

I Promise you!

Promises, promises and promises 🤖 👽



Cheeseburgers for everyone 🍔

1. We **order** a burger
2. We **pay**
3. The burger man give us a **ticket**
4. We **send** text messages
5. We **think** in a our hamburger
6. We still waiting...

We treat out ticket as a `placeholder` for a cheeseburger.

Our burger is a **Future** value!! Can the promise for a burger fail??

I **Promise** you (From Eric Elliot perspective)

A promise is an object that may produce a single value some time in the future: either a resolved value, or a reason that it's not resolved (e.g. a network error occurred).

Promises code

Basic Syntax

```
new Promise(function (resolve, reject) {});
```

Notes:

1. We call `resolve` if we have success
2. We call `reject` if we have an error

The thre...**four** Promise states

1. **Fullfiled**. Eeverything goes well
2. **Rejected**. Everything goes bad
3. **Pending**. We don't know yet
4. **Settled**. Not pending (it has been resolved or rejected)

Promises in real life 🤔

Imagine you are a kid. Your dad promises you that he will buy you a new toy next week

That is a promise.

- Pending: You don't know if you will get the toy
- Fulfilled: Dad is happy and he will get you a toy
- Rejected: Your dad is not happy, he withholds the toy

Why to do Promises?

Let's remember callbacks:

```
function add(x, y, cb) {  
  setTimeout(() => cb(x + y), 1000);  
}  
  
add(0, 2, function (sum) {  
  add(sum, 2, function (sum2) {  
    add(sum2, 2, function (sum3) {  
      // ...  
    }); // End last add  
  }); // End mid add  
}); // End first add
```

THE CALLBACK HELL!!!! 🐱

Why to do Promises? x2

Oh boy!!

```
doSomething()  
  .then(doSomethingElse)  
  .catch(handleError)  
  .then(doMoreStuff)  
  .then(doFinalThing)  
  .catch(handleAnotherError);
```

Looks cleaner and more maintainable.

Sooo **Then**, What's next?

I create a promise, how to use/access to its value?

```
promiseFn().then(/** I will be executed */);
```

The `then()` function will be executed when the promise is no longer pending.

Can receive two params:

```
doSomething()  
  .then(onFullfiled, onRejected);
```

1. `onFullfiled`, if ok.
2. `onRejected`, if not ok.

Catching everything!

In practice is not common to use `onFullfiled` and `onRejected` functions in the same `then`.

Instead we use...

```
doSomething()  
  .then(onFullfiled)  
  .catch(onRejected);
```

Can we have multiple `catch()`?

Parallelism using promises

But we thought that paralellism and JS were not friends?!!!

Its not paralellism its concurrency!!

```
const objs = [{...}, {...}];  
const promises = objs.map(asyncFn); // An array of promises  
  
Promise.all(promises)  
  .then(function(arrayOfResults) {  
    // When all promises will be resolved  
  })  
  .catch(function(err){  
    // If any of the promises fails  
  } )
```

All the promises are being executed at the same moment!

GO TO THE CODE!! 🤔