# Shorter Syntax



**Arrow**
Functions

```
() => {}
```

```
// ES5
var add = function (num1, num2) {
    return num1 + num2;
}

// ES6
var add = (num1, num2) => num1 + num2
```

Lexical This & (Arrow) => {Functions}



# Arrow functions =>

`this` is *undefined*

```
this.x = 'yes';
var y = function(){
  "use strict";
  console.log(this.x);
};
y(); // TypeError…
```

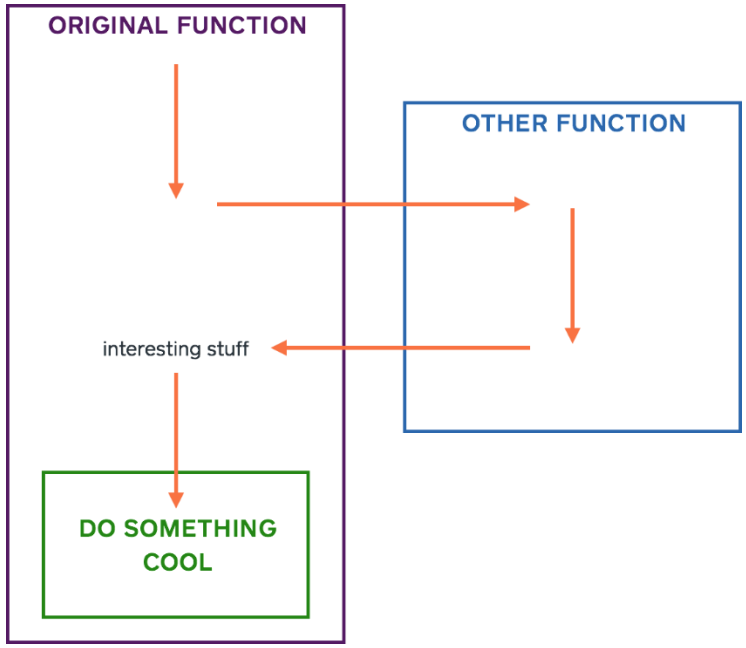
`this` from outer scope

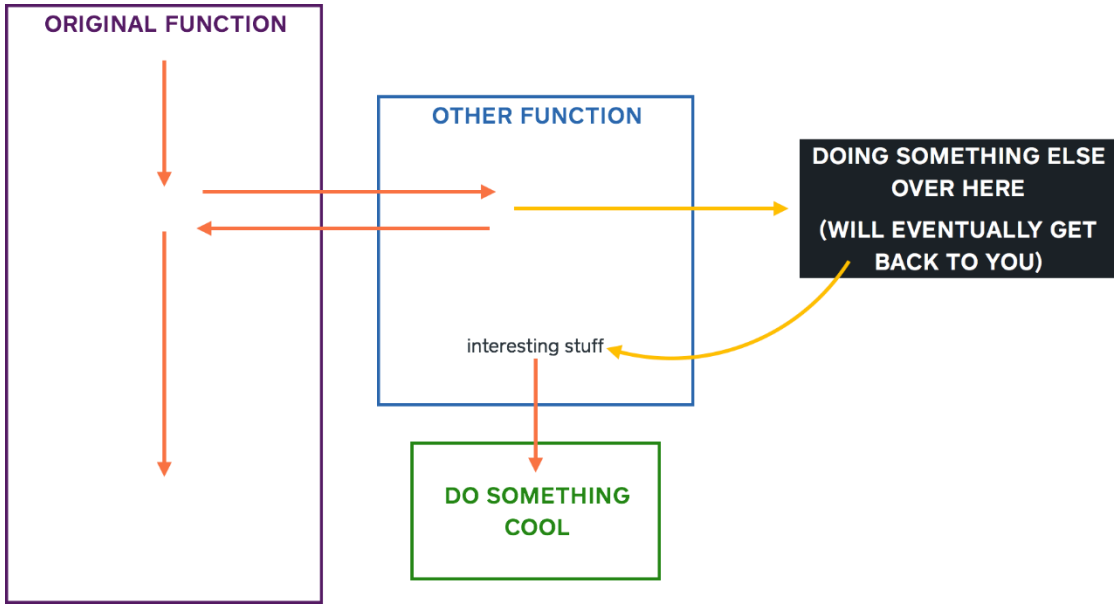
```
this.x = 'yes';
let y = () => {
  "use strict";
  console.log(this.x);
};
y(); // 'yes'
```

No need for var that = this, `bind` with arrow functions

# DEMO

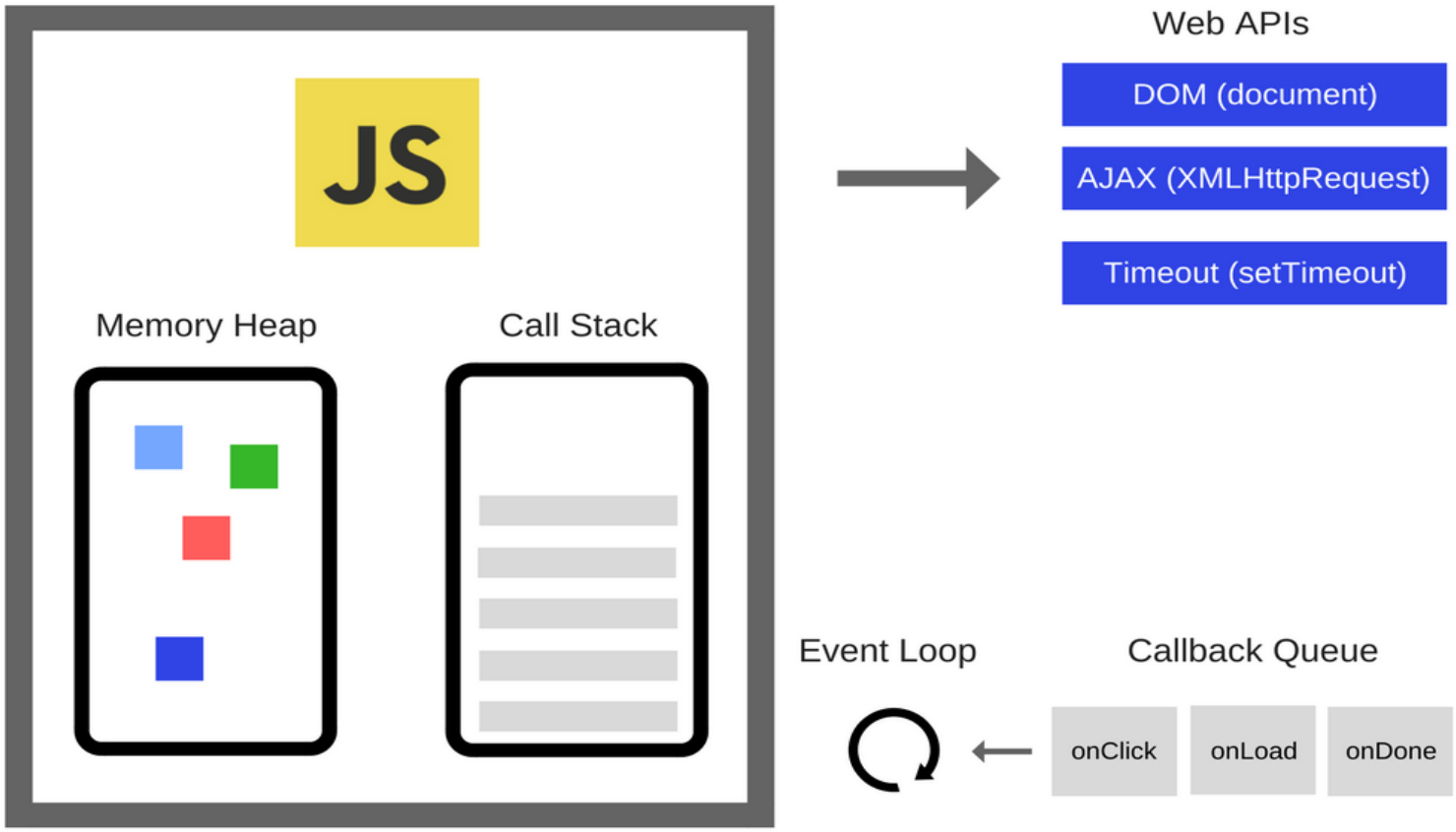
# Asynchronous functions



# Callbacks everywhere!



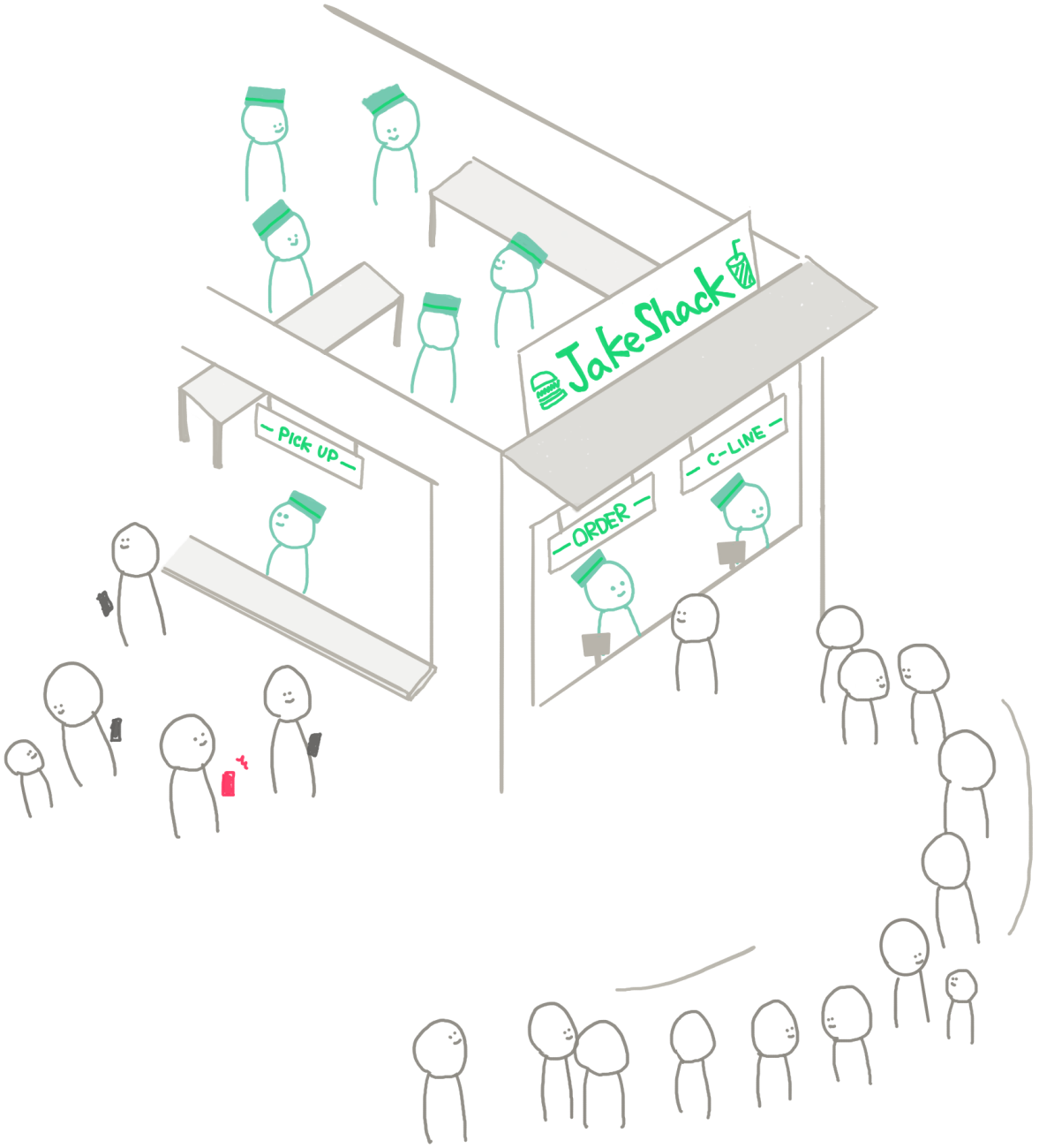


# DEMO

# The Promise of a BURGER PARTY

written by @kasamari

A quest to understanding JavaScript Promise



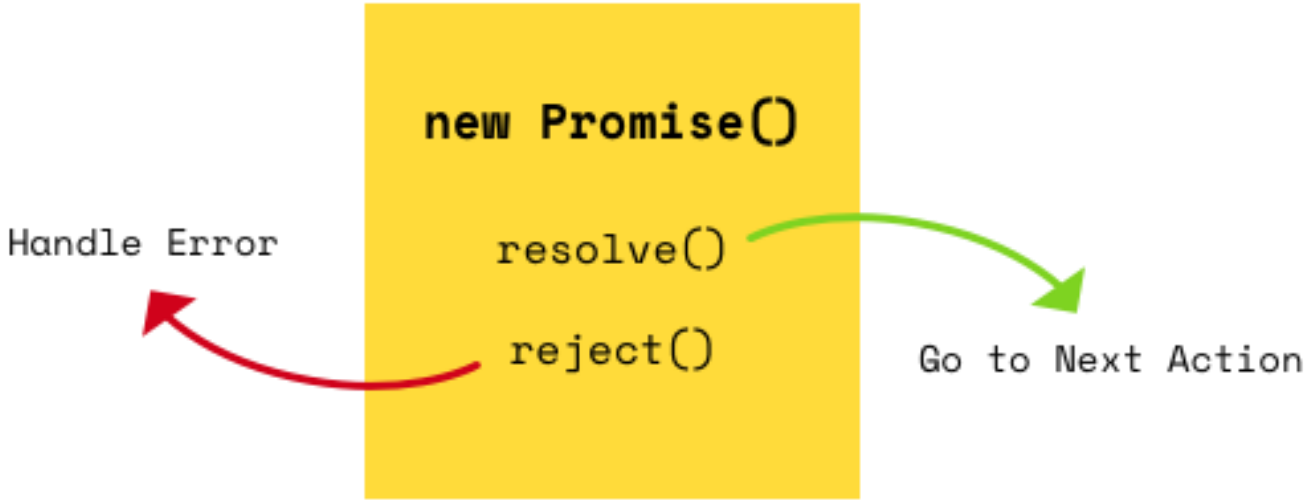
# Promise

## What is Promise?

- A promise is an object that represents the return value or the thrown exception that the function may eventually provide.
- In other words, a promise represents a value that is not yet known. *A promise is an **asynchronous value**.*
- The core idea behind promises is that a promise represents the result of an asynchronous operation.
- A Promise has 3 possible states
  - Pending
  - Fulfilled
  - Rejected

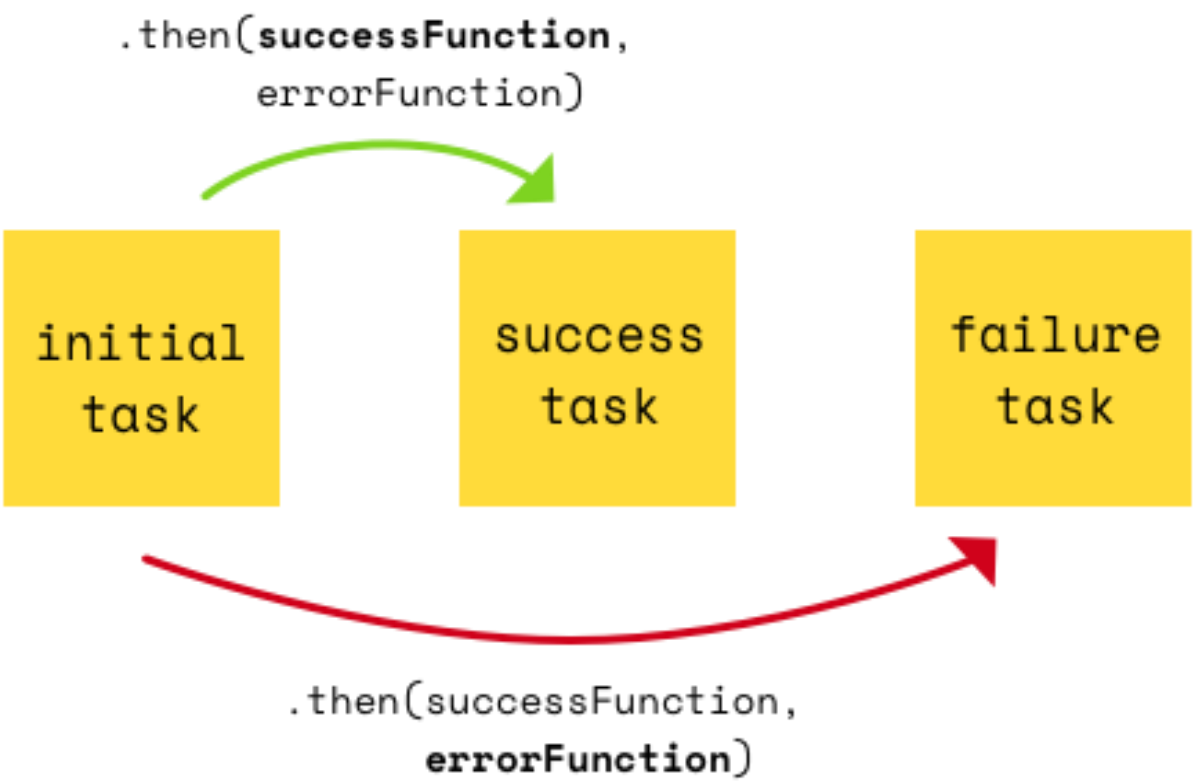
# Promise





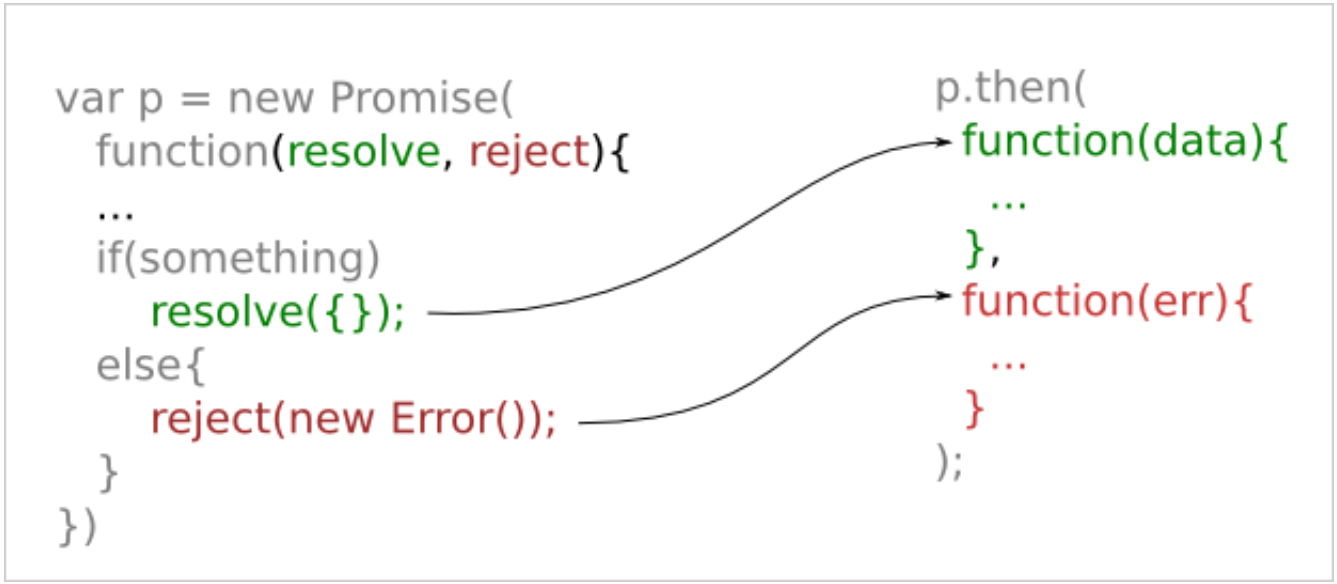
# Promise:then





# Promise




var p = new Promise(
function(resolve, reject){
...
if(something)
resolve({});
else{
reject(new Error());
}
})
p.then(
function(data){
...
},
function(err){
...
}
);

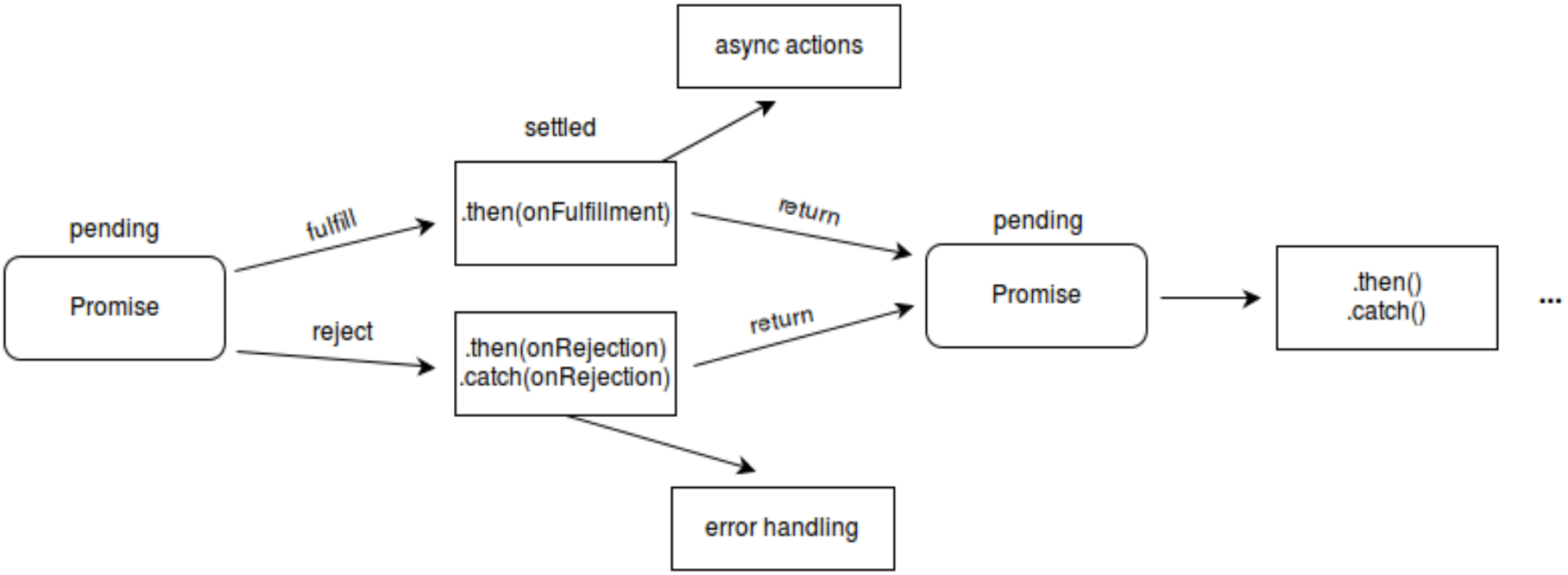
# DEMO

# Callback Hell

```
func1(param, function(err, res) {
  func2(param, function(err, res) {
    func3(param, function(err, res) {
      func4(param, function(err, res) {
        func5(param, function(err, res) {
          func6(param, function(err, res) {
            func7(param, function(err, res) {
              func8(param, function(err, res) {
                func9(param, function(err, res) {
                  // Do something...
                });
              });
            });
          });
        });
      });
    });
  });
});
```

# Chaining the promises





# DEMO