Chapter 10

***Listing 10-1.*** app.config.js: setting up a constant

angular.module('app.config',[])

.constant('myConfig', {

'apiUrl': 'http://localhost:8080',

'adminEmail': 'admin@mycompany.com'

});

***Listing 10-2.*** Using the configuration data in the app.config module

module.factory('memberDataStoreService', function ($http, **myConfig**) {

var memberDataStore = {};

memberDataStore.doRegistration = function (theData) {

var promise = $http({method: 'POST', url: **myConfig.apiUrl**, data: theData});

return promise;

}

return memberDataStore;

})

***Listing 10-3.*** app.development.config..js: our development configuration

angular.module('app.development.config',[])

.constant('myConfig', {

'database-host': '127.0.0.1',

'database-name: 'local-database'

});

***Listing 10-4.*** app.production.config.js: our development configuration

angular.module('app.production.config',[])

.constant('myConfig', {

'database-host': '168.63.165.103',

'database-name: 'production-database'

});

***Listing 10-5.*** An initial look at a set of tests

describe('user login form', function() {

it('ensures invalid email addresses are caught', function() {}); it('ensures valid email addresses pass validation', function() {}); it('ensures submitting form changes path', function() { });

});

***Listing 10-6.*** The “ensures valid email addresses pass validation test” it('ensures valid email addresses pass validation', function() {

var validEmails = [

'test@test.com',

'test@test.co.uk',

'test734ltylytkliytkryety9ef@jb-fe.com'

];

for (var i in validEmails) {

var valid = LoginService.isValidEmail(validEmails[i]); **expect(valid).toBeTruthy();**

}

});

***Listing 10-7.*** Sample expectations and matchers

// We expect 1 + 1 to equal 2.

// If it doesn’t the test fails expect(1 + 1).toEqual(2); // We expect myObject to be null.

// If its not the test fails var myObject = null; expect(myObject).toBe(null);

// We expect null to be null // If its not the test fails expect(null).toBeNull();

// We expect the kids array to contain the string "Natalie"

// If it doesn't the test fails

var kids = ["Jenna", "Christopher", "Natalie", "Andrew", "Catie"];

expect(kids).toContain("Natalie"); // We expect 10 to be less than 15.

// If it isn’t the test fails expect(10).toBeLessThan(15);

***Listing 10-8.*** Lazy error handling

if(user.paySuccess()){

// The users payment was succesful

goToSuccessScreen();

} else{ // The users payment failed, pop up an alert

// and log to the console.

alert('An error has occured'); console.log('An error has occured');

}

***Listing 10-9.*** A basic server logging mechanism

function log(sev, msg) {

var img = new Image(); img.src = "log.php?sev=" + encodeURIComponent(sev) +

"&msg=" + encodeURIComponent(msg);

}

***Listing 10-10.*** Using a logging service var paymentService = null;

try {

paymentService.payForItem();

} catch(e){

// alert user to the error showSomeFriendlyFeedback();

// Trasmit the error to the server

log(1, "001: The user was unable to complete his purchase"); }

***Listing 10-11.*** Hiding a specific portion of the DOM tree

<h1 ng-cloak>Hello, {{name}}!</h1>

<p ng-cloak>It's great to see you again, you haven't logged in since {{lastLoggedIn}}</p>

***Listing 10-12.*** Hiding a larger portion of the DOM tree

<div ng-controller="HelloCtrl" ng-cloak>

<h1>Hello, {{name}}!</h1>

<p>It's great to see you again, you haven't logged in since {{lastLoggedIn}}</p>

<p>We hope you have a {{typeOfDay}} day!</p>

</div>

***Listing 10-13.*** The ngBind directive in action

<div ng-controller="HelloCtrl"> Hello, <span **ng-bind="name"**></span>!

</div>

***Listing 10-14.*** Typical unminified source code function getElementsByClassName(className) {

var results = [];

walkTheDOM(document.body, function (node) { var array, // array of class names ncn = node.className; // the node's classname // If the node has a class name, then split it into a list of simple names. // If any of them match the requested name, then append the node to the list of results. if (ncn && ncn.split(' ').indexOf(className) >= 0) {

results.push(node);

}

});

return results; }

***Listing 10-15.*** Minified source code

function getElementsByClassName(e){var t=[];walkTheDOM(document.body,function(n) {var r,i=n.className;if(i&&i.split(" ").indexOf(e)>=0){t.push(n)}});return t}

***Listing 10-16.*** A sample Gruntfile

module.exports = function (grunt) {

// Part One

// Configure the various tasks we want to run

grunt.initConfig({ watch: { build: {

files: 'src/\*\*/\*.\*', **tasks: ['uglify', 'copy'**],

options: {

livereload: true

}

}

}, uglify: { options: { mangle: true

}, my\_target: { files: {

'site/js/js.min.js': ['src/js/\*.js']

}

}

}, copy: { main: { files: [

// includes files within path and its sub-directories

{ expand: true, cwd: 'src/', src: ['\*\*', '!\*\*/assets-master/\*\*', '!\*\*/css/\*\*',

'!\*\*/js/\*\*'], dest: 'site/'}

]

}

}

});

// Part Two

// Load the tasks we have just configured **grunt.loadNpmTasks('grunt-contrib-watch');**  **grunt.loadNpmTasks('grunt-contrib-uglify');**  **grunt.loadNpmTasks('grunt-contrib-copy');**

// The task we want to run by default when we start Grunt

grunt.registerTask('default', ['watch']);

}