SASCHA BRINK

ANGULARJS COOKBOOK

70 RECIPES FOR ANGULARJS

AngularJS Cookbook

70 Recipes for AngularJS 1.2

Sascha Brink

This book is for sale at http://leanpub.com/angularjs-cookbook

This version was published on 2014-03-12

Tweet This Book!

Please help Sascha Brink by spreading the word about this book on Twitter!

The suggested hashtag for this book is #angularjs.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#angularjs

Contents

Introduction

About the author

Sascha Brink is a web technology consultant, freelance writer and author. He's developing web applications since 12 years. For him AngularJS is the most promising framework on the client side today. On the server side he favors Ruby on Rails. Sascha has published several articles about AngularJS and is a part of the german chapter http://angularjs.de.

Create an analog clock with SVG

Problem

You want to create a simple animation without using the canvas element.

Solution

This is more an inspirational than a complex example. SVG[1] is a vector image format which can be embedded in HTML. So if you're a little creative, you can do otherwise complex animations with very little code.

Here we use an analog clock as an example for AngularJS in combination with SVG.

It consists of only a circle and 3 lines. The 3 lines are the hands for hour, minute and second. We rotate them with angular and the `$interval` service.

In the HTML, you see an example of how easy it is to embed a SVG. For more information, see here[2].

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <svg xmlns="http://www.w3.org/2000/svg" width="200" height="200">
    <g>
      <circle style="stroke: #ccc; fill: #fff;" cx="100" cy="100" r="100"/>
      <line x1="100" y1="100" x2="100" y2="50"
            style="stroke-width: 5px; stroke: #333;"
            ng-attr-transform="rotate({{hourRotation}} 100 100)" />
      <line x1="100" y1="100" x2="100" y2="20"
            style="stroke-width: 3px; stroke: #888;"
            ng-attr-transform="rotate({{minuteRotation}} 100 100)" />
      <line x1="100" y1="100" x2="100" y2="5"
            style="stroke-width: 2px; stroke: #bb0000;"
            ng-attr-transform="rotate({{secondRotation}} 100 100)" />
```

[1]http://en.wikipedia.org/wiki/Scalable_Vector_Graphics
[2]https://developer.mozilla.org/en-US/docs/Web/SVG

```
    </g>
  </svg>
</body>
</html>
```

The application code is easy, too. We inject and use the `$interval` service which runs every second. We then calculate the angle of rotation for each hand.

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope, $interval) {

    function calculateRotation() {
      var now = new Date();
      $scope.hourRotation   = 360 * now.getHours()   / 12;
      $scope.minuteRotation = 360 * now.getMinutes() / 60;
      $scope.secondRotation = 360 * now.getSeconds() / 60;
    }
    $interval(calculateRotation, 1000);
    calculateRotation();
  });
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-svg-clock

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-svg-clock/

Prevent duplicate warnings in ng-repeat

Problem

If you use `ng-repeat` with duplicates in an array, you get the error message “Duplicates in a repeater are not allowed”.

Solution

This happens because AngularJS exspects every element to have an unique identifier. This is for tracking the insertion, deletion and moving of element. To change the identifier for the element you can use the `track by` syntax.

If the the following snippet throws an error:

```
<li ng-repeat="item in [4,4,4]">
```

You can enforce artifical uniqueness by using the index of the current element in the repeater:

```
<li ng-repeat="item in [4,4,4] track by $index">
```

Write a decorator - change a service result without monkey patching

Problem

You want to change the result of a service or extend it without changing the service itself.

Solution

The solution is to write a decorator. Decorators can intercept calls to service (provider, factory, service, value) and modify them.

In this example we decorate the `$log` service to prepend the used log level to the output.

Decorator can only be initialized in a config block. This adds some limitations because you can't inject other services in the config block. You can only use the config blocks of providers.

For a decorator to work, we use the `$provide` provider and call the method decorator on it. In the decorator function, $delegate is automatically injected and contains the decorated service. In this example `$log`.

We create a new object which is api compatibility to the `$log` service. We do this by generating the `$log` methods dynamically and call the original service after we modified the log message.

```
.config(function($provide) {
  $provide.decorator('$log', function($delegate) {
    var logger = {};
    ['log','info','warn','error','debug'].forEach(function(level) {
      logger[level] = function(message) {
        $delegate[level]('[' + level.toUpperCase() + '] ' + message);
      };
    });
    return logger;
  });
})
```

Complete example

<<(code/directives-log-decorator/application.js)

<<(code/directives-log-decorator/index.html)

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/services-log-decorator

Online demo: http://sbrink.github.io/angularjs-cookbook-code/services-log-decorator/

How to cache data with promises

Problem

You want to cache an asynchronous request and always want to work with a promise. The fetching of the data from the memory cache is synchronous. If the cache misses, it's asynchronous.

Solution

The solution is to always return a promise. If the data is cached, we just immediately resolve the promise. We could you how to do this very easily.

Covert a value to a promise

The first question is how to convert a cached value into a promise.

A naive solution would be

```
var deferred = $q.defer();
deferred.resolve(cachedValue);
return deferred.promise;
```

Because it's such a common pattern, AngularJS has a shortcut for it:

```
$q.when(cachedValue)
```

`$q.when` is capable of a lot more, see convert 3rd party promises.

Promise all the time

To always return a promise, we check if the the return value is already cached. If it is, we return a resolved promise with `$q.when`. If not, we call our promise and on success we'll cache the result.

```
if (cache) {
    return $q.when(cache);
} else {
    return promise.then(function(result) {
        cache = result;
        return reulst;
    });
}
```

You’ll find a full working example in the code section.

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/promises-cache-data

Online demo: http://sbrink.github.io/angularjs-cookbook-code/promises-cache-data/

Testing only a subset of tests

Problem

You’re testing and don’t want to always run all of your tests.

Solution

Jasmine has two really handy methods for this:

- ddescribe: Runs only the current describe block
- iit: Runs only the current test

If you want, you can have more than one ddescribe or iit. All tests with this special marker will run.

Redirect to an error page

Problem

If an error occurs, you want to redirect the user to a general error page.

Solution

For the solution we use the `$exceptionHandler`. The tricky part here is to avoid a cycliomatic dependency error `$location <- $exceptionHandler <- $rootScope`. In order to solve this, we avoid using the `$location` service directly. Instead we use an indirect way and inject the `$injector`. With this service we get `$location` manually.

```
.factory('$exceptionHandler', function($injector) {
  var $location;
  return function(exception, cause) {
    $location = $location || $injector.get('$location');
    $location.path('/error');
  };
});
```

Deregister an event listener

Problem

You have registered an AngularJS event listener with `scope.$on` and want to deregister it but you haven't found sth. like an `.off()` method.

Solution

The solution is found in the source code. If we look at the function definition of the `$on` function we see that it return a function itself. This function is capable of deregistering the listener.

```
$on: function(name, listener) {
  var namedListeners = this.$$listeners[name];
  if (!namedListeners) {
    this.$$listeners[name] = namedListeners = [];
  }
  namedListeners.push(listener);

  return function() {
    namedListeners[indexOf(namedListeners, listener)] = null;
  };
}
```

To get this to work, we have to save a reference to our the returned function of `$on`. When we finally want to remove the listener, we just have to execute it.

```
var myEventOffFn = $scope.$on('onMyEvent', myListener);

// remove listener
myEventOffFn();
```