devfest 2022

book-nav-toggle''
dden='''' fixed='''' aria-label=''Hide
Hide side navigation''

A Promise to complete understanding of JS Promises

Deepa Chaurasia
Frontend Engineer, Decimal Technologies

Google Developer Groups
[Noida]

We already Know promises

According to oxford definition, Promise is declaration or assurance that one will do something or that a particular thing will happen.

In javascript also,it is pretty much the same

But Why Promises????

devfest
2022

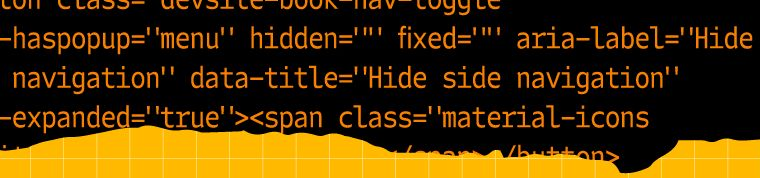
What the hell is Callback Hell??

```
const devfestToday=getCountry(country,(countryCode)⇒{
      getState(countryCode,(state)⇒{
        getCity(state,(city)⇒{
          console.log('devfest at',city);
        })
      })
    })

devfestToday('INDIA')
```

```
ton class= devsite-book-nav-toggle
-haspopup=''menu'' hidden='"' fixed='"' aria-label=''Hide
 navigation'' data-title=''Hide side navigation''
-expanded=''true''><span class=''material-icons
```

Callback hell is when we have lot of nested callbacks in order to do asynchronous task in a sequence. In big applications it get super messy and make code difficult to read or maintain.

```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```



Problems with Callback??

- Inversion of Control- You don’t have control of your functions, if any error occur in nested callbacks ,tracking down which api call failed or where error occurred is extremely difficult

- Code becomes super messy, difficult to maintain.

Developers :

devfest
2022

Promises comes to Rescue

Promises are javascript object that is used as a placeholder for the future result of asynchronous operation

In simple words it is a container for a future value

How to create a Promise??

```
let promise = new Promise(function(resolve, reject) {
    // Make an asynchronous call and either resolve or reject
});
```

Creating promise is pretty simple

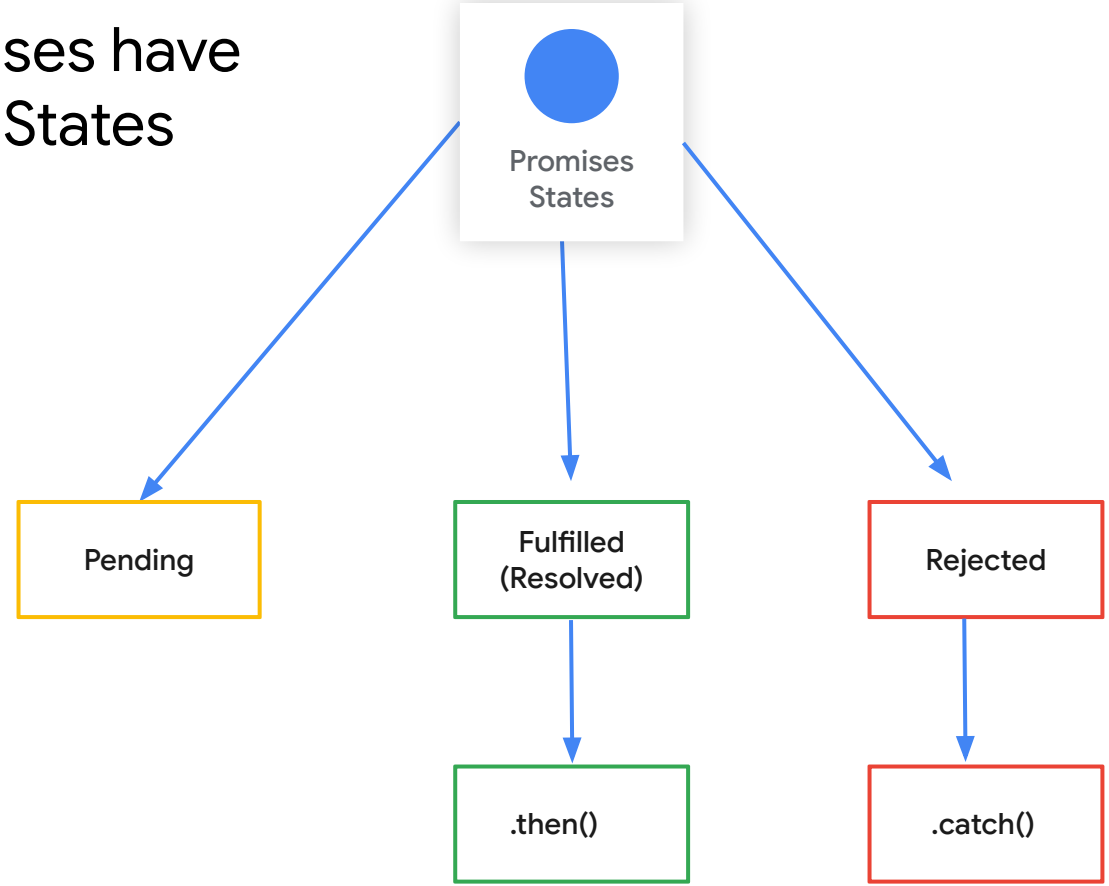
- You will create promise by creating instance of Promise using new keyword
- It will create a promise object and store in promise variable

```
▼Promise {<pending>}
  ▶[[Prototype]]: Promise
   [[PromiseState]]: "pending"
   [[PromiseResult]]: undefined
```

```
▼Promise {<fulfilled>: 'hey resolved'}
  ▶[[Prototype]]: Promise
   [[PromiseState]]: "fulfilled"
   [[PromiseResult]]: "hey resolved"
```

```
▶Uncaught (in promise) rejected
▼Promise {<rejected>: 'reject'}
  ▶[[Prototype]]: Promise
   [[PromiseState]]: "rejected"
   [[PromiseResult]]: "reject"
```

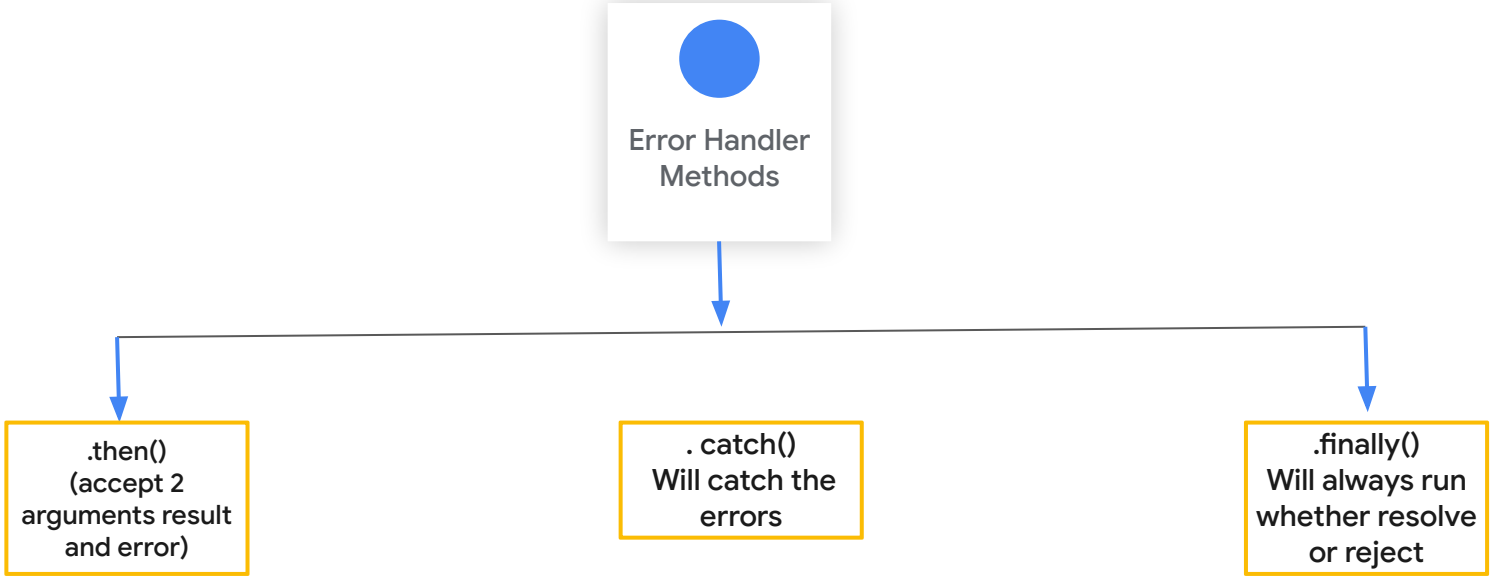
Promises have three States



Now you see the Power of Promises

- Promises give the control of function back to us.
- We can track if any api call failed in catch() handler.
- So the problem of Inversion of control resolved

Error handling in promises



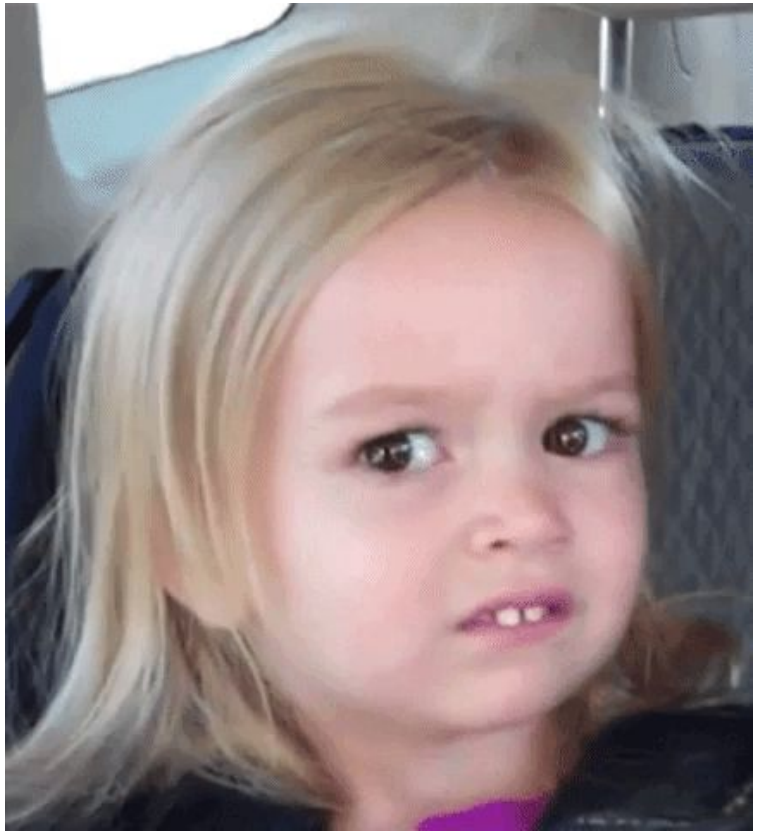
“Talk is cheap. Show me the code.”

Linus Torvalds

Let’s see code then Demo

Now you know everything about Promises

It’s Quiz Time Now

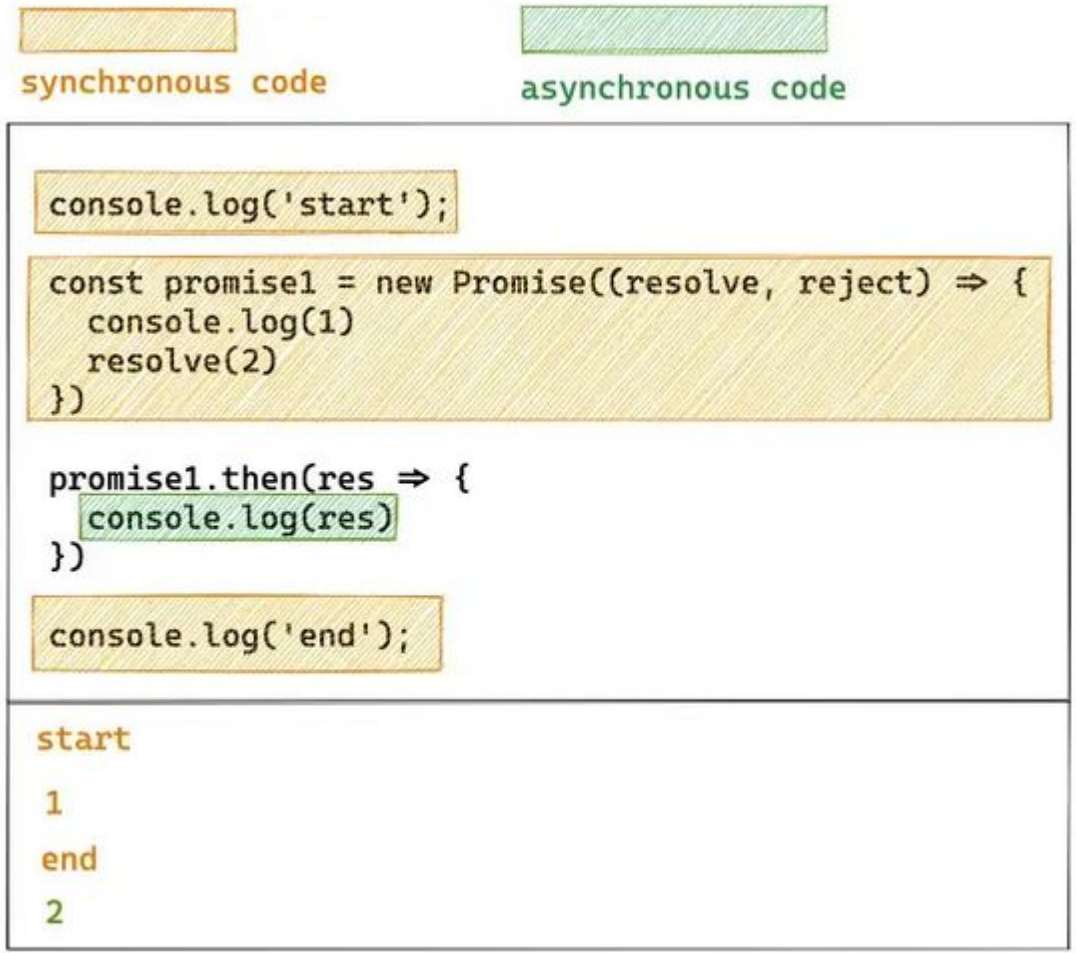


What's the output of this code?

```
console.log('start');

const promise1 = new Promise((resolve, reject) => {
  console.log(1)
  resolve(2)
  console.log(3)
})

promise1.then(res => {
  console.log(res)
})

console.log('end');
```

Output:

start,1,3,end,2



Credits-
@bytefish

What's the output of this code?

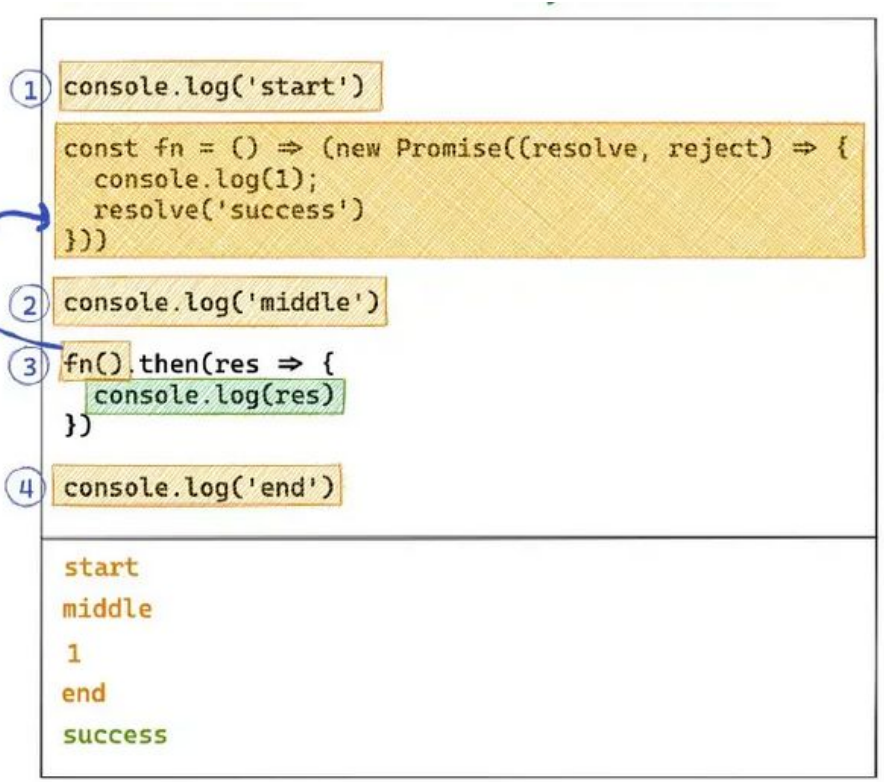
```
console.log('start')

const fn = () => (new Promise((resolve, reject) => {
  console.log(1);
  resolve('success')
}))

console.log('middle')

fn().then(res => {
  console.log(res)
})

console.log('end')
```

Output:

start , middle, 1 , end and success.

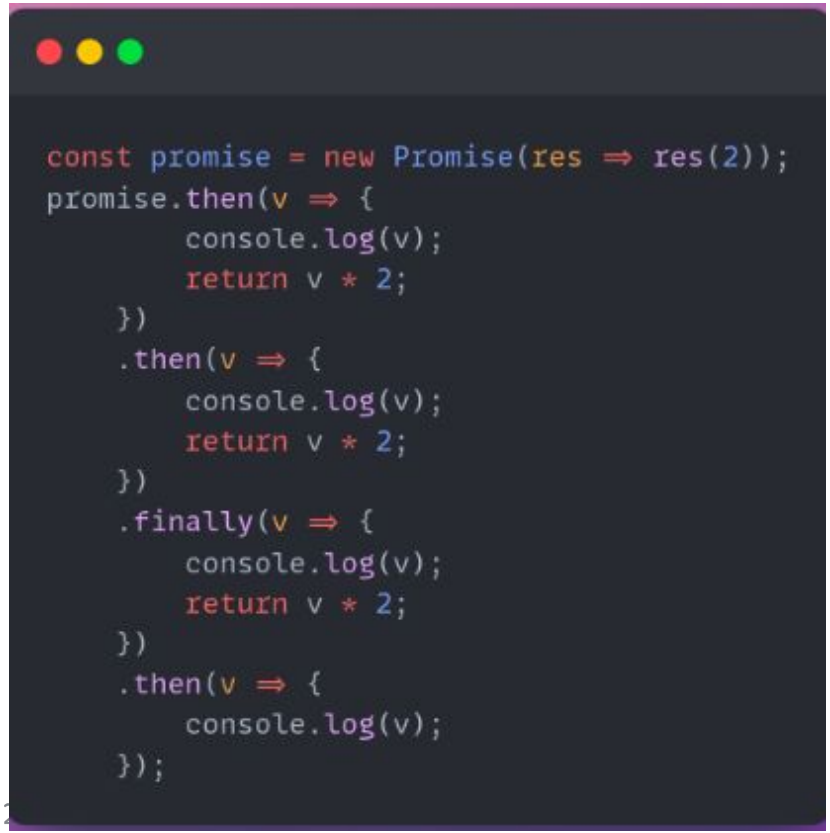
synchronous code

asynchronous code



Credits-
@bytefish

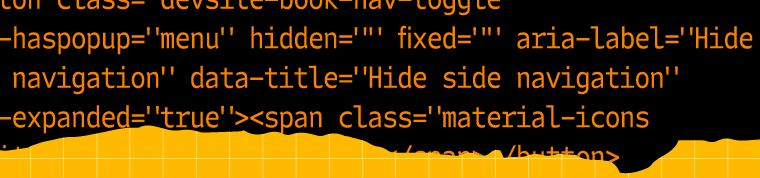
What's the output of this code?



```
const promise = new Promise(res ⇒ res(2));
promise.then(v ⇒ {
        console.log(v);
        return v * 2;
    })
    .then(v ⇒ {
        console.log(v);
        return v * 2;
    })
    .finally(v ⇒ {
        console.log(v);
        return v * 2;
    })
    .then(v ⇒ {
        console.log(v);
    });
```

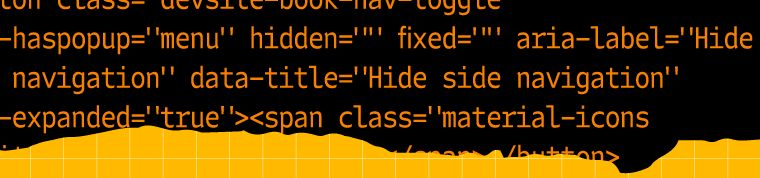
Output:

2,4,undefined,8



We have covered:

- CallBack Hell
- How to create a Promise
- Promise states and value
- Converted callbacks to promises
- Error handling



Thankyou

Connect with me on LinkedIn for Learning Resources :)

linkedIn