How to return many Promises in a loop and wait for them all to do other stuff

Asked 4 years, 3 months ago Active 8 days ago Viewed 41k times

48

28

I have a loop which calls a method that does stuff asynchronously. This loop can call the method many times. After this loop, I have another loop that needs to be executed only when all the asynchronous stuff is done. So this illustrates what I want:

```
for (i = 0; i < 5; i++) {
    doSomeAsyncStuff();
}

for (i = 0; i < 5; i++) {
    doSomeStuffOnlyWhenTheAsyncStuffIsFinish();
}
```

I'm not very familiar with promises, so could anyone help me to achieve this?

This is how my `doSomeAsyncStuff()` behaves:

```
function doSomeAsyncStuff() {
    var editor = generateCKEditor();
    editor.on('instanceReady', function(evt) {
        doSomeStuff();
        // There should be the resolve() of the promises I think.
    })
}
```

Maybe I have to do something like this:

```
function doSomeAsyncStuff() {
    var editor = generateCKEditor();
    return new Promise(function(resolve,refuse) {
        editor.on('instanceReady', function(evt) {
            doSomeStuff();
            resolve(true);
        });
    });
}
```

But I'm not sure of the syntax.

javascript promise ecmascript-6 es6-promise

edited Aug 30 at 10:09 asked Jul 15 '15 at 9:36

AuxTaco
3,677 1 5 23

Ganbin
705 1 6 15

Are you in control of the asynchronous calls? Do they already return promises, or can you make them return promises? – T.J. Crowder Jul 15 '15 at 9:42

What exactly is the sequence? Do you need to call the other functions after *all* the previous async ones are finished? Or do you just need to call a function after each of the async are finished? – Sosdoc Jul 15 '15 at 9:43

For now the first function doesn't return promises. That I have to implement. I want to edit my message to add some details of the workflow of my functions. And yes I need that all the stuff of the first loop to be finish before start to execute the stuff in the second loop. – Ganbin Jul 15 '15 at 9:44

@T.J.Crowder Yep thanks, I write to fast ^^ – Ganbin Jul 15 '15 at 9:58

1 Re your edit: *"Maybe I have to do something like that"* Yup, very much like that, except there's no `s` at the end of `Promise`. – T.J. Crowder Jul 15 '15 at 10:10

2 Answers

112

You can use `Promise.all` (spec, MDN) for that: It accepts a bunch of individual promises and gives you back a single promise that is resolved when all of the ones you gave it are resolved, or rejected when any of them is rejected.

So if you make `doSomeAsyncStuff` return a promise, then:

```
var promises = [];

for(i=0;i<5;i+){
    promises.push(doSomeAsyncStuff());
}

Promise.all(promises)
    .then(() => {
        for(i=0;i<5;i+){
            doSomeStuffOnlyWhenTheAsyncStuffIsFinish();
        }
    })
    .catch((e) => {
        // handle errors here
    });
```

Axel Rauschmayer has a good article on promises here.

Here's an example - live copy on Babel's REPL:

```
function doSomethingAsync(value) {
     return new Promise((resolve) => {
       setTimeout(() => {
         console.log("Resolving " + value);
         resolve(value);
       }, Math.floor(Math.random() * 1000));
     });
   }

   function test() {
```

```
  let i;
  let promises = [];

  for (i = 0; i < 5; ++i) {
    promises.push(doSomethingAsync(i));
  }

  Promise.all(promises)
      .then((results) => {
        console.log("All done", results);
      })
      .catch((e) => {
          // Handle errors here
      });
}

test();
```

Run code snippet | Copy snippet to answer | Expand snippet

(Didn't bother with `.catch` on that, but you do want `.catch` on your real-world ones, as shown earlier.)

Sample output (because of the `Math.random`, what finishes first may vary):

```
Resolving 3
Resolving 2
Resolving 1
Resolving 4
Resolving 0
All done [0,1,2,3,4]
```

edited Nov 14 '17 at 17:09

Julian E.
4,032 5 26 45

answered Jul 15 '15 at 9:45

T.J. Crowder
747k 136 1359 1416

Ok thanks I try this now and I come with feedback in few minutes. – Ganbin Jul 15 '15 at 10:00

11 Wow, thanks a lot, now I understand much more the promises.I read a lot about promises, but until we need to use them in real code, we don't really understand all the mechanisms. Now I get it better and I can start to write cool stuff, thanks to you. – Ganbin Jul 15 '15 at 10:22

Really helpful, thank you! – Lucy Sep 14 '17 at 23:16

1 Working like a charm! Many thanks to you. Finally, i understood Promises. – Sasi Rekha Apr 10 at 11:45

1 @user1063287 - You can do that if the code is in a context where `await` is allowed. At the moment, the only place you can use `await` is inside an `async` function. (At some point you'll also be able to use it at the top level of modules.) – T.J. Crowder Jul 24 at 9:37

A reusable function works nicely for this pattern:

0

```
function awaitAll(count, asyncFn) {
  const promises = [];

  for (i = 0; i < count; ++i) {
    promises.push(asyncFn());
  }

  return Promise.all(promises);
}
```

OP example:

```
awaitAll(5, doSomeAsyncStuff)
  .then(results => console.log('doSomeStuffOnlyWhenTheAsyncStuffIsFinished', results))
  .catch(e => console.error(e));
```

A related pattern, is iterating over an array and performing an async operation on each item:

```
function awaitAll(list, asyncFn) {
  const promises = [];

  list.forEach(x => {
    promises.push(asyncFn(x));
  });

  return Promise.all(promises);
}
```

Example:

```
const books = [{ id: 1, name: 'foo' }, { id: 2, name: 'bar' }];

function doSomeAsyncStuffWith(book) {
  return Promise.resolve(book.name);
}

awaitAll(books, doSomeAsyncStuffWith)
  .then(results => console.log('doSomeStuffOnlyWhenTheAsyncStuffIsFinished', results))
  .catch(e => console.error(e));
```

answered Oct 23 at 22:17

2Toad

12.4k 6 32 34