**[Understanding AngularJS $q service and promises](http://haroldrv.com/2015/02/understanding-angularjs-q-service-and-promises/" \o "Understanding AngularJS $q service and promises)**

Before diving into AngularJS $q service it is essential to understand what a promise is but most importantly why we need it.

**What is a promise?**

A promise in the Javascript and AngularJS world is an assurance that we will get a result from an action at some point in the future, let’s see the two possible results of a promise:

* A promise is said to be **fulfilled** when we get a result from that action (meaning that we get a response, regardless of whether the response is good or bad)
* A promise is said to be **rejected** when we don’t get a response(for instance if we were retrieving some data from an API and for some reason we never got a response because the API endpoint was down etc.)

**Why do we need promises?**

We need promises because we need to make decisions based on the possible results of our call (or the possibility that we don’t get a response from that call at all), probably an example will better help describe this:

Our program contacts an external API to get the list of clients  
while the response is received the program works on something else  
Once the response is received (if received) the program displays the client info on the screen  
If the response was not received (the API was down) then we display a message to the end user.  
Here is a really good example of what promises are and it’s explained as cartoon

Using Angular’s $q service to deal with promises

Angular JS provides a service called $q which allows you to work with asynchronous functions and user their return values when the execution has been completed, and what its really cool about it is that it will let you write your custom promises as well (so you can resolve or reject a promise when appropriate).

Let’s have a look at a simple example



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var deferred = $q.defer();  // deferred contains the promise to be returned  // to resolve (fulfill) a promise use .resolve  deferred.resolve(data);  // to reject a promise use .reject  deferred.reject(error); |

Now let’s have a look at how this would be implemented inside an AngularJS service:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | app.service("githubService", function ($http, $q) {        var deferred = $q.defer();        this.getAccount = function () {          return $http.get('https://api.github.com/users/haroldrv')              .then(function (response) {                  // promise is fulfilled                  deferred.resolve(response.data);                  // promise is returned                  return deferred.promise;              }, function (response) {                  // the following line rejects the promise                  deferred.reject(response);                  // promise is returned                  return deferred.promise;              })          ;      };  }); |

Finally, the AngularJS controller will use the service and either display the results on the page (if the promise was fulfilled and the data received)  or will display a message indicating that there was an error when attempting to retrieve the data from github



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | app.controller("promiseController", function ($scope, $q, githubService) {        githubService.getAccount()          .then(              function (result) {                  // promise was fullfilled (regardless of outcome)                  // checks for information will be peformed here                  $scope.account = result;              },              function (error) {                  // handle errors here                  console.log(error.statusText);              }          );  }); |