

# Angular Js and google api client.js (gapi)

Asked 11 years, 2 months ago   Modified 7 years, 4 months ago   Viewed 25k times

 Part of [Google Cloud](#) Collective



It took me one day to make it works so I think my experience may be useful from someone. And maybe some others will find improvement.

30



So I start angularJS two days ago. And I want it works with Google Cloud Endpoints to create a backend interface. Here comes the trouble for me.



The javascript client for gapi comes with asynchronous loading, so angular initialization will crash having gapi undefined.



**So you need to bootstrap angular when gapi is initialized:**

1. remove ng-app="myApp"
2. Add `<script src="https://apis.google.com/js/client.js?onload=googleOnLoadCallback"></script>`
3. Add the callback:

```
function googleOnLoadCallback(){
  var apisToLoad = 1; // must match number of calls to gapi.client.load()
  var gCallback = function() {
    if (--apisToLoad == 0) {
      //Manual bootstrapping of the application
      var $injector = angular.bootstrap(document, ['myApp']);
      console.log('Angular bootstrap complete ' + gapi);
    }
  };
  gapi.client.load('helloWorld', 'v1', gCallback, '/' + window.location.host
+ '/_ah/api');
}
```

**Feel good but how about a call ?**

So here is the controller:

```
angular.module('myApp.controllers', []).
  .controller('MyCtrl', ['$scope', 'helloWorldService',
    function($scope, greetingsService) {
      helloWorldService.loadData($scope);
    }]);
```

And here is the service:

```
angular.module('myApp.services', [])
service('helloWorldService', [function() {
  this.loadData = function($scope) {
```

```
//Async call to google service
gapi.client.helloWorld.greetings.listGreeting().execute(
  function(resp) {
    if (!resp.code) {
      console.debug(resp);
      $scope.greetings = resp.items;
      // Because it's a callback,
      // we need to notify angular of the data refresh...
      $scope.$apply();
    }
  });
};
})();
```

And magically your page updates thanks to angular.

Feel free to mark anywhere I go wrong.



javascript

angularjs

google-app-engine

google-cloud-endpoints

google-api-js-client

Share Follow

edited Nov 11, 2014 at 17:59



willma

7,533 ● 2 ● 32 ● 45

asked Oct 16, 2013 at 9:13



Samuel

369 ● 1 ● 4 ● 9

Hi @Samuel its not working . i am stucked in this problem. can you make it more detailed. – [ARV](#)  
Dec 10, 2013 at 6:56 ✎

Thank you for posting this - helped me out a lot :) – [Xander](#) Nov 2, 2014 at 0:29

Just an optimization note: `$scope.$digest` would be better used here since it starts a digest cycle on just the current `$scope` object; `$scope.$apply` is more costly because it starts an app-wide digest cycle. – [hyang123](#) Apr 29, 2016 at 20:36

Do we have a turnaround for angular2? – [Rahul Sharma](#) Jan 23, 2018 at 10:15

8 Answers

Sorted by: Highest score (default)



25



Rather than bootstrapping or setting a timeout, it's most efficient to let Angular load before/while you're making the server requests. I followed the advice described in [AngularJS + Cloud Endpoints: A Recipe for Building Modern Web Applications](#), which does the following.

Keep your `ng-app` directive as usual (no bootstrapping)



```
<html ng-app="myApp">
<head>
  <script src="angular.js" type="text/javascript"></script>
  <script src="app.js" type="text/javascript"></script>
  <script src="https://apis.google.com/js/client.js?onload=init"></script>
```

```
</head>
<body ng-show="backendReady">
```

Create a global variable for the GAPI callback function anywhere in your JS

```
var app = angular.module('myApp', []);

var init = function() {
  window.initGapi();
}

app.controller('MainController', function($scope, $window, gapiService) {
  var postInitiation = function() {
    // load all your assets
  }
  $window.initGapi = function() {
    gapiService.initGapi(postInitiation);
  }
});

app.service('gapiService', function() {
  this.initGapi = function(postInitiation) {
    gapi.client.load('helloWorld', 'v1', postInitiation, restURL);
  }
});
```

From link above:

The reason why you would not want to execute the initialization in the first init() method is so you can put as much of the code as possible in the AngularJS world, such as controllers, services and directives. As a result, you can harness the full power of AngularJS and have all your unit tests, integrations tests, and so forth.

This may seem like a roundabout way of doing things, but it optimizes for speed, testability, *and* separation of concerns.

Share Follow

edited Apr 21, 2015 at 17:37

answered Nov 25, 2014 at 0:11



willlma

7,533 ● 2 ● 32 ● 45

@CadeThacker Your edit was rejected, but it was correct, so I replicated it. Thanks. – willlma Apr 21, 2015 at 18:43

what about infinit loop [stackoverflow.com/questions/28732667/...](http://stackoverflow.com/questions/28732667/...) ? – mpgn Jun 3, 2015 at 14:37

1 See my answer there. TLDR: use different function names for `init` and `initGapi` – willlma Jun 3, 2015 at 15:20

Hi Willlma, I am trying to use this method. But i met an error saying window.initGapi is not defined. I define the method in a BaseController which tied to the Body. – Tianxiang Zhang Jul 17, 2015 at 17:05

Hmm, but site why unit would die before the controller. Make a question with your code and I'll have a look. – [willma](#) Jul 17, 2015 at 20:51

Nice post and thanks! This approach worked for me. It might matter what order that the code appears in your index.html file. It did not work for me until I had things in this order.

6

```
...
<script>
  function googleOnLoadCallback(){
    alert('googleOnLoadCallback called');
    var apisToLoad = 1; // must match number of calls to gapi.client.load()
    var gCallback = function() {
      if (--apisToLoad == 0) {
        //Manual bootstrapping of the application
        var $injector = angular.bootstrap(document, ["myApp"]);
        console.log("myApp bootstrap complete " + gapi);
      }
    };
    gapi.client.setApiKey("my_client_id");
    gapi.client.load("translate", "v2", gCallback);

  }
</script>
<!-- See https://developers.google.com/api-client-library/javascript/samples/samples -->
<script src="https://apis.google.com/js/client.js?onload=googleOnLoadCallback">
</script>
</head>
```

Share Follow

answered Mar 22, 2014 at 17:59



[davedonohue](#)

312 ● 5 ● 14

Although pretty much on progress maybe also worth to mention [angular-googleapi](#), which wraps nicely some Google Calendar and Google Plus API calls and easy extendable.

4

You'd need to add this bit to your controller when checking for authorisation:

```
$scope.authenticated = false;

$scope.$on("google:authenticated", function(){
  $scope.authenticated = true;
  $scope.$on('googleCalendar:loaded', function(){
    # work your magic here
    # $scope.calendars = googleCalendar.listCalendars();
    # $scope.$apply();
  });
});

function checkAuth() {
  setTimeout(function(){
    gapi.auth === undefined ? checkAuth() : googleLogin.checkAuth();
  }, 20);
}
```

```
checkAuth();
```

Share Follow

edited Jun 10, 2014 at 9:22

answered Jun 10, 2014 at 9:15



fer

1,016 ● 1 ● 10 ● 20

Not sure that I fully understand. But why we can't append checkAuth() to the provider? It seems to be more comfortable. Thank you for your advice. – [oshliaer](#) Jun 9, 2015 at 10:27



3



I wrote a simple directive to load the google map API asynchronously :

```
// js/directives/gmapAsync.js

(function(){
  'use strict';

  angular.module('app').directive('gmapAsync',
    ['$window', '$rootScope', gmapAsync]
  );

  function gmapAsync($window, $rootScope){

    var gmapScript = $window.document.createElement('script');

    $window.onGmapScriptLoaded = function(){
      console.log('google maps script loaded');

      $rootScope.gmapApiLoaded = true;
      $rootScope.$broadcast('gmap.api.loaded');
    };

    return {
      restrict: 'A',
      transclude: false,
      scope: false,
      link: function(scope, element, attributes){

        if (navigator.onLine) {
          appendScript();
        } else {
          $window.addEventListener('online', appendScript);
        }

        function appendScript(){
          gmapScript.type = 'text/javascript';
          gmapScript.src = 'https://maps.googleapis.com/maps/api/js?v=3.exp&' + 'callback=onGmapScriptLoaded';
          $window.document.body.appendChild(gmapScript);
        }
      }
    };
  }
})();
```

Then in your main controller, you can handle the event :

```
// js/controllers/AppCtrl.js

(function(){
  'use strict';

  angular.module('app').controller('AppCtrl',[$scope,AppCtrl])

  function AppCtrl($scope){

    $scope.$on('gmap.api.loaded',function(){
      // your stuff to init after the api is loaded
    });
  }

})();
```

You just have to declare the directive in your body tag :

```
<!DOCTYPE html>
<html>

  <head></head>

  <body data-ng-app="app" data-gmap-async data-ng-controller="AppCtrl">

    <!-- template body -->

    <script type="text/javascript" src="js/app.js"></script>
    <script type="text/javascript" src="js/controllers/AppCtrl.js">
  </script>
    <script type="text/javascript" src="js/directives/gmapAsync.js">
  </script>
  </body>

</html>
```

Share Follow

edited Feb 23, 2016 at 14:04

answered Jan 28, 2015 at 13:54



Rémi Becheras

15.2k ● 14 ● 54 ● 85



I did the following

0

**gapi-service.js**

```
'use strict';

app.factory('Gapi', ['ENV', function(ENV) {

  return {
    load: function load() {
      console.log('loading google apis...');
      if (typeof gapi.client === 'undefined') {
        setTimeout(load, 500);
      }
    }
  };
}]);
```

```

    } else {
      gapi.client.setApiKey(ENV.googleToken);
      gapi.client.load('storage', 'v1', function() {
        console.log('loaded! :');
        var request = gapi.client.storage.buckets.list({ project: ''});
        console.log(request);
        request.execute(function(response) { console.log(response); });
      });
    }
  }
};
}]);

```

## index.html

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-
scalable=no, width=device-width">
    <title>"Txtbinge"</title>
  </head>

  <body ng-app="myApp">
    <script src="bower_components/jquery/dist/jquery.js"></script>
    <script src="bower_components/angular/angular.js"></script>

    <script src="scripts/client.js"></script>
    <script src="scripts/app.js"></script>
    <script src="scripts/gapi-service.js"></script>

  </body>
</html>

```

## controllers.js

```

'use strict';

app.controller('AppController', function($scope, $state, Camera, Gapi) {

  Gapi.load();

});

```

Share Follow

answered May 27, 2014 at 4:26



**Harry Moreno**

11.5k ● 8 ● 73 ● 119

See my answer on how to avoid setting a timer. – [willlma](#) Nov 25, 2014 at 0:12



Take look at this: <https://github.com/canemacchina/angular-google-client>.

0

I've write this module to use Google Api or Google Cloud Endpoint in an Angular application.



Share Follow

answered Aug 15, 2015 at 16:14



[Canemacchina](#)

179 ● 13



So I was having the same problem. Putting this code in my factory worked

0



```
var initialize = function() {
  if(gapi.client == undefined) {
    setTimeout(function() {
      initialize()
    }, 1000);
  } else {
    gapi.client.setApiKey("<api_key>");
    gapi.client.load('youtube', 'v3').then(function() {
      console.log("youtube is ready")
    });
  }
};

initialize()
```

Basically, the problem is trying to call gapi.client before it loaded. If you just check that it's loaded, and if it isn't then try again in a second (you can set the time for whatever you want, set it lower if you expect the user to need this relatively quickly after the page loads).

I was struggling with this for a while, and this is all that worked for me...Hope this helps!

Share Follow

edited Apr 18, 2016 at 3:56

answered Apr 18, 2016 at 0:53



[Cody Jacques](#)

524 ● 5 ● 9



I used a solution similar to willlma, but my application makes use of UI Router, so there's no knowing which controller will be called.

0



I was able to solve this with a Javascript Promise.



## index.html

```
<html ng-app="myApp">
<head>
  <script src="angular.js" type="text/javascript"></script>
  <script src="app.js" type="text/javascript"></script>
  <script src="https://apis.google.com/js/client.js?onload=init">
</head>
```



## app.js

```
var app = angular.module('myApp', []);

app.controller('MainController', function($scope, gapiService) {
  gapiService.then(function(gapi) {
    // You can use gapi normally here;
  });
});

app.service('gapiService', function($window) {
  return new Promise(function(resolve, reject) {
    if ($window.gapi !== undefined) {
      console.log("Have gapi already");
      resolve($window.gapi);
    } else {
      console.log("Waiting for gapi");
      $window.init = function() {
        resolve($window.gapi);
      }
    }
  });
});
```

Share Follow

answered Aug 16, 2017 at 14:16



Derick

78 ● 7