Waiting for Promises in a Loop

JANUARY 22, 2016

Need to run a series of HTTP calls, and wait for them all to complete? Use `$q.all`.

This won't work:

```
for(var i = 0; i < 5; i++) {
    $http.get('/data' + i);
}
// At this point, all the requests will have fired...
// But probabaly, none of them have finished
```

Do this instead:

```
var promises = [];
for(var i = 0; i < 5; i++) {
    var promise = $http.get('/data' + i);
    promises.push(promise);
}
$q.all(promises).then(doSomethingAfterAllRequests);
```

Run the promsies in order (not in parallel)

When you queue up promises like above, they all start at the same time. But what if you want them to run in the order you called them?

You can build up a chain of promises:

```
var chain = $q.when();
for(var i = 0; i < 5; i++) {
    chain = chain.then(function() {
        return $http.get('/data' + i);
    });
}

// This is the same as:
chain = $q.when();
chain.then(function() {
            return $http.get('/data1');
        }).then(function() {
            return $http.get('/data2');
        }).then(function() {
            return $http.get('/data3');
        }).then(function() {
            return $http.get('/data4');
        }).then(function() {
            return $http.get('/data5');
        });
```

`$q.when` is used to kick off the chain with a resolved promise.

Moses Ndeda 'Subaru' • a month ago
This helped me a lot. Thanks.
• Reply • Share ›

Ilche Ivanovski • 10 months ago
var responses = [];

```
var rejections = [];

arrays.forEach(function (userArray, index) {

var body = {
users: userArray
};

//here we create parallel promises
chain = chain
.then(function (r) {
if (typeof r !== 'undefined') {
responses.push(r.data.message);
}
//timeout to each parallel promises
return $timeout(function () {
}, 5000).then(function () {
return $http.post(serviceBase + 'api/admin/aaaaaaaaaaa', body);
```

see more

• Reply • Share ›

Ilche Ivanovski • 10 months ago

Do you have any idea how to return callback for all chained parallel promises called in loop?

```
var chain = $q.when();
for(var i = 0; i < 5; i++) {
chain = chain.then(function() {
return $http.get('/data' + i);
});
}
```

In this case I want to resolve or reject 5 times...

• Reply • Share ›

Dave Ceddia Mod ➚ Ilche Ivanovski • 10 months ago

You want the individual promises? Maybe you could push them on an array as you chain them, and then return the array.

• Reply • Share ›

Ilche Ivanovski ➚ Dave Ceddia • 10 months ago

thx! I manage to do that with $q.defer()..

• Reply • Share ›

tina • a year ago

I'm writing node.js library and it gives me error when I use $q. Is there any package that I need to import or something?

• Reply • Share ›

Dave Ceddia Mod ➚ tina • a year ago

$q is built into Angular, but if you're writing node you'd want to use another library, or the built-in promises.

You can use "new Promise((resolve, reject) => {...})", "Promise.resolve", and "Promise.reject" instead of $q. Check out this article: https://strongloop.com/stro...

1 ^ v • Reply • Share ›

disqus_MEjKNsLwgg • 2 years ago

This will not work because of closure on "i" and will produce "/data5" for all calls (at least this is how it behaves in my usecase). I modified your example to handle this uising IIFE to scope "i":

```
(function(index) {
$http.get('/data' + index);
})(i);
```

Thanks for this article which helped me!

^ v • Reply • Share ›

saif a ➚ disqus_MEjKNsLwgg • a year ago

You are correct. But could you please elaborate your modified code including the loop? I could not get it.

1 ^ v • Reply • Share ›

disqus_MEjKNsLwgg ➚ saif a • a year ago

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

    (function(index) {
         chain = chain.then(function() {
                return $http.get('/data' + index);
          });

    })(i);
}
```

IIFE is concept I got from AngularJS. Basically it creates a function and runs it immediately.
When you do it "i" variable is copied to new function and it's value from the loop iteration moment is remembered. Otherwise all promises get value from last iteration of the loop (as they run asynchronously after loop has ended).

3 ^ v • Reply • Share ›

Diogo Fernandes • 2 years ago

Thanks!

^ v • Reply • Share ›

Ravi • 2 years ago

how call $q.all inside $q.all. The first $q.all gets data and based on that second $q.all works. Nesting one inside the other does not work.

^ v • Reply • Share ›

Dave Ceddia Mod ➚ Ravi • 2 years ago

I'd think of it this way:
- kick off a bunch of promises to get data

- kick off a bunch of promises to get data
- wait for them to finish with $q.all
- THEN, kick off more promises
- wait for that second batch to finish with $q.all
- THEN do whatever you need to do.

Nesting them seems like it should work - I think I'd approach it like this:

```
var firstWavePromises = [];
firstWavePromises.push($http.get(...));
firstWavePromises.push($http.get(...));
$q.all(firstWavePromises).then(function(firstWaveCompleted) {
  // firstWaveCompleted is an array of the results,
  // in the order that the promises were pushed
  var secondWavePromises = [];
  secondWavePromises.push($http.get(...));
  secondWavePromises.push($http.get(...));
  $q.all(secondWavePromises).then(function(secondWaveCompleted) {
    // All promises are done. Do the work you need to do here.
  });
});
```


Ladna Meke • 2 years ago

Nice. Read somewhere that we can use Array.prototype.every() to run each http get and return true whenever a valid response returns

Previous Next