RICE®

Web Development

COMP 431 / COMP 531

more

ANGULARJS
by Google

Scott E Pollack, PhD

February 23, 2016

Front-End Recap

- HTML and HTML5
- JavaScript and JS Libraries
- Forms and Events
- CSS and Style Frameworks
- MVC and HTML Templates
- COMP 531 Draft Front-End Review
  - **Due TONIGHT**

~~Unit Testing~~
Angular Services
Headless Browsing
SPRING BREAK

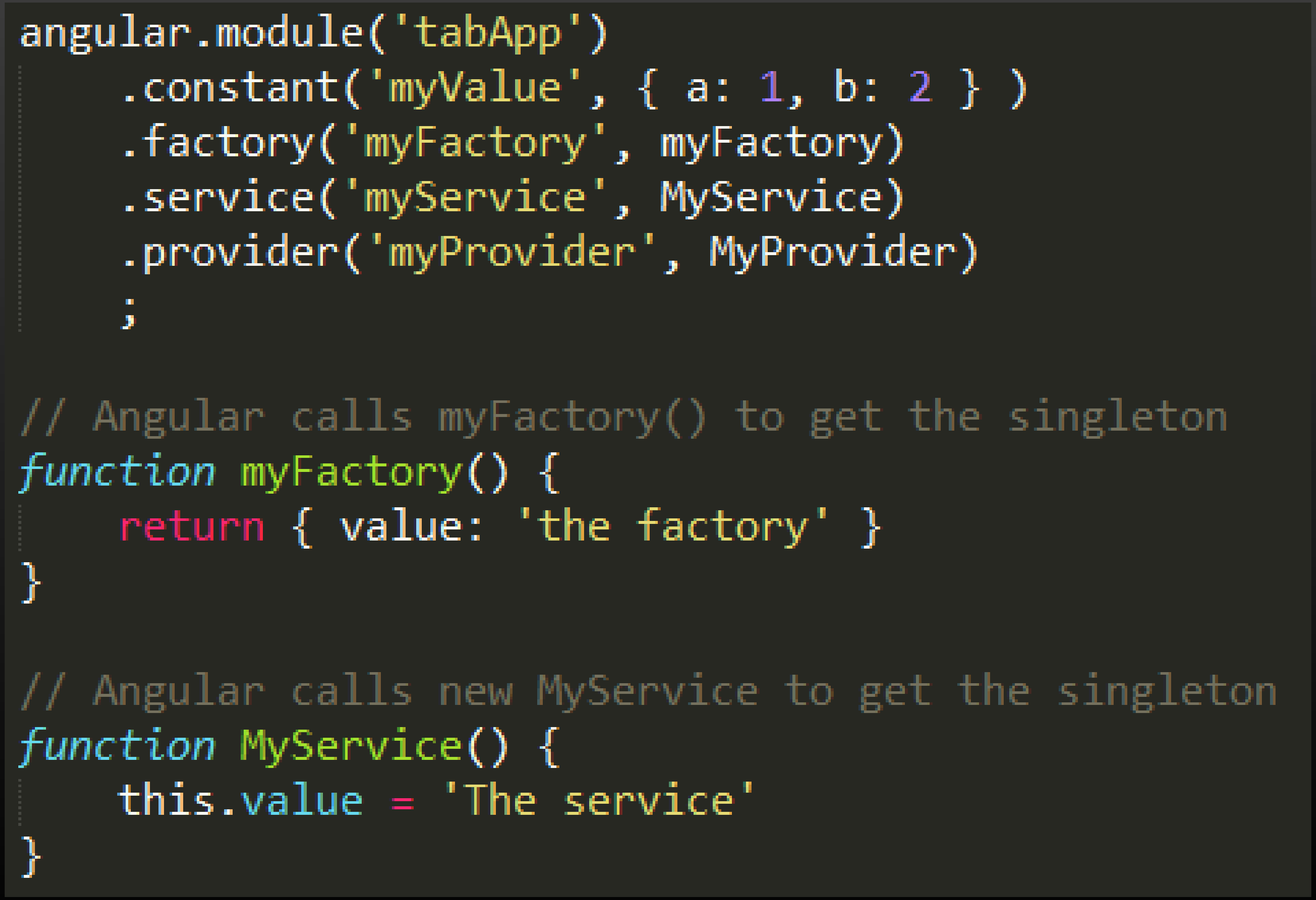
Homework Assignment 5
(Front-End App)
Due Thursday 3/10

Today's Topics

- ~~Filters~~
- ~~Forms~~
- ~~Routing~~
- Services
- $http
- $resource
- ~~Directives~~
- Mocking Services

https://www.clear.rice.edu/comp431/sample/TabApp

Constant, Factory, Service, and Provider



```
angular.module('tabApp')
    .constant('myValue', { a: 1, b: 2 } )
    .factory('myFactory', myFactory)
    .service('myService', MyService)
    .provider('myProvider', MyProvider)
    ;

// Angular calls myFactory() to get the singleton
function myFactory() {
    return { value: 'the factory' }
}

// Angular calls new MyService to get the singleton
function MyService() {
    this.value = 'The service'
}
```

DEMO

$http service

$http functions

get, head, post, put, delete, jsonp, patch

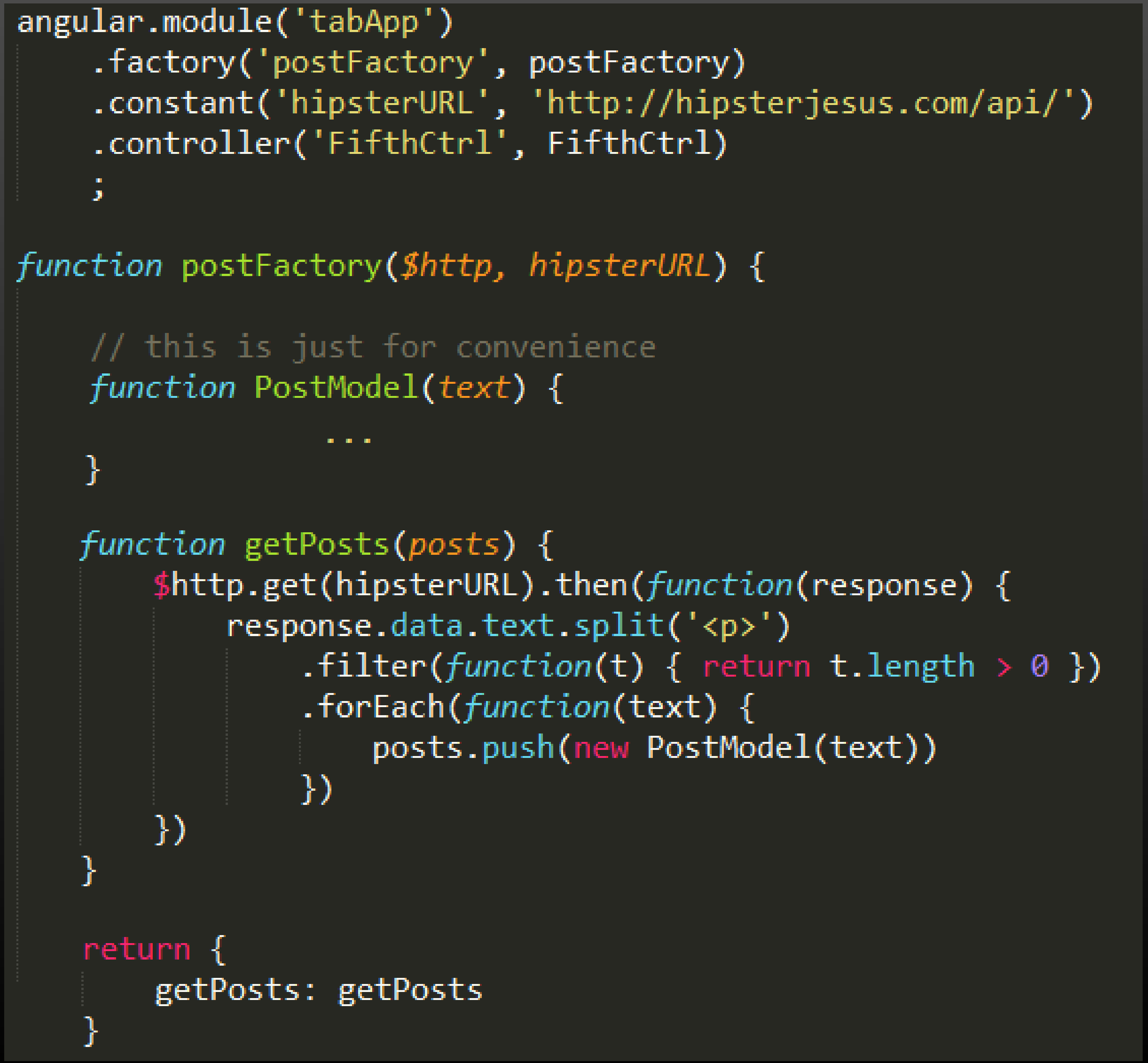
… *all return a promise*

Recall: jQuery's $.ajax returns a jQuery Deferred object (promise), with (non-standard) done(), fail(), always()

Angular's $http returns a $q promise

$http has deprecated success() and error() which provide "data" directly

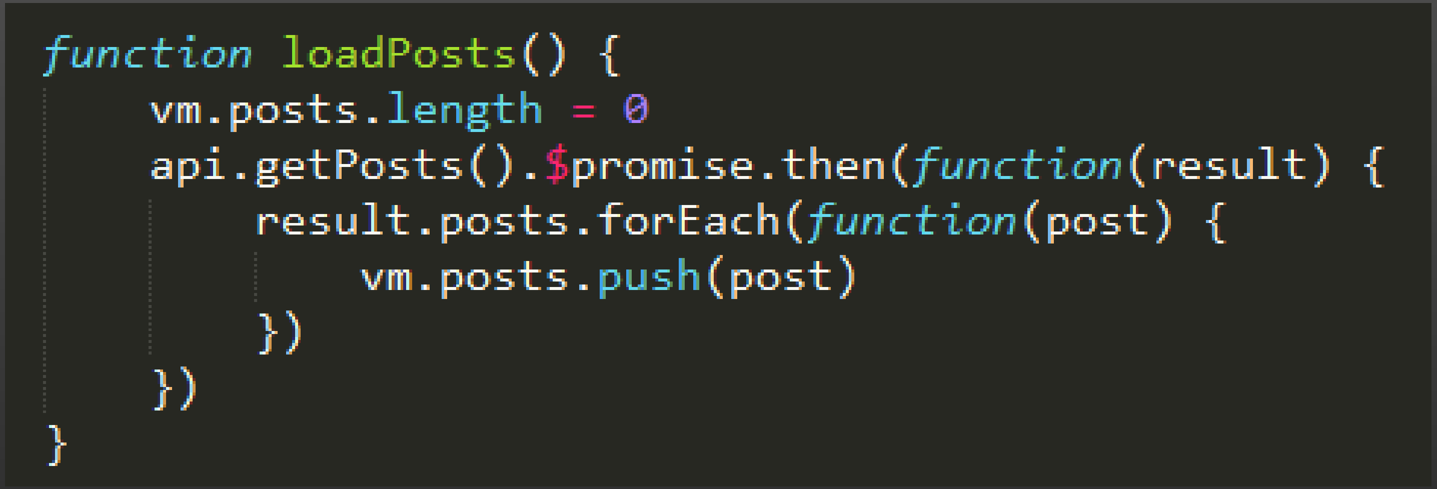
Here I use the modern promise language, then() which gives the response. I then have to dig for "data"



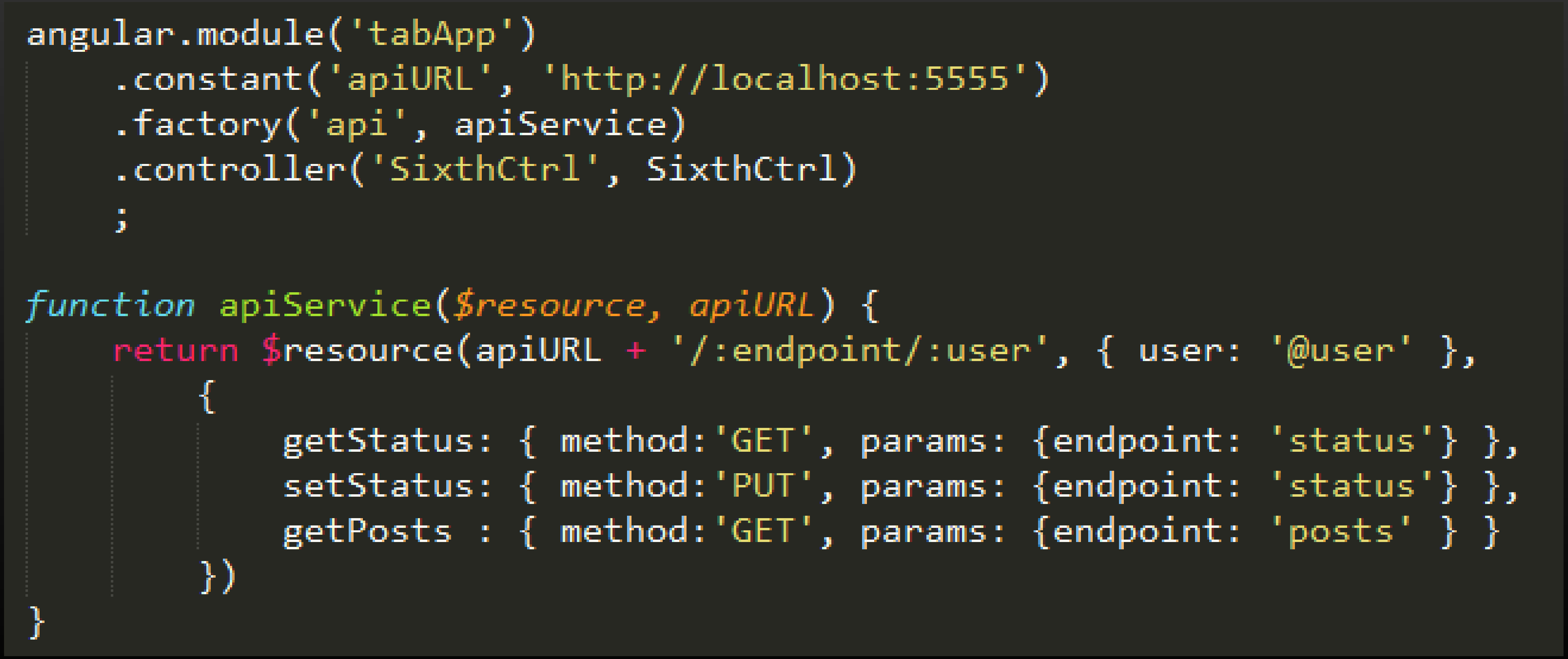
```
angular.module('tabApp')
    .factory('postFactory', postFactory)
    .constant('hipsterURL', 'http://hipsterjesus.com/api/')
    .controller('FifthCtrl', FifthCtrl)
    ;

function postFactory($http, hipsterURL) {

    // this is just for convenience
    function PostModel(text) {
            ...
    }

    function getPosts(posts) {
        $http.get(hipsterURL).then(function(response) {
            response.data.text.split('<p>')
                .filter(function(t) { return t.length > 0 })
                .forEach(function(text) {
                    posts.push(new PostModel(text))
                })
        })
    }

    return {
        getPosts: getPosts
    }
```

$resource

DEMO



```
function loadPosts() {
    vm.posts.length = 0
    api.getPosts().$promise.then(function(result) {
        result.posts.forEach(function(post) {
            vm.posts.push(post)
        })
    })
}
```



```
angular.module('tabApp')
    .constant('apiURL', 'http://localhost:5555')
    .factory('api', apiService)
    .controller('SixthCtrl', SixthCtrl)
    ;

function apiService($resource, apiURL) {
    return $resource(apiURL + '/:endpoint/:user', { user: '@user' },
        {
            getStatus: { method:'GET', params: {endpoint: 'status'} },
            setStatus: { method:'PUT', params: {endpoint: 'status'} },
            getPosts : { method:'GET', params: {endpoint: 'posts' } }
        })
}
```

Mocking Services

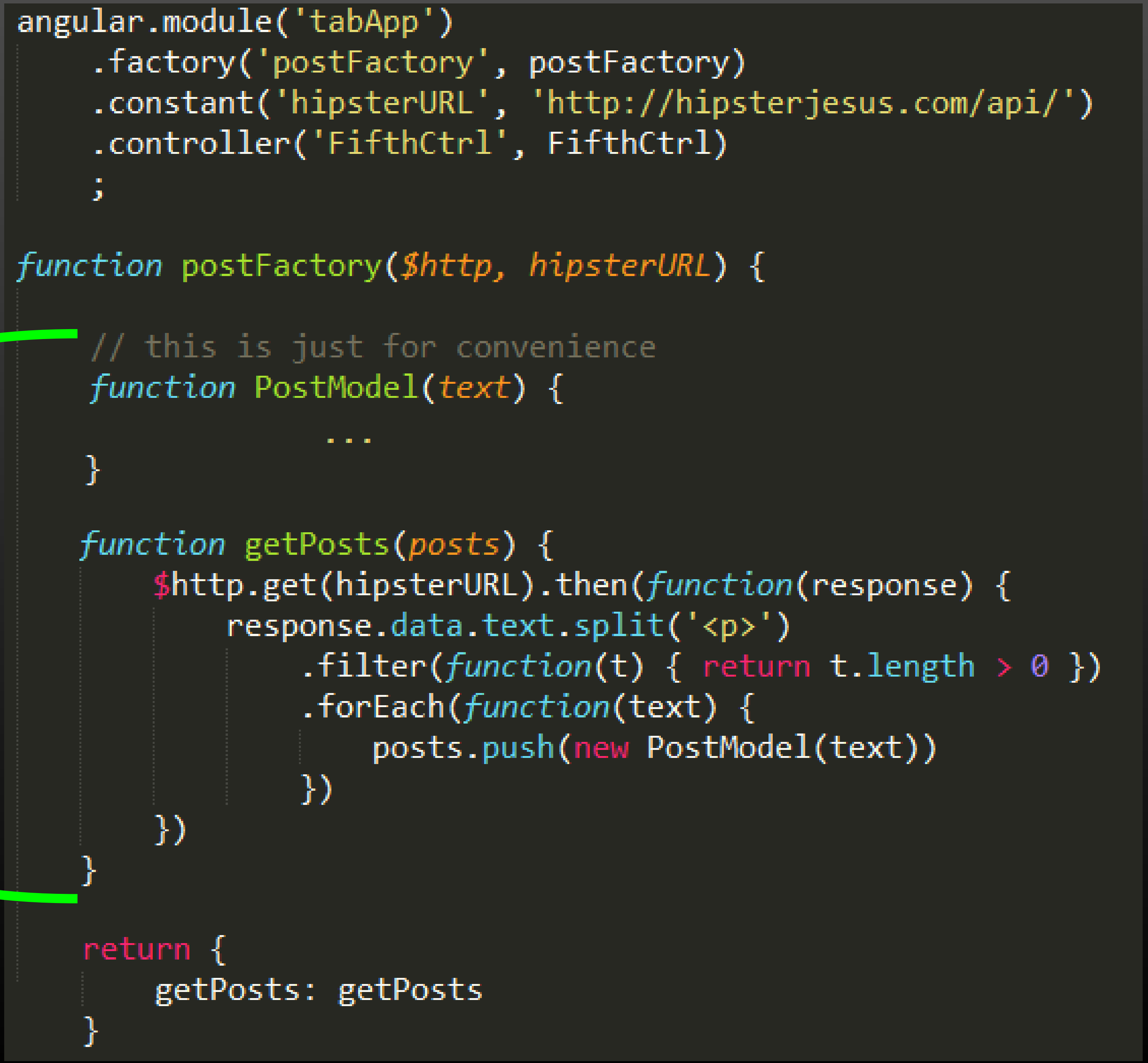
```
function loadPosts() {
    vm.posts.length = 0
    postFactory.getPosts(vm.posts)
}
```

```
angular.module('tabApp')
    .factory('postFactory', postFactory)
    .constant('hipsterURL', 'http://hipsterjesus.com/api/')
    .controller('HttpCtrl', HttpCtrl)
    ;

function postFactory($http, hipsterURL) {
```

```
// register Mock providers
beforeEach(module(function($provide) {
    $provide.factory('postFactory', function() {
        return {
            getPosts: function(posts) {
                posts.push({
                    'author':'Test',
                    'title':'A Test Post',
                    'date':'Today',
                    'body':'... test post ...'
                })
```

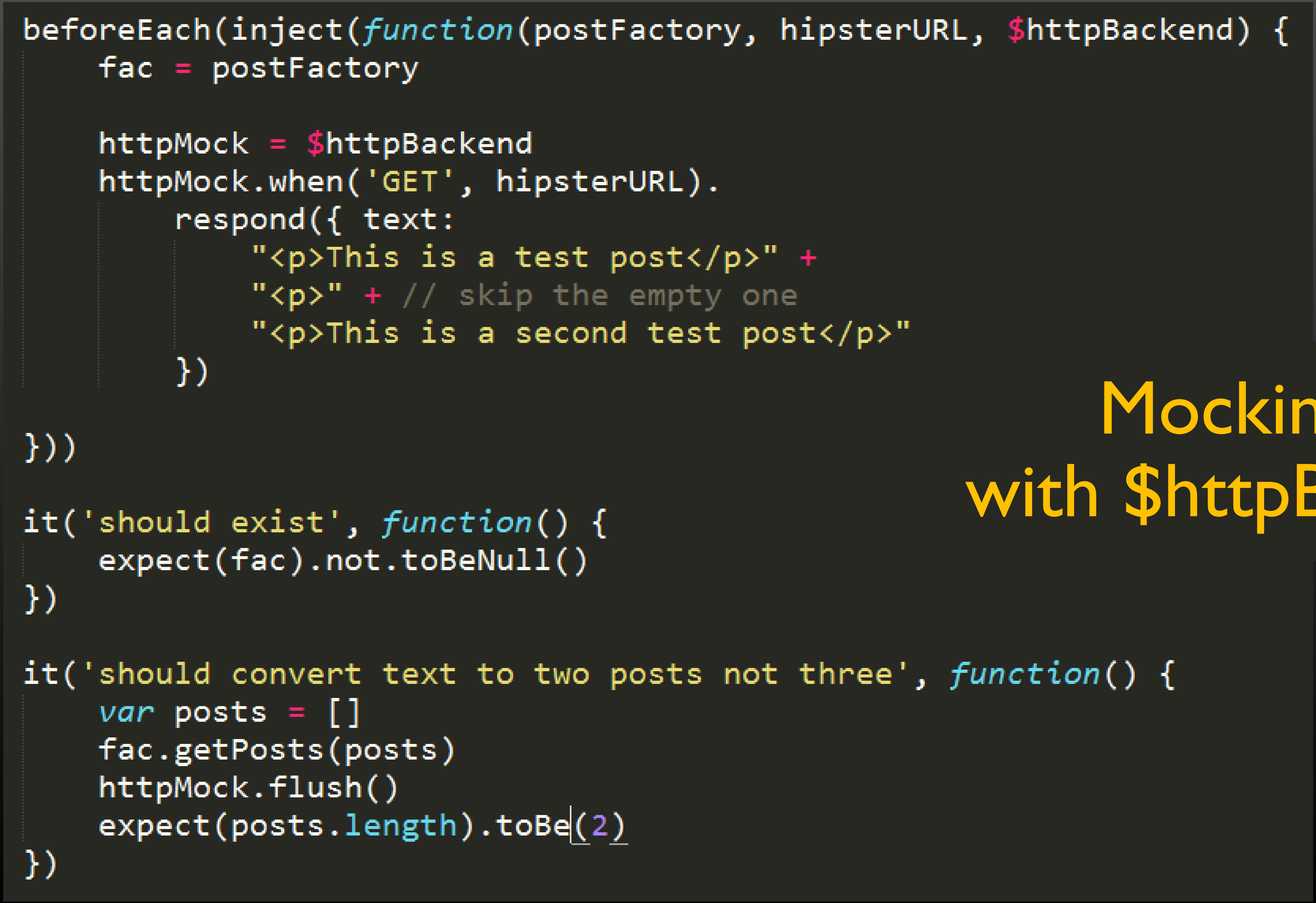
Mocking Services
$http

Need to test this logic



```
angular.module('tabApp')
    .factory('postFactory', postFactory)
    .constant('hipsterURL', 'http://hipsterjesus.com/api/')
    .controller('FifthCtrl', FifthCtrl)
    ;

function postFactory($http, hipsterURL) {

    // this is just for convenience
    function PostModel(text) {
                 ...
    }

    function getPosts(posts) {
        $http.get(hipsterURL).then(function(response) {
            response.data.text.split('<p>')
                .filter(function(t) { return t.length > 0 })
                .forEach(function(text) {
                    posts.push(new PostModel(text))
                })
        })
    }

    return {
        getPosts: getPosts
    }
```

Mocking $http with $httpBackend



```
beforeEach(inject(function(postFactory, hipsterURL, $httpBackend) {
    fac = postFactory

    httpMock = $httpBackend
    httpMock.when('GET', hipsterURL).
        respond({ text:
            "<p>This is a test post</p>" +
            "<p>" + // skip the empty one
            "<p>This is a second test post</p>"
        })

}))

it('should exist', function() {
    expect(fac).not.toBeNull()
})

it('should convert text to two posts not three', function() {
    var posts = []
    fac.getPosts(posts)
    httpMock.flush()
    expect(posts.length).toBe(2)
})
```

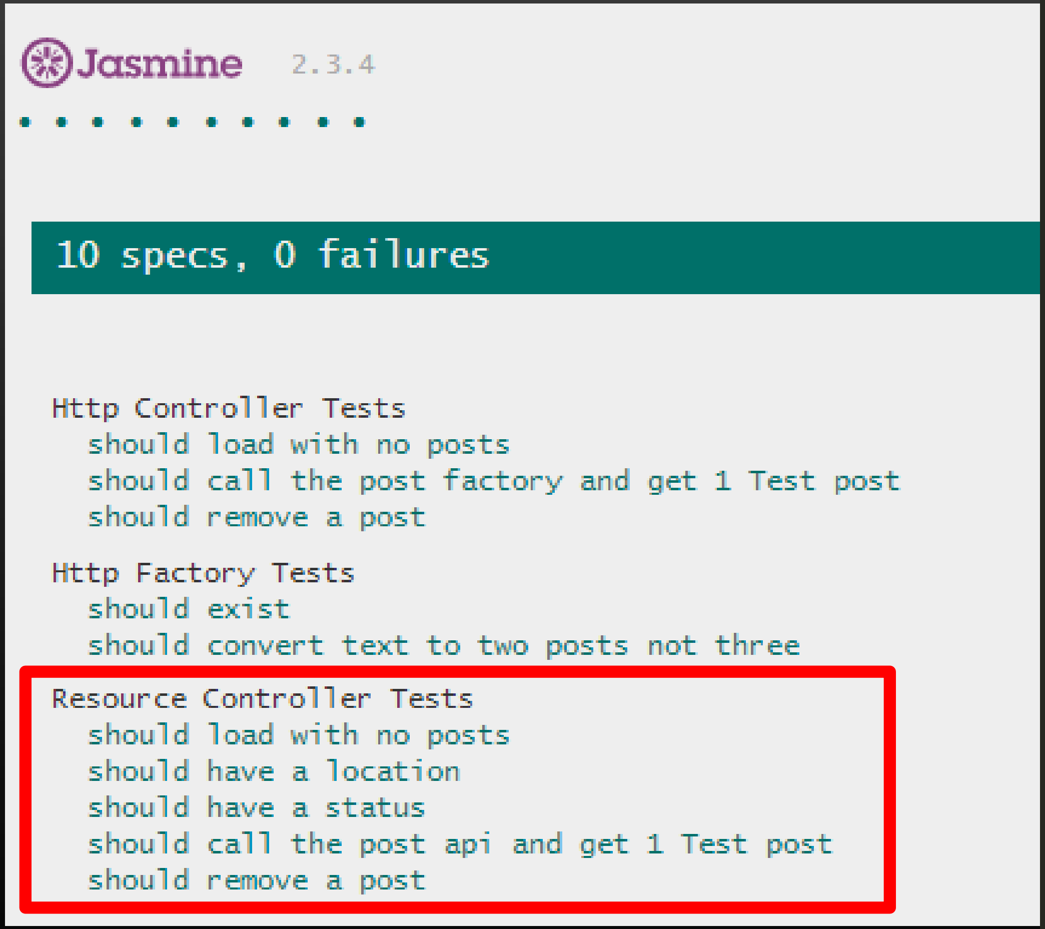
Mocking $resource

```
function loadPosts() {
    vm.posts.length = 0
    api.getPosts().$promise.then(function(result) {
        result.posts.forEach(function(post) {
            vm.posts.push(post)
        })
    })
}
```

use $httpBackend?

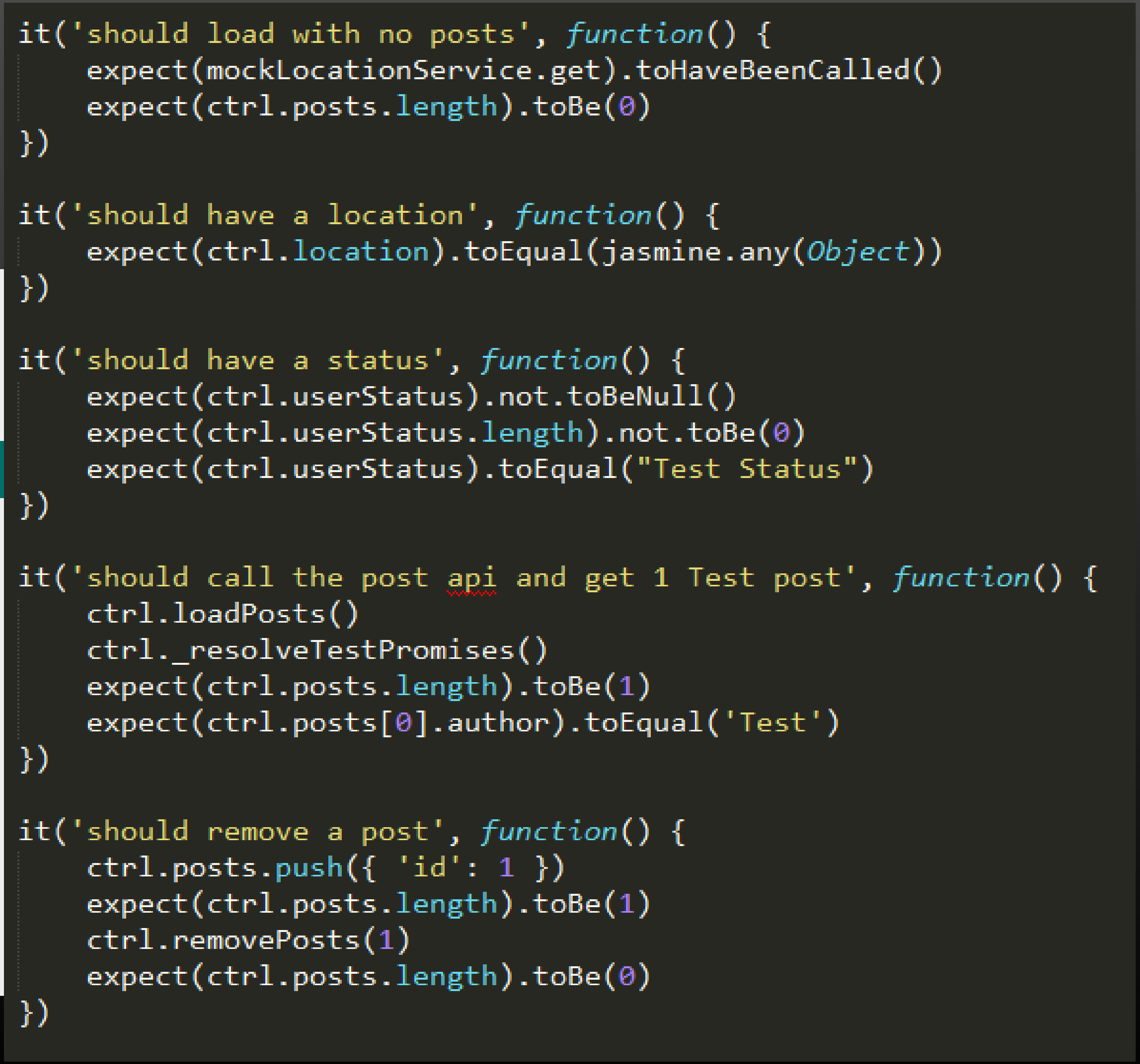
```
angular.module('tabApp')
    .constant('apiURL', 'http://localhost:5555')
    .factory('api', apiService)
    .controller('SixthCtrl', SixthCtrl)
    ;

function apiService($resource, apiURL) {
    return $resource(apiURL + '/:endpoint/:user', { user: '@user' },
        {
            getStatus: { method:'GET', params: {endpoint: 'status'} },
            setStatus: { method:'PUT', params: {endpoint: 'status'} },
            getPosts : { method:'GET', params: {endpoint: 'posts' } }
        })
}
```

Mocking Resources



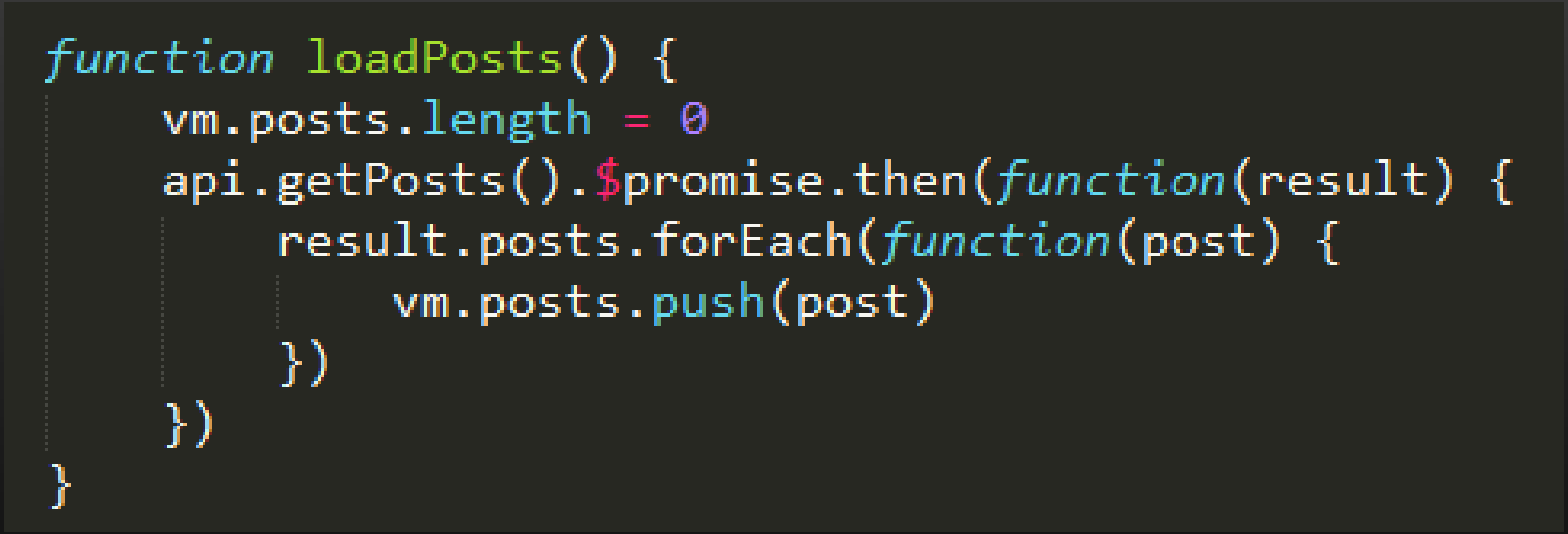
Jasmine 2.3.4

```
• • • • • • • • • •

10 specs, 0 failures

Http Controller Tests
  should load with no posts
  should call the post factory and get 1 Test post
  should remove a post

Http Factory Tests
  should exist
  should convert text to two posts not three

Resource Controller Tests
  should load with no posts
  should have a location
  should have a status
  should call the post api and get 1 Test post
  should remove a post
```



```
it('should load with no posts', function() {
    expect(mockLocationService.get).toHaveBeenCalled()
    expect(ctrl.posts.length).toBe(0)
})

it('should have a location', function() {
    expect(ctrl.location).toEqual(jasmine.any(Object))
})

it('should have a status', function() {
    expect(ctrl.userStatus).not.toBeNull()
    expect(ctrl.userStatus.length).not.toBe(0)
    expect(ctrl.userStatus).toEqual("Test Status")
})

it('should call the post api and get 1 Test post', function() {
    ctrl.loadPosts()
    ctrl._resolveTestPromises()
    expect(ctrl.posts.length).toBe(1)
    expect(ctrl.posts[0].author).toEqual('Test')
})

it('should remove a post', function() {
    ctrl.posts.push({ 'id': 1 })
    expect(ctrl.posts.length).toBe(1)
    ctrl.removePosts(1)
    expect(ctrl.posts.length).toBe(0)
})
```

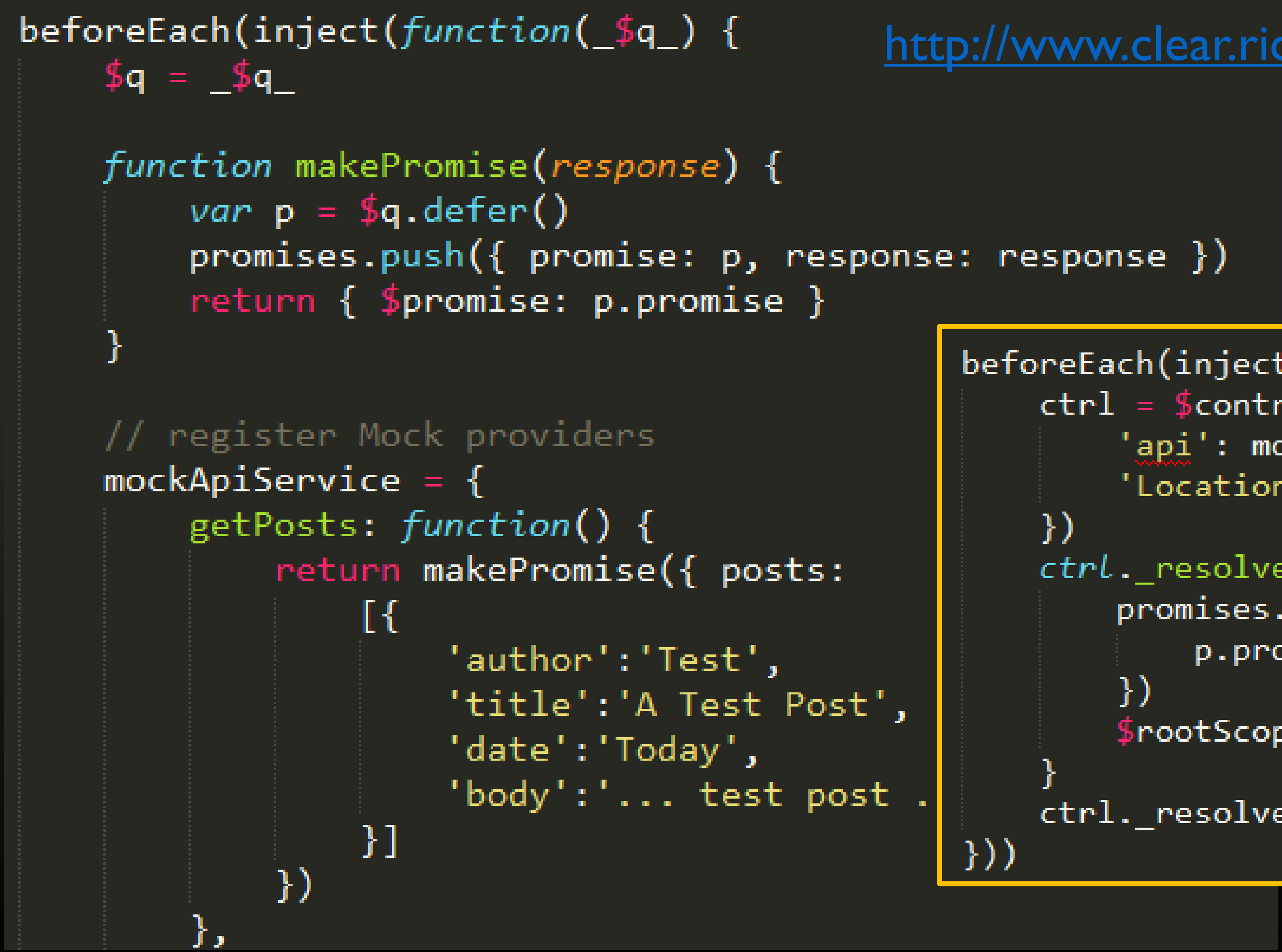
Mocking a resource with a promise



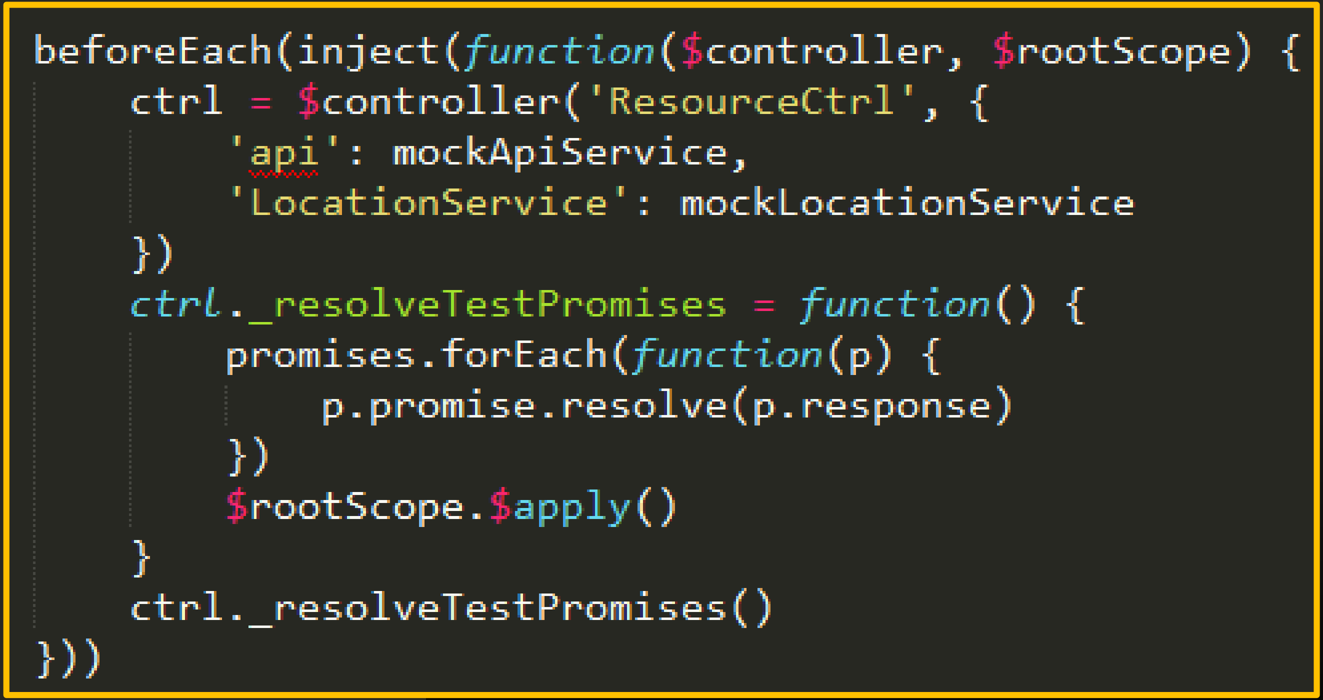
```javascript
function loadPosts() {
    vm.posts.length = 0
    api.getPosts().$promise.then(function(result) {
        result.posts.forEach(function(post) {
            vm.posts.push(post)
        })
    })
}
```

Mocking a resource with a promise

http://www.clear.rice.edu/comp431/sample/TabApp.zip

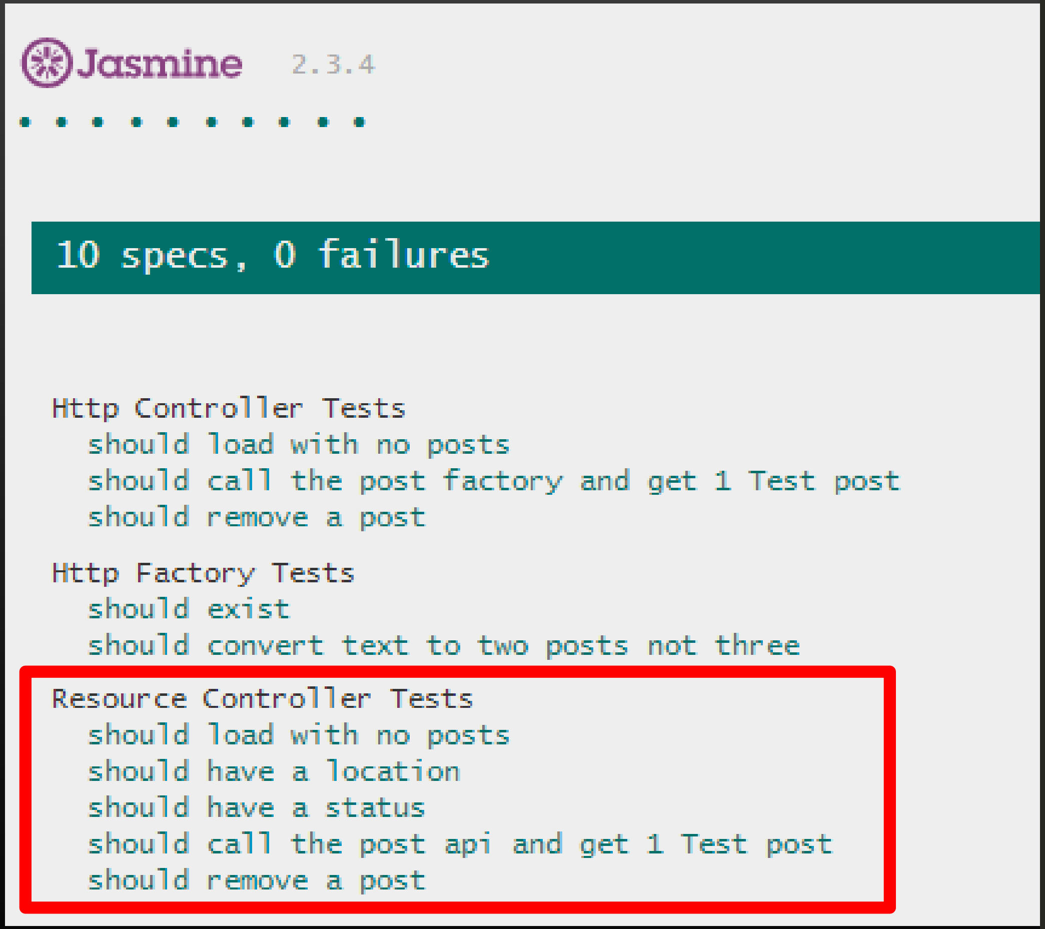


```
beforeEach(inject(function(_$q_) {
    $q = _$q_

    function makePromise(response) {
        var p = $q.defer()
        promises.push({ promise: p, response: response })
        return { $promise: p.promise }
    }

    // register Mock providers
    mockApiService = {
        getPosts: function() {
            return makePromise({ posts:
                [{
                    'author':'Test',
                    'title':'A Test Post',
                    'date':'Today',
                    'body':'... test post .
                }]
            })
        },
```

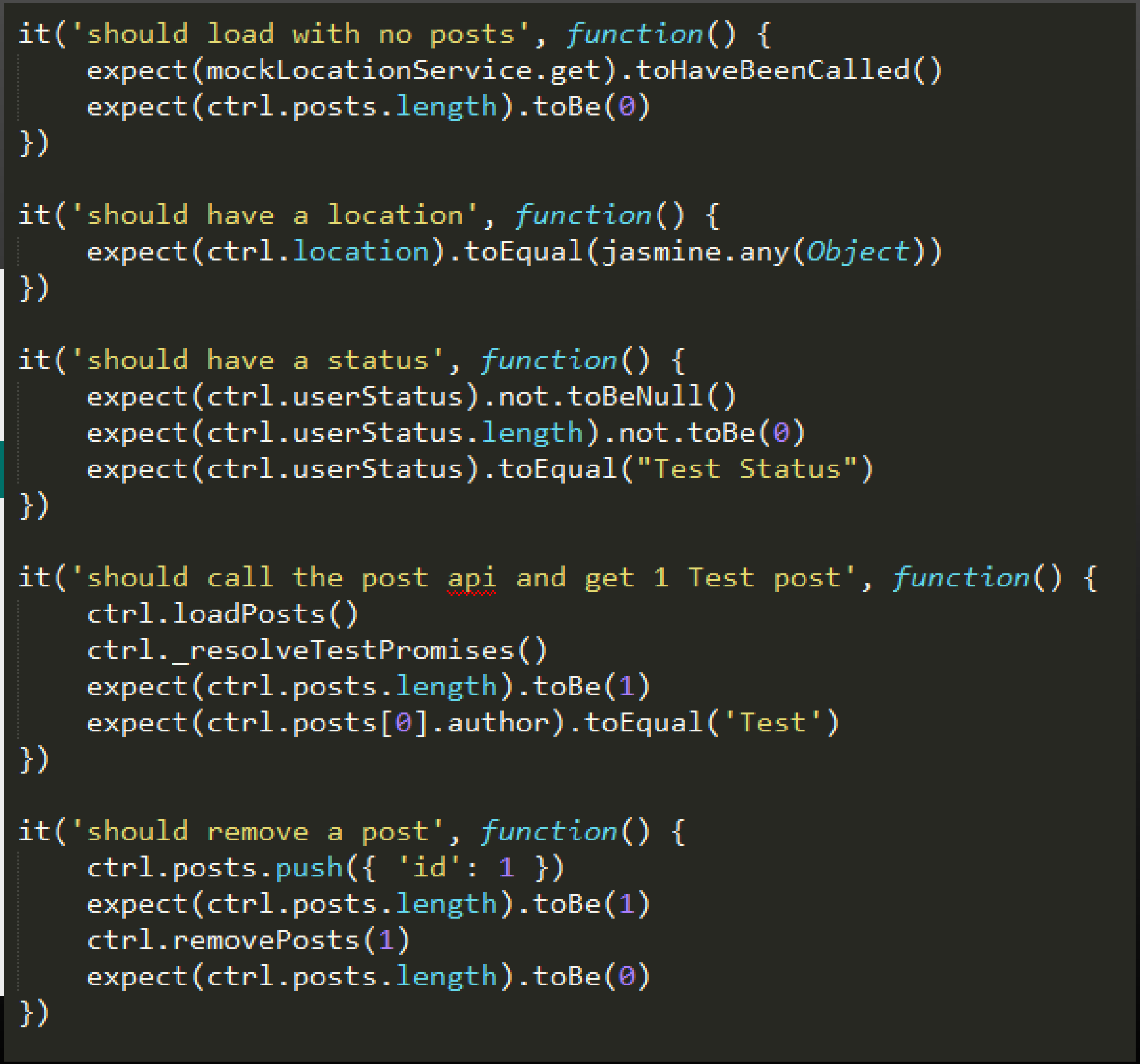


```
beforeEach(inject(function($controller, $rootScope) {
    ctrl = $controller('ResourceCtrl', {
        'api': mockApiService,
        'LocationService': mockLocationService
    })
    ctrl._resolveTestPromises = function() {
        promises.forEach(function(p) {
            p.promise.resolve(p.response)
        })
        $rootScope.$apply()
    }
    ctrl._resolveTestPromises()
}))
```

Mocking Resources



```
Jasmine 2.3.4
• • • • • • • • • •

10 specs, 0 failures

Http Controller Tests
  should load with no posts
  should call the post factory and get 1 Test post
  should remove a post

Http Factory Tests
  should exist
  should convert text to two posts not three

Resource Controller Tests
  should load with no posts
  should have a location
  should have a status
  should call the post api and get 1 Test post
  should remove a post
```



```
it('should load with no posts', function() {
    expect(mockLocationService.get).toHaveBeenCalled()
    expect(ctrl.posts.length).toBe(0)
})

it('should have a location', function() {
    expect(ctrl.location).toEqual(jasmine.any(Object))
})

it('should have a status', function() {
    expect(ctrl.userStatus).not.toBeNull()
    expect(ctrl.userStatus.length).not.toBe(0)
    expect(ctrl.userStatus).toEqual("Test Status")
})

it('should call the post api and get 1 Test post', function() {
    ctrl.loadPosts()
    ctrl._resolveTestPromises()
    expect(ctrl.posts.length).toBe(1)
    expect(ctrl.posts[0].author).toEqual('Test')
})

it('should remove a post', function() {
    ctrl.posts.push({ 'id': 1 })
    expect(ctrl.posts.length).toBe(1)
    ctrl.removePosts(1)
    expect(ctrl.posts.length).toBe(0)
})
```

http://www.clear.rice.edu/comp431/sample/TabApp and http://www.clear.rice.edu/comp431/sample/TabApp.zip

In-Class Exercise: Mock a Login

- The Plunker example shows how to login to the dummy server and redirect views using routing
  - http://plnkr.co/edit/EEBZXvZ5ZdrgEaQk4Xas?p=preview
- Mock the $resource api and write a unit test that logs in a user
  - Store the username of the user in a UserService
- Test logging out the user, validate the UserService has no username
- If you have time, test that the UserService will share the username between multiple controllers

Turnin your spec.js and js files
COMP431-S16:inclass-13