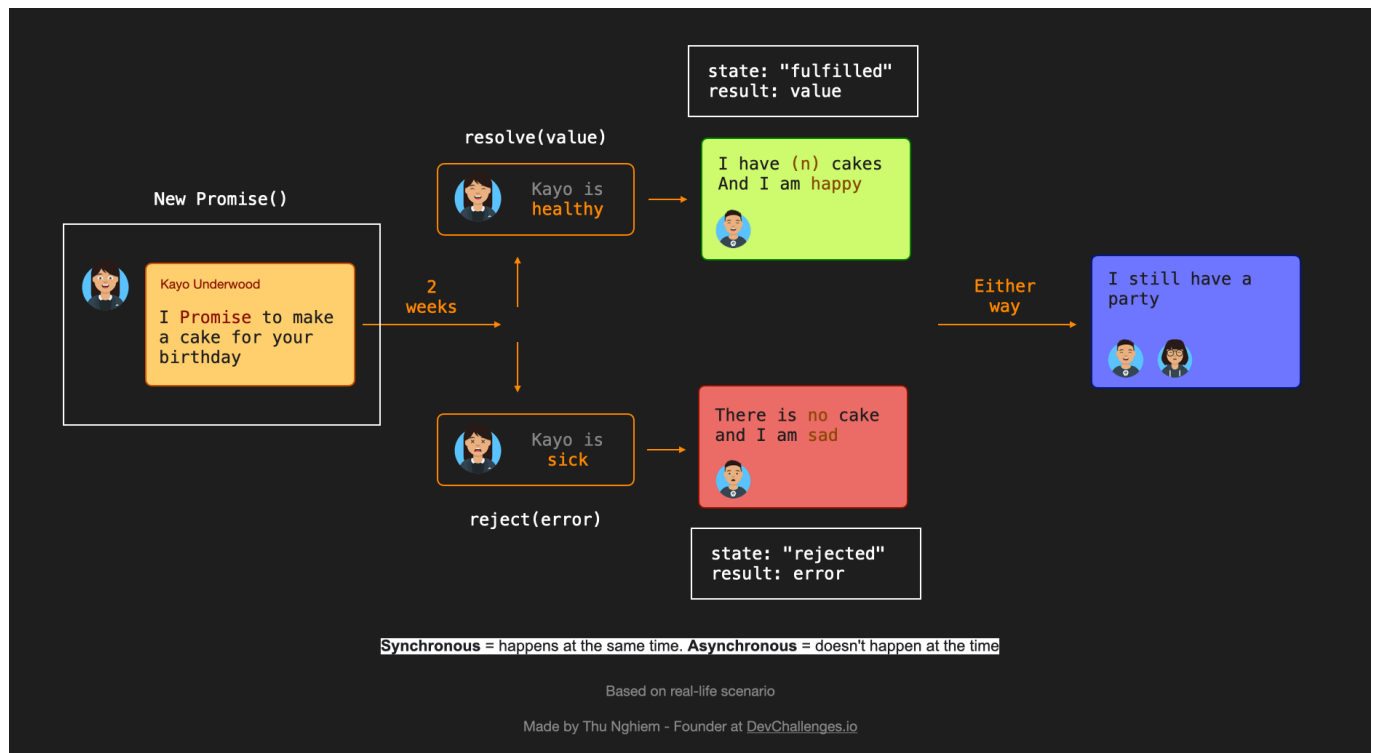


Promise



Source: <https://www.freecodecamp.org/news/learn-promise-async-await-in-20-minutes/>

AGENDA

1. Callback hell
2. Promise overview
3. Promise methods
4. Promise chaining
5. Fake API

1. Callback hell

```

1 // Callback Hell
2
3
4 a(function (resultsFromA) {
5     b(resultsFromA, function (resultsFromB) {
6         c(resultsFromB, function (resultsFromC) {
7             d(resultsFromC, function (resultsFromD) {
8                 e(resultsFromD, function (resultsFromE) {
9                     f(resultsFromE, function (resultsFromF) {
10                         console.log(resultsFromF);
11                     })
12                 })
13             })
14         })
15     })
16 });
17

```

Source: <https://medium.com/@jaybhoyar1997/avoiding-callback-hell-in-node-js-7c1c16ebd4d3>

2. Promise overview

Promise states

	Desc
pending	initial state
fulfilled	operation completed successfully
rejected	operation failed



Source: <https://scoutapm.com/blog/async-javascript>

Create a new promise

```
> const promise = new Promise();
console.log(promise);
```

```
✖ ▶ Uncaught TypeError: Promise resolver undefined is not a function
    at new Promise (<anonymous>)
    at <anonymous>:1:17
```

```
const promise = new Promise((resolve, reject) => {});
console.log(promise);
```

```
> const promise = new Promise((resolve, reject) => {});
console.log(promise);
```

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

[react_devtools_backend.js:4049](#)

Retrieve promise value

```
const promise = new Promise((resolve, reject) => {
  setTimeout(() => resolve(1), 3000);
});
```

```
promise
  .then(result => console.log(result))
  .catch(error => console.log(error));
```

```
const promise = new Promise((resolve, reject) => {
  reject(new Error('oops, something wrong!!!'))
});
```

```
promise
  .then(result => console.log(result))
  .catch(error => console.log(error));
```

GOLDEN RULE: Always handle error when using promise.

```
> const promise = new Promise((resolve, reject) => {
  reject(new Error('oops, something wrong!!!'))
});

promise.then(result => console.log(result))
< Promise {<rejected>: Error: oops, something wrong!!!
  ▶   at <anonymous>:2:10
    at new Promise (<anonymous>)
    at <an...}
```

```
✖ ▶ Uncaught (in promise) Error: oops, something wrong!!!
    at <anonymous>:2:10
    at new Promise (<anonymous>)
    at <anonymous>:1:17
```

Otherwise, you will get uncaught (in promise) error as captured above.

3. Promise methods

Static methods

#	Name	Desc
1	Promise.resolve(value)	returns a promise, value can be a promise or not
2	Promise.reject()	return a promise with rejected reason
3	Promise.all()	Wait for all promises to be resolved, or for any to be rejected.
4	Promise.allSettled()	Wait until all promises have settled (each may resolve or reject).
5	Promise.any()	Wait until any of the promises is fulfilled .
6	Promise.race()	Wait until any of the promises is fulfilled or rejected .

```
const b = new Promise((resolve) => {
  resolve('BBB');
})

const promiseA = Promise.resolve('AAA');
const promiseB = Promise.resolve(b);

Promise.all([promiseA, promiseB])
  .then(([resultA, resultB]) => {
    console.log(resultA, resultB)
  })
  .catch(error => console.log(error))
```

Instance methods

#	Name	Desc
1	then	return a new promise, happens when promise is fulfilled
2	catch	catch error if promise is rejected
3	finally	go here no matter the promise is fulfilled or rejected

```
Promise.resolve('tada')
  .then(message => console.log(message))
  .catch(error => console.log(error))
  .finally(() => {
    // usually we hide loading here
  })
```

4. Promise chaining

```
Promise.resolve(5)
  .then(n => n * 2)
  .then(n => Promise.resolve(n * 2))
  .then(n => {
    const finalNumber = n + 10;
    console.log(finalNumber);

    return finalNumber;
  })
  .catch(error => console.log(error));
```

```
fetch('https://js-post-api.herokuapp.com/api/students?_page=1')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.log(error));
```

5. Fake API

- Sometimes, the API from Backend is not ready, you need to fake it your self first to not block your works.

```
// studentApi.js
const studentApi = {
  getAll() {
    return new Promise((resolve) => {
      setTimeout(() => {
        resolve({
          data: [
            { id: 1, name: 'Alice' },
            { id: 2, name: 'Bob' },
          ],
          pagination: {
            _total: 2,
            _page: 1,
            _limit: 10,
          }
        })
      }, 1000);
    })
  }
}
```

```
// app.js
studentApi.getAll()
  .then(response => console.log(response))
  .catch(error => console.log(error));
```

Tham khảo

- <https://scoutapm.com/blog/async-javascript>
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Khoá học Javascript cho người mới bắt đầu 2021 🎉

- Tác giả: **Hậu Nguyễn** - Founder Easy Frontend
- Khoá học chỉ được published trên Udemy, không thông qua trung gian.
- Khoá học không bán dạng videos upload trên Google Drive hay bất cứ hình thức nào tương tự.
- Khoá học có nhóm discord để hỗ trợ trong quá trình học tập.

☎️ Liên hệ tác giả để được hỗ trợ:

- ✅ Facebook: <https://www.facebook.com/nvhauesmn/>
- ✅ Fanpage: <https://www.facebook.com/learn.easyfrontend>
- ✅ Youtube Channel: <https://www.youtube.com/easyfrontend>