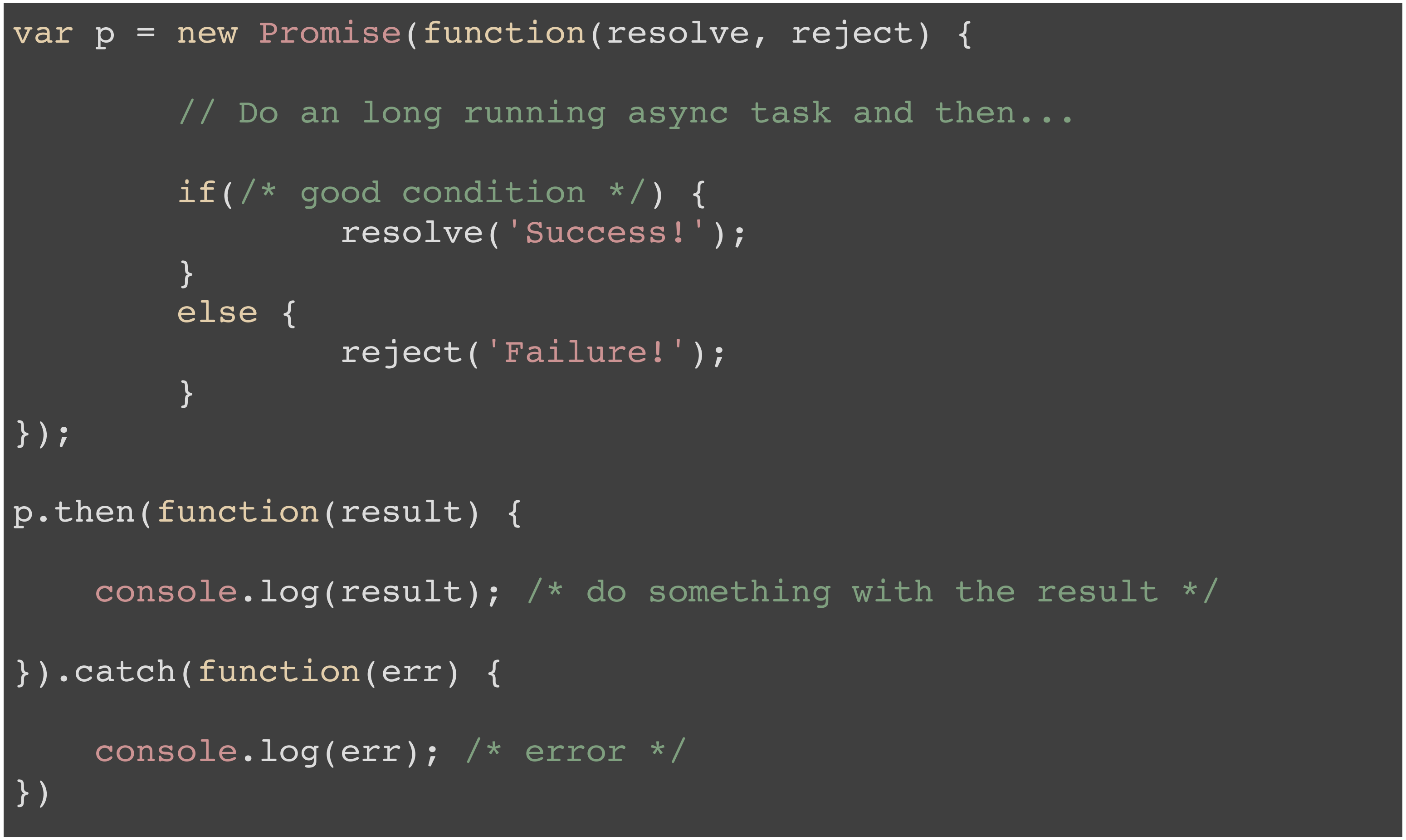
Why we need Promises?

Entire chain of code collapse if one function errors out in the chain & will make program run abnormal



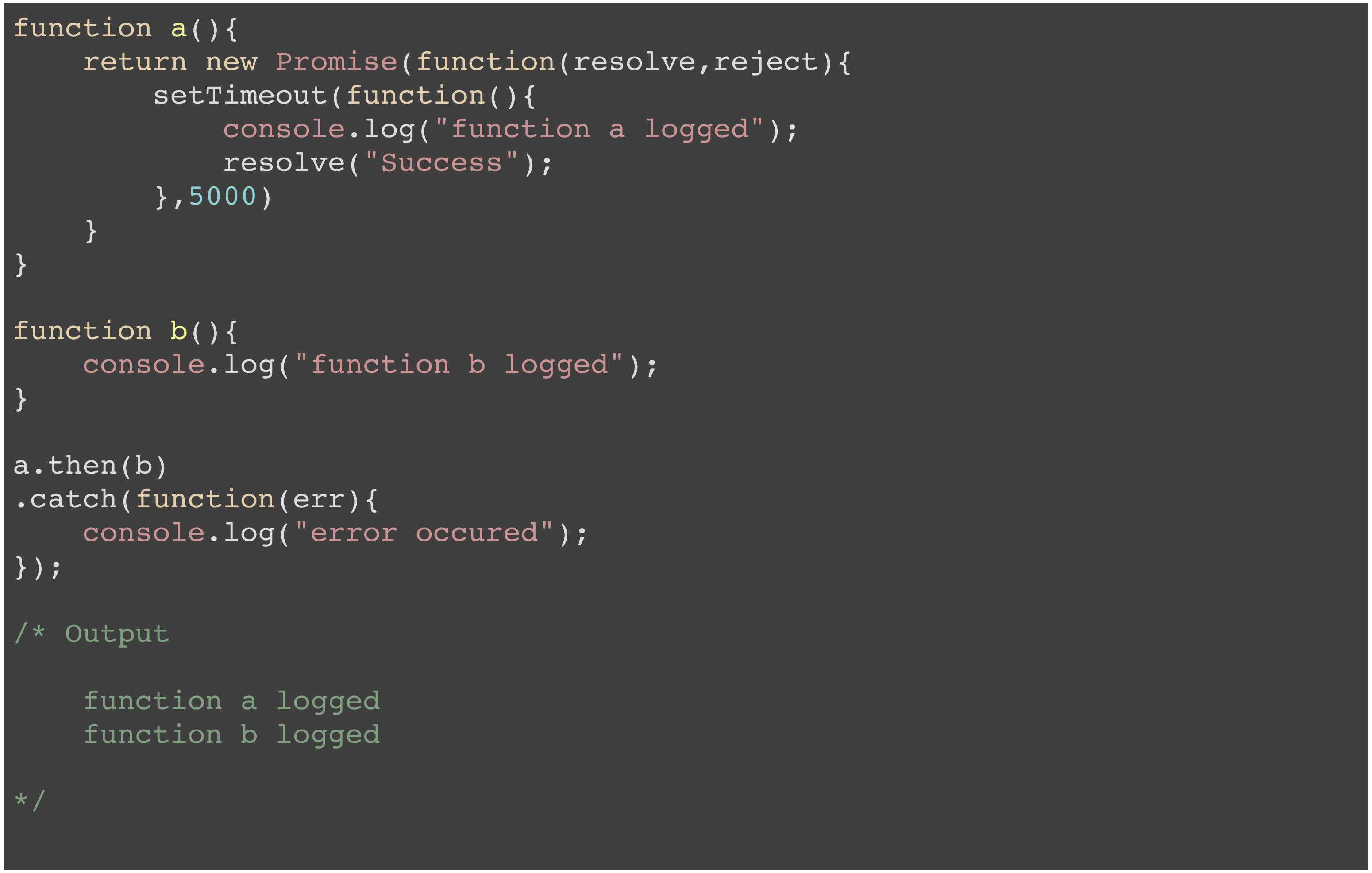
```
a(function (resultsFromA) {
  b(resultsFromA, function (resultsFromB) {
    c(resultsFromB, function (resultsFromC) {
      d(resultsFromC, function (resultsFromD) {
        e(resultsFromD, function (resultsFromE) {
          f(resultsFromE, function (resultsFromF) {
            console.log(resultsFromF);
          })
        })
      })
    })
  })
});
```

Here is our solution Promises (ES6 only)



```
var p = new Promise(function(resolve, reject) {

        // Do an long running async task and then...

        if(/* good condition */) {
                resolve('Success!');
        }
        else {
                reject('Failure!');
        }
});

p.then(function(result) {

    console.log(result); /* do something with the result */

}).catch(function(err) {

    console.log(err); /* error */
})
```

Here is our solution Promises (ES6 only)



```
function a(){
    return new Promise(function(resolve,reject){
        setTimeout(function(){
            console.log("function a logged");
            resolve("Success");
        },5000)
    }
}

function b(){
    console.log("function b logged");
}

a.then(b)
.catch(function(err){
    console.log("error occured");
});

/* Output

    function a logged
    function b logged

*/
```

Here is our solution Promises (ES6 only)

```
a.then(b)
.then(c)
.then(d)
.then(e)
.catch(function(err){
    console.log("error occured");
});
```

When do we use promises?

- When connect and make database queries
- When make call to external APIs
- When reading files
- When writing any long running code