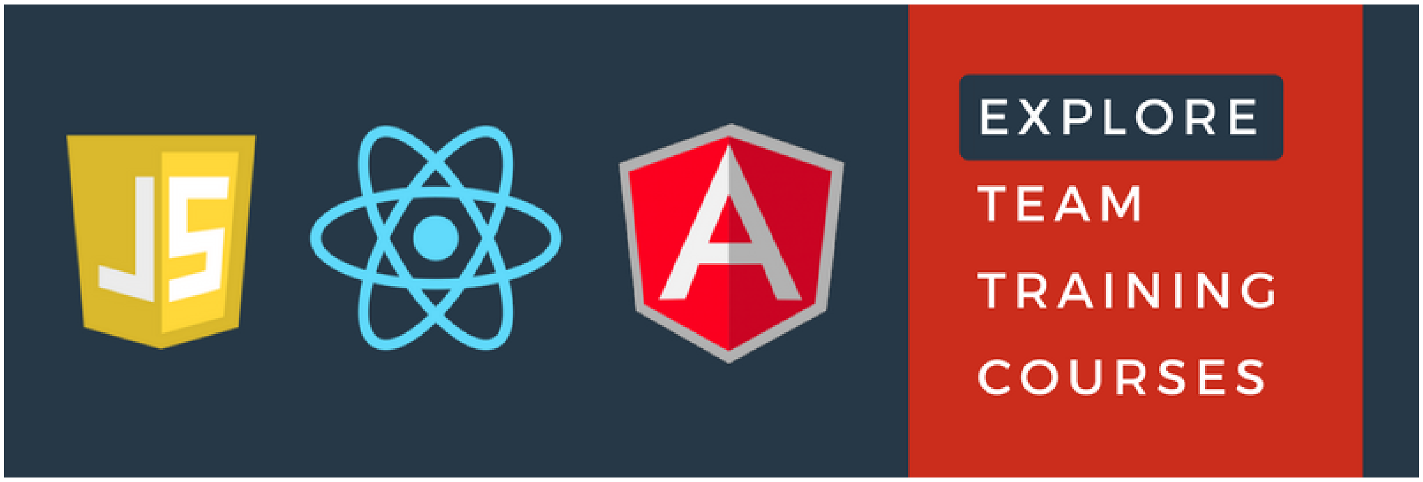
# Working With Promises in AngularJS



Let’s talk about working with promises in AngularJS. To start, **$q** is a great implementation of promises, allowing you work with deferred objects and be more efficient with Ajax calls.

**$q** api has few useful methods, first one is to create new deferred object using **$q.defer()**, so this object could be resolve in the future using $.q.defer().resolve and also to be reject using $.defer().reject methods.

Explore Angular Courses

When executing Ajax calls from AngularJS services, usually we use $http which return by default a promise object which we can attach “then” method to for result notification.

But when the service should fetch data from the backend and allow more than one AngularJS controllers to use, it doesn't make the make the most sense to fetch everything from the server every time. Then, we should save the data in local iable inside the service.

en the AngularJS controller calls the service api to get the data, it could be a promise from the Ajax call or the data stored from previous call. The AngularJS controller should be able to differentiate between the cases, so thats not an ideal practice.

The way of doing it is to wrap the data stored in AngularJS service with another $q api called $q.when which can treated as a promise for both cases, so when the $q.when receive a promise it will be resolved when the promise will and if it receive a real data it will be resolved immediately.

That is transparent to the controller and this is a good practice.

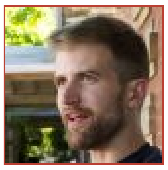
Here's a full code snippet below based on the above overview.

```
(function() {
  'use strict';

  angular
    .module('testApp')
    .service('testService', testService)
    .controller('testCtrl', testCtrl)
    .controller('testCtrl2', testCtrl2);

  function testService($http, $q) {

    // will hold backend posts
    var posts = undefined;

    // fetch all posts in deferred technique
    this.getPosts = function() {

      // if posts object is not defined then start the new process for fetch it
      if (!posts) {

        // create deferred object using $q
        var deferred = $q.defer();

        // get posts form backend
        $http.get('http://jsonplaceholder.typicode.com/posts')
          .then(function(result) {
            // save fetched posts to the local variable
            posts = result.data;
            // resolve the deferred
            deferred.resolve(posts);
          }, function(error) {
            posts = error;
            deferred.reject(error);
          });

        // set the posts object to be a promise until result comeback
        posts = deferred.promise;
      }

      // in any way wrap the posts object with $q.when which means:
      // local posts object could be:
      // a promise
      // a real posts data
      // both cases will be handled as promise because $q.when on real data will resolve it immediately
      return $q.when(posts);
    };

  }

  function testCtrl($scope, testService) {
    $scope.getPosts = function() {
      testService.getPosts()
        .then(function(posts) {

          console.log(posts);

        });
    };

    $scope.getPosts();
  }

  function testCtrl2($scope, testService) {
    $scope.getPosts = function() {
      testService.getPosts()
        .then(function(posts) {

          console.log(posts);

        });
    };

    $scope.getPosts();
  }

})();

```

Explore Angular Courses

Share this:

Kyle Pennell



Kyle Pennell is the editor of appendTo.com.

February 5, 2016 by Kyle Pennell in

AngularJS

## Comments

Comment

Comment

Need Help?

ne

Email

Add Comment