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

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

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.

l Directive / View Recipes

1 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/

2 Build a date select

Problem

You want to give the user a date selection with month, day and year.

Solution

For this we build a custom directive which takes and date object. The directive uses two-way-databinding to sync the select fields with the attribute `model`.

We will call our directive `dateselect` and will use the following html code for it. We use two directive to show the synchronization between them:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <dateselect model="current"></dateselect><br/>
  <dateselect model="current"></dateselect><br/>

  {{current | date:'yyyy-MM-dd'}}
</body>
</html>
```

We initialize the `current` variable with the current date in the controller.

```
angular.module('cookbookApp', [])
  .directive('dateselect', function() {
    return {
      restrict: 'E',
      template:
        '<select ng-model="date.month" ' +
          'ng-options="month for month in months"></select>' +
          '<select ng-model="date.day" ' +
          'ng-options="day for day in days"></select>' +
          '<select ng-model="date.year" ' +
          'ng-options="year for year in years"></select>',
      scope : {
        model: '='
      },
      controller: function($scope) {
        var i;
        $scope.date = {};

        $scope.days = [];
        for (i = 1; i <= 31; i++) { $scope.days.push(i); }

        $scope.months = [];
        for (i = 1; i <= 12; i++) { $scope.months.push(i); }

        $scope.years = [];
        for (i = 1980; i <= (new Date().getFullYear()); i++) {
          $scope.years.push(i);
        }

        $scope.$watch('model', function(newDate) {
          $scope.date.month = newDate.getMonth()+1;
          $scope.date.day = newDate.getDate();
          $scope.date.year = newDate.getFullYear();
        }, true);

        $scope.$watch('date', function(newDate) {
          $scope.model.setDate(newDate.day);
          $scope.model.setMonth(newDate.month-1);
          $scope.model.setFullYear(newDate.year);
        }, true);
      }
    };
```

```
43  })
44  .controller('MainController', function($scope) {
45    $scope.current = new Date();
46  });
```

The directive works as follows:

We set `restrict` the *E* (for element), to allow dateselect as tag. We use an isolated scope with `model` as our communicator to the outside world. In the template we simply draw three select fields for month, day and year. We generate the necessary options in the controller.

Because we can fully rely on the template, we don't need the link function here and just use the controller.

The controller is the part where the whole work is done. Here we have three loops for days, months and years for the select boxes. Furthermore we initialize an object `date`, to hold the selected month, day and year.

Additionally we have to watches. One to reflect a model change from the outside and one to reflect a change from the select boxes.

Be careful to set the `true` as second parameter for `$scope.$watch('...', function() {}, true)`. This signals to do a deep watch. Otherwise the updating won't work all of the time.

The two watches are very similar. If the `model` attribute changes, we update the select fields. If we change one of the select boxes, we update the model. Just just have to be careful with the month. The month is inconsistent. While days begin with `1`, months begin with `0`. This means you have to add or subtract `1`.

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-birthdate-select/

3 How to use AngularJS with a curly braced template engine

Problem

You want to change the double curly braces `{{` from AngularJS because they're conflicting with your template engine on the backend.

Solution

You can indeed change the curly braces to e.g. `[[`. In AngularJS, the `$interpolate` provider does the template interpretation and has an option for changing the symbols.

To do so, you can inject the `$interpolateProvider` in the config block of your module. There you can set `startSymbol` and `endSymbol` as demonstrated below:

```
angular.module('cookbookApp', [])
    .config(function($interpolateProvider){
        $interpolateProvider.startSymbol('[[');
        $interpolateProvider.endSymbol(']]');
    });
```

If you split your application into different modules, you have to configure the the alternative syntax for every module!

Tip 1: $interpolate stand alone

You can also use the interpolate provider for your own templating purposes. Just inject $interpolate and use:

```
$interpolate('Hello {{name}}!')({name: 'World'}); // => 'Hello World!'
```

Tip 2: Skipping interpolation

For a part of the DOM, you can disable the translation of curly braces altogether.

```
<div ng-non-bindable>
   ...
</div>
```

4 Use the $compile function

Problem

You created an html string with some AngularJS directives inside your link function. As you attach it to the DOM, you just get exactly what you wrote as output but you expected AngularJS to compile it for you.

Solution

If you you use AngularJS directive in your normal html, AngularJS will compile it for you when the site is rendered. If you create a string inside a link function of a directive, you have to do this manually. You can do this by injecting the `$compile` service. You can use `$compile` with either a `string` or an `angular.element`. This is returning a linking function which is called with a scope. This again returns an element which the can be inserted into the DOM.

With a string:

```
newElement = $compile(myString)(scope)
```

With an element:

```
newElement = $compile(angular.element)(scope)
```

Returns an `angular.element`, **not a string**!

What if don't have a scope to compile it against?

Usually you would use `$compile` inside a link function of a directive and there you would have a scope. If for some reason you use it somewhere else, e.g. inside a service, you could of pass the scope to the function from outside. If you just need a scope, you can also just use the root scope. Inject `$rootScope` and you're ready to go.

```
newElement = $compile(myString)($rootScope)
```

5 Show a confirm box before ng-click is executed

Problem

For an delete button you don't want to execute a ng-click immediately but first show a confirm box

Solution

We create a new directive which is called `confirmed-click` instead of `ng-click`. This directive works the same (you can also use `$event` in it) but with a confirm box before the click is executed. Here is an example:

```
<button confirmed-click="removeTask($index)">
```

The full view we use is a simple list with tasks you can delete. Before the deletion comes a confirm box:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body ng-controller="MainController">
  <table>
    <tr ng-repeat="task in tasks">
      <td>{{task}}</td>
      <td>
        <button confirmed-click="removeTask($index)">Delete</button>
      </td>
    </tr>
  </table>
</body>
</html>
```

For this to works, we bind the element to the click event and create a new box and disable the butotn when its clicked. The box itself has a new scope with two functions `ok()` and `cancel()`. If we click `ok()` we trigger the original click event. On `cancel()` we close the box an reenable the button.

```
angular.module('cookbookApp', [])
  .directive('confirmedClick', function($parse, $q, $compile, $rootScope) {
    var box = '<div class="box"><div>Really?</div> ' +
      '<button ng-click="close($event, true)">OK</button>' +
      '<button ng-click="close($event)">Cancel</button>' +
      '</div>';
    return {
      link: function(scope, element, attrs) {
        var fn = $parse(attrs.confirmedClick);
        element.on('click', function() {
          var boxScope = $rootScope.$new();
          var boxElement = $compile(box)(boxScope);

          element.attr('disabled', 'disabled');
          element.append(boxElement);

          boxScope.close = function(event, execute) {
            event.stopPropagation();
            element.removeAttr('disabled');
            boxElement.remove();
            if (execute) {
              fn(scope, {$event: event});
            }
          };
        });
      }
    };
  })
  .controller('MainController', function($scope) {
    $scope.tasks = ['Tidy up', 'Wash the dishes'];
    $scope.removeTask = function(index){
      $scope.tasks.splice(index, 1);
    };
  });
```

For the style we give the directive element a relative position, so that we can place the box we show absolute to it:

```
button[confirmed-click] {
  position: relative;
}

button[confirmed-click] .box {
  position: absolute;
  border: 1px solid #ccc;
  width: 100px;
  background-color: #f9f9f9;
  top: 0;
  right: -100px;
}
```

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-confirm-box/

6 Create a digital clock

Problem

You want to create a directive which shows the current time.

Solution

The solution is a little directive with a timer. For displaying the date, we use the date filter which does the formatting for us (`<div ng-bind="now | date:\'HH:mm:ss\'"></div>`). The updating is done through a timer with `$interval`. The important thing is to remove the timer when the directive is removed from the DOM. This is done by listening to the `$destroy` event which is called when the removal of the directive happens.

$interval is first available since Angular 1.2.

```
angular.module('cookbookApp', [])
  .directive('digitalClock', function($interval) {
    return {
      restrict: 'E',
      scope: {},
      template: '<div ng-bind="now | date:\'HH:mm:ss\'"></div>',
      link: function (scope) {
        scope.now = new Date();
        var clockTimer = $interval(function() {
          scope.now = new Date();
        }, 1000);

        scope.$on('$destroy', function(){
          $interval.cancel(clockTimer);
        });
      }
    };
  });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body>
  <digital-clock />
</body>
</html>
```

Code

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

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

7 Enable the save button only on valid form data

Problem

You have a form with a save button and you want to only enable it when the form data use typed is valid.

Solution

Not only single form fields but the whole form has its own states to test its validity. You can test for `$dirty` and `$valid`.

For example: If we a use a form with the name `form`, we could only enable the submit button if the user changes one of the fields (dirty) and if then all fields are valid.

```
<button ng-disabled="!(form.$dirty && form.$valid)">Save</button>
```

Be sure to give the form a name. Additionally, set `novalidate` as attribute and disable the browser's own validations.

Here's the full working example:

```
<html ng-app>
<head>
  <script src="../vendor/angular.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body>
  <form name="form" novalidate>
    <label>Login:</label> <input type="text" ng-model="login" required/>
    <button ng-disabled="!(form.$dirty && form.$valid)">Save</button>
  </form>
</body>
</html>
```

```
input.ng-invalid.ng-dirty {
  background-color: #ffc4d0;
}

input.ng-valid.ng-dirty {
  background-color: #d8ffd0;
}
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-disable-save-button

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-disable-save-button/

8 Disable trimming in input fields

Problem

You need the data from a textarea as it is on saving without the automatic trimming.

Solution

AngularJS makes good assumptions as default. For example, automatic trimming. If you don't want it, you can disable it with:

```
<textarea ng-model="bio" ng-trim="false"></textarea>
```

9 Dynamic service loading with the $injector

Problem

You have a lot of select boxes whose items are loaded from the server. You don't want to repeat yourself here. Instead of injecting the needed services into the controller, you just want the directive to handle it automatically.

Solution

Instead of letting the injector automatically inject the services by name in the directive definition, you trigger the injection process manually. To do this, you use the `$injector` instance.

In this example, we create a directive with the following attributes:

- *model:* Translates to ng-model in the select field
- *resource:* The service name and the function of the service you want to use separated by a dot.
- *resource-id:* The key in the returning object you want to use as options value.
- *resource-label:* The key in the returning object you want to use as options label.

The directive with looks like this:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <dynamic-select model="personId" resource="People.getList"
                  resource-id="id" resource-label="name" />
</body>
</html>
```

The directive `dynamicSelect` itself is not that complicated. We have an isolated scope where we `model`, `resourceId` and `resourceLabel`. `resource` is directly read through the attrs function parameters because we don't allow to dynamically change the service and we're not needing it in the template. (See recipe use the scope right TODO).

In the link function, we split the string from resource into two parts and write them into an object for better readability. Then we just use `$injector.get` to return the service as object. We then use the function as read into `params.fn`. We use `.then` here directly because we assume that our service function returns a promise.

```
angular.module('cookbookApp', [])
  .factory('People', function($http) {
    return {
      getList: function() { return $http.get('person.json'); }
    };
  })
  .directive('dynamicSelect', function($injector) {
    return {
      restrict : 'E',
      scope: {
        model: '=',
        resourceId: '@',
        resourceLabel: '@'
      },
      template: '<select ng-model="model" ng-options="item[resourceId] ' +
        'as item[resourceLabel] for item in items" />',
      link: function(scope, element, attrs) {
        var temp = attrs.resource.split('.');
        var params = { name: temp[0], fn: temp[1] };

        var service = $injector.get(params.name);

        service[params.fn]().then(function(serviceResponse) {
          scope.items = serviceResponse.data;
        });
      }
    };
  })
  .controller('MainController', function($scope) {
    $scope.personId = 2;
  });
```

This is example is greatly simplified. You would of course need error checking if you're consuming services other than $http.

Just for a complete example, the used JSON file.

```
[
  { "id": 1, "name": "John" },
  { "id": 2, "name": "Bill" },
  { "id": 3, "name": "Phil" }
]
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-dynamic-service-loading

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-dynamic-service-loading/

10 Create a dynamic templateUrl

Problem

You have several html templates which you want to use dynamically inside your directive.

Solution

Inside a directive, `templateUrl` cannot only take a string but also a function. The function has the element and the attributes as parameters.

In the following example, we dynamically fill a textarea with two different templates which are loaded dynamically. We write a directive which is named 'prefill'. This takes the name of the template as argument.

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body>
  <textarea prefill="filler1"></textarea>
  <textarea prefill="filler2"></textarea>
</body>
</html>
```

The directive itself consists only of the `templateUrl` with a function. Here we just take the argument from the `prefill` attribute and append `.html` to it to create the url.

```
angular.module('cookbookApp', [])
  .directive('prefill', function() {
    return {
      templateUrl: function(element, attrs) {
        return attrs.prefill + '.html';
      }
    };
  });
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-dynamic-template-url

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-dynamic-template-url/

11 Show a box on outstanding http requests (simple)

Problem

While you sending http request, you want to show a box or a loading spinner

Solution

As the solution here, we write a directive which shows an element if a condition is true. This could be a simple div with a string or a loading spinner. It doesn't matter where you place it in your DOM because the scope is isolated.

The way it works is, that it injects the `$http` because we need to know the count of the pending requests. After a request is started, the count changes and so does `$http.pendingRequests.length`. When it does, we get the result of `http.pendingRequests.length > 0` in the variable `value`. This is either `true` or `false`. We write the result in `$scope.waiting` and our div is visible or hidden depending of the state.

```
.directive('waitingForRequest', function($http) {
  var pendingRequests = function() {
    return $http.pendingRequests.length > 0;
  };
  return {
    restrict: 'E',
    scope: {},
    template: '<div ng-show="waiting">Waiting for request to finish...</div>',
    controller: function($scope) {
      $scope.$watch(pendingRequests, function(value) {
        console.log('Pending requests: '+ $http.pendingRequests.length);
        $scope.waiting = value;
      });
    }
  };
})
```

Discussion

This is a really simple but not the best solution. But for most applications it should be good enough. There are to issues here:

The watcher is executed every time an apply() is called. This means `$http.pendingRequests.length > 0` is called a lot of times. But for most applications it's does not mean a real performance hit.

12 Show only initialized data on startup

Problem

You start your not so small app and while AngularJS is booting up, your whole screen is crowded with curly braces.

Solution

First, the why. This situation can appear when open your `index.html`. The browser has to download all the javascript and css files and during this time, AngularJS hasn't parsed the page yet. So all over the page, you see curly braces for a short moment which are replaced when AngularJS is initialized.

To prevent this, you can use 2 directives: `ng-cloak` and `ng-bind`.

ng-clock

`ng-cloak` is a directive which removes itself when the application is initialized. If used with the following CSS, all elements are hidden during the initialization process.

```
[ng\:cloak], [ng-cloak], [data-ng-cloak], [x-ng-cloak], .ng-cloak, .x-ng-cloak {
  display: none !important;
}
```

ng-bind

If you don't want to hide some elements completely because then the layout is broken, you can also use `ng-bind`. `ng-bind` is similar to `{{}}` but replaces everything which is inside the tag where it is defined. If you would use curly braces and define a counter like this:

```
<span>{{counter}}</span>
```

You could also rewrite it with `ng-bind` to:

```
<span ng-bind="counter">?</span>
```

This has the advantage that a question mark is displayed as long the app isn't fully initialized. When the app is ready, the question mark is replaced with the real data.

ng-bind-template

If you want to use `ng-bind` but have more than one expression, you can use `ng-bind-template`. It's like `ng-bind` but can contain multiple expressions.

```
<span ng-bind-template="{{hello}} {{world}}">?</span>
```

13 Create a markdown live preview

Problem

You're using markdown somewhere in your application. Users who edit the markdown templates should see a live preview of the resulting html.

Solution

In this example, we implement the directive with the Showdown[1] library which is a popular markdown interpreter.

We create a directive 'markdownPreview' which has an isolated scope and takes a model as input attribute. As we create a new tag `<markdown-preview>`, we have to restrict it to element (E).

In the link function, we're watching for a change of the model attribute and convert the markdown input to an html output. The result is pasted into the elements' inner html.

```
angular.module('cookbookApp', [])
  .directive('markdownPreview', function() {
    var converter = new Showdown.converter();
    return {
      restrict: 'E',
      scope : {
        model: '='
      },
      link: function(scope, element) {
        scope.$watch('model', function(markdownText){
          var resultHtml = converter.makeHtml(markdownText || '');
          element.html(resultHtml);
        });
      }
    };
  });
```

[1] https://github.com/coreyti/showdown

```html
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="showdown/showdown.js"></script>
  <script src="application.js"></script>
</head>
<body>
  <textarea ng-model="markdown" cols="60" rows="10"></textarea>
  <markdown-preview model="markdown"></markdown-preview>
</body>
</html>
```

Discussion

Oftentimes, there are more solutions to a problem. If you want, you could also implement the markdown as a filter instead of directive. But be aware that you have to include the `ngSanitize` module and use `ng-bind-html`. Otherwise, your resulting html will be escaped.

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-markdown-live-preview

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-markdown-live-preview/

14 Table double rows with ng-repeat

Problem

You want to to repeat multiple elements which are siblings. Usually ng-repeat only repeat the current element and its children. See what's new in AngularJS 1.2.

Solution

Since AngularJS 1.2 there's a new optional syntax for `ng-repeat`. With it you can define a start and an end element for the repeater. The only restriction is that the element have to be siblings.

```
<table border="1">
    <tr ng-repeat-start="user in users">
        <td colspan="2">{{user.name}}</td>
    </tr>
    <tr ng-repeat-end>
        <td>{{user.age}}</td>
        <td>{{user.gender}}</td>
    </tr>
</table>
```

The source:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <table border="1">
    <tr ng-repeat-start="user in users">
      <td colspan="2">{{user.name}}</td>
    </tr>
    <tr ng-repeat-end>
      <td>{{user.age}}</td>
      <td>{{user.gender}}</td>
```

```
    </tr>
  </table>
</body>
</html>
```

<<(code/directives-ng-repeat-double-rows/style.css)

```
[
  { "name": "Bill", "age": 42, "gender": "male" },
  { "name": "John", "age": 52, "gender": "male" },
  { "name": "Anne", "age": 19, "gender": "female" },
  { "name": "Phil", "age": 21, "gender": "male" },
  { "name": "Mary", "age": 23, "gender": "female" }
]
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-ng-repeat-double-rows

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-ng-repeat-double-rows/

15 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">
```

16 Slide right/left with animations

Problem

You want to add slide effects for left and right to your ng-view div.

Solution

With a little trick, ng-animate will do all the heavy lifting for us. The idea is to set a variable for the direction before the url is changed. The animation direction depends on a css class which is altered when the direction changes.

Step 1: The styling

The first thing we need is some styling for the slide effects and a container to wrap them:

```
.container {
  position:relative; overflow:hidden;
  border: 1px solid black; height:100px;
}

.slide-left, .slide-right {
  position: absolute; top: 0; left: 0; right: 0; bottom: 0;
  width: 100%; padding: 10px;
}

.slide-left.ng-enter, .slide-left.ng-leave,
.slide-right.ng-enter, .slide-right.ng-leave {
  -webkit-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) .5s;
  -moz-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) .5s;
  -o-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) .5s;
  transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) .5s;
}

.slide-left.ng-enter { left: 100%; }
.slide-left.ng-enter.ng-enter-active { left: 0; }
.slide-left.ng-leave { left: 0; }
```

```
.slide-left.ng-leave.ng-leave-active { left: -100%; }

.slide-right.ng-enter { left: -100%; }
.slide-right.ng-enter.ng-enter-active { left: 0; }
.slide-right.ng-leave { left: 0; }
.slide-right.ng-leave.ng-leave-active { left: 100%; }
```

The container must have `position: absolute or relative`. The slide classes must have `position: absolute`. For left and right sliding, we define `slide-left` and `slide-right` with additional classes for ng-animate[1].

Step 2: The code

For demo purposes, we define two sample routes with direct template declaration with the `$routeProvider`. For the url change, we define two functions, `moveLeft` and `moveRight`, in a controller for the left and right slide. With `$location` service, we can change the url from a controller.

```
angular.module('cookbookApp', ['ngRoute', 'ngAnimate'])
  .config(function($routeProvider) {
    $routeProvider
      .when('/1', { template: '<h1>first page</h1>' })
      .when('/2', { template: '<h1>second page</h1>' })
      .otherwise({redirectTo: '/1' });
  })
  .controller('MainController', function($scope, $location) {
    $scope.moveLeft = function(href) {
      $scope.direction = 'left';
      $location.path(href);
    };

    $scope.moveRight = function(href) {
      $scope.direction = 'right';
      $location.path(href);
    };
  });
```

Since AngularJS 1.2, ngRoute is a module on its own. Be sure to include ngRoute and ngAnimate!

[1] http://docs.angularjs.org/api/ngAnimate.\protect\char"0024\relaxanimate

Step 3: The markup

In the app code itself we use the defined methods for the links with `ng-click="moveRight('/1')"`. Depending on the direction variable, `ng-view` gets the right animation with `ng-class="'slide-'+direction"`.

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="../vendor/angular-route.js"></script>
  <script src="../vendor/angular-animate.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" href="style.css"/>
</head>
<body ng-controller="MainController">
  <a href ng-click="moveRight('/1')">Page 1</a>
  <a href ng-click="moveLeft('/2')">Page 2</a>

  <div class="container">
    <div ng-view ng-class="'slide-'+direction"></div>
  </div>
</body>
</html>
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-ng-view-slide-animation

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-ng-view-slide-animation/

17 Pass a function to a directive with an isolated scope (&)

Problem

You have a directive with an isolated scope want to call a function in the outside scope. Or you to understand what the `&` in the isolated scope definition is doing.

Solution

While isolated scopes with `@` and `=` are easy understood by most people, a lot of them have a problem to understand what exactly the `&` is doing.

This is a basic concept but we repeat it here, because it is so often misunderstood.

As a general information. Every declaration of the scope isolation refers to an attribute of the element of the directive.

An declaration like

```
directive('colorpicker', function()  {
    ...
    scope: {
        a: '@',
        b: '=',
        c: '&'
    }
```

means the that you can use a, b and c inside the directive. The connection to the outer world is declared through attributes on the directive element. So the element would look like `<colorpicker a="4" b="myVar" c="myCallback()">`. This is important. You don't connect to the parent scope directly, but only through attribute declaration.

So, what is the `&` doing?

If you have the following situation:

```
+- controller scope
+--- directive scope
```

The directive is isolated and nested under the controller scope, you can execute a function on the controller scope from the directive scope.

Why is this important

This is important because you're free to choose your function name and which parameters you pass.

This means you could use `<colorpicker c="myCallback(paramA, paramB)">` in one controller and `<colorpicker c="mySuperCallback (paramB, paramA)">` in another.

How do I use it?

So, you want to call a function like this in the controller

```
$scope.myCallback = function(first, second) { ... }
```

The directive could have a execution of the function like this:

```
scope.c({ paramA: 123, paramB: 'xzy');
```

The you would have to use the directive in the template like this

```
`<colorpicker c="myCallback(paramA, paramB)">`
```

This means, every time the function is trigger inside the directive, it is translated to:

```
myCallback(123, 'xzy')
```

18 Select box with multiple option

Problem

You want to use a select box with the option `multiple`.

Solution

The `ng-options` directive of AngularJS does work well with the `multiple` option. You don't need an extra configuration for this. If `multiple` is present as an attribute, `ng-model` contains an array instead of a simple type.

The binding here is also bidirectional. Below is an example of how to use it. As you can see, you can set initial values just by declaring an array.

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <select multiple ng-model="selected"
          ng-options="person.id as person.name for person in people">
  </select>
  {{ selected }}
</body>
</html>
```

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope) {
    $scope.people = [
      { id:1, name:'John' },
      { id:2, name:'Bill' },
      { id:3, name:'Phil' }
    ];

    $scope.selected = [1,2];
  });
```

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-select-multiple/

19 Select and ng-options

Problem

You struggle with the instructions for `ng-options` given in the api.

Solution

Here we give examples for every way you can use `ng-options`. Source code ((((here))))

Array data sources

For all the examples, we use the following array:

```
users = [
    { id: '800', name: 'Bill Mayer', role: 'Admin' },
    { id: '801', name: 'Anne Black', role: 'User' }
];
```

What you'll see in all examples is that `ng-options` sets the values for arrays on it's own. You can't change the value of the options. `ng-options` translates them automatically.

label for value in array

Select tag with `ng-options`:

```
<select ng-model="arr1" ng-options="user.name for user in users">
```

The resulting options:

```
<option value="0" selected="selected">Bill Mayer</option>
<option value="1">Anne Black</option>
```

The contents of `ng-model`:

```
{"id":"800","name":"Bill Mayer","role":"Admin"}
```

select as label for value in array

Select tag with `ng-options`:

```
<select ng-model="arr2" ng-options="user.id as user.name for user in users">
```

The resulting options:

```
<option value="0" selected="selected">Bill Mayer</option>
<option value="1">Anne Black</option>
```

The contents of `ng-model`:

```
800
```

label group by group for value in array

Select tag with `ng-options`:

```
<select ng-model="arr3"
        ng-options="user.name group by user.role for user in users">
```

The resulting options:

```
<optgroup label="Admin">
    <option value="0">Bill Mayer</option>
</optgroup>
<optgroup label="User">
    <option value="1">Anne Black</option>
</optgroup>
```

The contents of `ng-model`:

```
{"id":"800","name":"Bill Mayer","role":"Admin"}
```

select as label group by group for value in array

Select tag with `ng-options`:

```
<select ng-model="arr4"
        ng-options="user.id as user.name group by user.role for user in
     users">
```

The resulting options:

```
<optgroup label="Admin">
    <option value="0">Bill Mayer</option>
</optgroup>
<optgroup label="User">
    <option value="1">Anne Black</option>
</optgroup>
```

The contents of `ng-model`:

```
800
```

Object data sources

For all the examples we use the following object:

```
roles = {
    150: { name: 'Admin', rights: 'Read+Write' },
    151: { name: 'User',  rights: 'Read' }
};
```

label for (key, value) in object

Select tag with `ng-options`:

```
<select ng-model="obj1" ng-options="obj.name for (roleId, obj) in roles">
```

The resulting options:

```
<option value="150" selected="selected">Admin</option>
<option value="151">User</option>
```

The contents of `ng-model`:

```
{"name":"Admin","rights":"Read+Write"}
```

select as label for (key , value) in object

Select tag with `ng-options`:

```
<select ng-model="obj2" ng-options="id as obj.name for (id, obj) in roles">
```

The resulting options:

```
<option value="150" selected="selected">Admin</option>
<option value="151">User</option>
```

The contents of `ng-model`:

```
150
```

label group by group for (key, value) in object

Select tag with `ng-options`:

```
<select ng-model="obj3"
        ng-options="obj.name group by obj.rights for (id, obj) in roles">
```

The resulting options:

```
<optgroup label="Read+Write">
    <option value="150">Admin</option>
</optgroup>
<optgroup label="Read">
    <option value="151">User</option>
</optgroup>
```

The contents of `ng-model`:

```
{"name":"Admin","rights":"Read+Write"}
```

select as label group by group for (key, value) in object

Select tag with `ng-options`:

```
<select ng-model="obj4"
        ng-options="id as obj.name group by obj.rights for (id, obj) in roles">
```

The resulting options:

```
<optgroup label="Read+Write">
    <option value="150">Admin</option>
</optgroup>
<optgroup label="Read">
    <option value="151">User</option>
</optgroup>
```

The contents of `ng-model`:

```
150
```

Custom empty option

If you don't want to show a blank field but your own custom label, you can just insert an option tag with a blank value.

```
<select ng-model="choose" ng-options="user.name for user in users">
    <option value="">- Choose user -</option>
</select><br>
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-select-ng-options

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-select-ng-options/

20 Make a sortable table

Problem

You want to make your table sortable. The element in the header which the table is ordered by is highlighted. If you click again on the active element the sort order is reversed.

Solution

For this we write a directive `thSortable` which you can later use like this:

```
<th th-sortable="sort" th-column="name" th-default>Name</th>
```

The directive wraps the contents of an th tag into a link. The link gets a class `active` if it is the current selected sort order. If the element is already active and you click the link again the order is reversed.

We isolate the scope of the directive and set two variables:

```
angular.module('cookbookApp', [])
  .directive('thSortable', function() {
    return {
      transclude: true,
      template: '<a href ng-click="changeColumn()" ng-class="{' +
        'active: sort.column == column, ' +
        'asc: !sort.reverse, ' +
        'desc: sort.reverse' +
        '}">' +
        '<span ng-transclude></span></a>',
      scope : {
        sort: '=thSortable',
        column: '@thColumn'
      },
      controller: function($scope, $attrs) {
        $scope.sort = $scope.sort || {};

        if (angular.isDefined($attrs.thDefault)) {
```

```
          $scope.sort.column = 'name';
        }

        $scope.changeColumn = function() {
          if ($scope.sort.column === $scope.column) {
            $scope.sort.reverse = !$scope.sort.reverse;
          } else {
            $scope.sort = { column: $scope.column, reverse: false };
          }
        };
      }
    };
  })
  .controller('MainController', function($scope, $http) {
    $http.get('users.json').then(function(usersResponse) {
      $scope.users = usersResponse.data;
    });
  });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body ng-controller="MainController">
<table>
  <tr>
    <th th-sortable="sort" th-column="name">Name</th>
    <th th-sortable="sort" th-column="age" >Age</th>
    <th th-sortable="sort" th-column="gender" th-default>Gender</th>
  </tr>
  <tr ng-repeat="user in users | orderBy:sort.column:sort.reverse">
    <td ng-bind="user.name"></td>
    <td ng-bind="user.age"></td>
    <td ng-bind="user.gender"></td>
  </tr>
</table>
</body>
</html>
```

We use stylesheets to highlight the current column and show the order. Because we set .asc or .desc as class for the direction, we can use the the css content attribute and append and arrow.

```
th a {
  display: block;
  text-decoration: none;
}

th a.active {
  background-color: #ccc;
}

th a.active.asc:after {
  content: "\2193";
}

th a.active.desc:after {
  content: "\2191";
}
```

Just for demo purposes the json file with the demo data.

```
[
  { "name": "Bill", "age": 42, "gender": "male" },
  { "name": "John", "age": 52, "gender": "male" },
  { "name": "Anne", "age": 19, "gender": "female" },
  { "name": "Phil", "age": 21, "gender": "male" },
  { "name": "Mary", "age": 23, "gender": "female" }
]
```

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-sortable-table/

21 Make a stacked bar chart

Problem

You want to create a chart with stack bars.

Solution

We solve it through a directive which takes an array of percentages and draws the chart.

For the bars, we define an array of integers in the controller. In the directive, it's enough to just use a template with an `ng-repeat`. There's one trick here. `ng-repeat` doesn't allow duplicates. If we would just use `ng-repeat="bar in bars"` and use an array like `[20,20,40]`, AngularJS would throw an error like `Duplicates in a repeater are not allowed`. We solve this by using `track by $index"` and force uniqueness.

```
.stacked-bar-chart {
  overflow: hidden;
  margin: 10px 0;
  height: 20px;
  border: 1px solid #ccc;
  background-color: #f9f9f9;
}

.stacked-bar   { float: left; height: 100%; }
.stacked-bar-0 { background-color: #bb0000; }
.stacked-bar-1 { background-color: #00bb00; }
.stacked-bar-2 { background-color: #0000bb; }
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body ng-controller="MainController">
  <stacked-bar bars="bars"></stacked-bar>

  <input type="number" ng-model="bars[0]" />
  <input type="number" ng-model="bars[1]" />
  <input type="number" ng-model="bars[2]" />
</body>
</html>
```

```
angular.module('cookbookApp', [])
  .directive('stackedBar', function() {
    return {
      restrict: 'E',
      template:   '<div class="stacked-bar-chart">' +
        '<div ng-repeat="bar in bars track by $index" ' +
        'class="stacked-bar stacked-bar-{{$index}}" ' +
        'ng-style="{width: bar+\'%\'}">' +
        '</div>' +
        '</div>',
      scope : {
        bars: '='
      }
    };
  })
  .controller('MainController', function($scope) {
    $scope.bars = [30,30,40];
  });
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-stacked-bar-chart

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-stacked-bar-chart/

22 Prevent event propagation from ng-click

Problem

You have several nested elements with `ng-click`. If you click the front element, you don't want the click function of the back element to be triggered.

Solution

The trick here is to use `$event` which is injected into `ng-click`. `$event` is the normal JavaScript event object and thus you can use `stopPropagation()` to prevent the bubbling of events.

You pass `$event` as parameter of the topmost `ng-click` and call `event.stopPropagation()` in the function declaration inside of the controller.

```
<div ng-click="back()">
    ...
    <a ng-click="front($event)">Click me!</a>
    ...
</div>
```

Alternative solution

You can of course use `$event` directly inside the `ng-click` definition.

```
<div ng-click="back()">
    ...
    <a ng-click="front(); $event.stopPropagation()">Click me!</a>
    ...
</div>
```

23 Submit form on enter key

Problem

Usually, if you press enter in a form field, the form is submitted to the action defined in the ´<form>´ tag but you want to call a function in your controller.

Solution

The solution is to leave out the `action` attribute in `<form>` and use `ng-submit`. The expression in `ng-submit` is what gets executed when you press return inside a form field.

An complete example:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body ng-controller="MainController">
  <form name="form" ng-submit="addTask(taskName)" novalidate>
    <input type="text" ng-model="taskName" />
  </form>

  <ul>
    <li ng-repeat="task in tasks track by $index" ng-bind="task"></li>
  </ul>
</body>
</html>
```

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

    $scope.addTask = function(taskName) {
      $scope.tasks.push(taskName);
    };
  });
```

We use `track by $index` in `ng-repeat` here. This is a hack to allow duplicate entries in your list. Try to see what happens if you leave it out.

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-submit-form-on-enter

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-submit-form-on-enter/

24 Make a syntax highlighter

Problem

You want to make a syntax highlighter but your code inside the code block is interpreted by AngularJS.

Solution

As an example, we take the following code:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="prism/prism.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="prism/prism.css"/>
</head>
<body>
  <pre><code class="language-markup">
    <ul>
      <li ng-repeat="task in tasks">{{ task }}</li>
    </ul>
    <button ng-click="add()">Add task</button>
  </code></pre>
</body>
</html>
```

If we look at the code block, there are several problems we have to tackle.

- Start: From a html perspective, the code snippet starts right behind `code class="language-markup">` but we don't want to insert an empty line before the code snippet.
- Indention: For reasons of readability, we want the indention in the html file to be 4 spaces. But when we present it to the user, we want it to start at 0.
- Angle brackets: We don't want to paste our code like '>pre<. We have to convert it.
- AngularJS: We don't want AngularJS to interpret the code inside the code block because if we would use an expression, it would be interpreted by AngularJS.

As a syntax highlighter, we use Prism[1] by Lea Verou. It's used by a lot of well-known companies for their sites.

```
angular.module('cookbookApp', [])
  .directive('code', function() {
    function escapeAngleBrackets(text) {
      return text.replace(/</gi, '&lt;').replace(/>/gi, '&gt;');
    }
    function trimSurroundingEmptyLines(text) {
      return text.replace(/^(\n)*/, '').replace(/(\n)*(\s)*$/, '');
    }
    function fixIndention(text) {
      return text.replace(
        new RegExp('^ {' + text.search(/[^\s-]/) + '}', 'gm'), '');
    }
    return {
      restrict: 'E',
      terminal: true,
      link: function(scope, element) {
        var content = element.html();
        content = escapeAngleBrackets(content);
        content = trimSurroundingEmptyLines(content);
        content = fixIndention(content);
        element.html(content);
        Prism.highlightElement(element.get(0));
      }
    };
  });
```

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-syntax-highlighter/

[1] http://prismjs.com

25 Textarea char limit with remaining counter

Problem

You want a counter for remaining chars for your textarea

Solution

We create a directive which appends an counter element after the textarea. To get the value of `ng-model`, we require the `ngModel` controller which is the preferred way if there's a `ng-model`.

For the counter of the remaining characters we create a template in a variable `counterTemplate`. for this template we create a new fresh scope `counterScope` and compile the template against this scope. We then append it after the textarea element with `element.after(...)`.

Everytime there's a change in the model, the parser chain of the `ngModel` controller is called which we use. There we just calculate the current length, truncate it when it's to long and refresh the variable `remaining` on the `counterScope`.

```
angular.module('cookbookApp', [])
  .directive('maxlengthCounter', function ($compile) {
    var counterTemplate = '<p>Remaining charaters: {{remaining}}</p>';
    return {
      require:'ngModel',
      link:function (scope, element, attrs, ngModelCtrl) {
        var maxLength = parseInt(attrs.maxlengthCounter, 10);

        var counterScope = scope.$new();
        counterScope.remaining = maxLength;
        element.after($compile(counterTemplate)(counterScope));

        ngModelCtrl.$parsers.push(function (value) {
          var currentLength = parseInt(value.length, 10);

          if (currentLength > maxLength) {
            element.val(value.substr(0, maxLength));
```

```
            currentLength = maxLength;
          }
          counterScope.remaining = maxLength - currentLength;
        });
      }
    };
  });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body>
  <textarea ng-model="result" maxlength-counter="16"></textarea>
  <p>{{result}}</p>
</body>
</html>
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-textarea-maxlength-counter

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-textarea-maxlength-counter/

26 Theme support

Problem

You want to support different themes which you can change on the fly.

Solution

Overview:

- Create a controller for the `<head>` tag which changes the href to your stylesheet on the fly.
- Create a service which contains the active theme and supports the changing of it.
- Inject the service where you want to change the theme.

The code:

First we create a `ThemeService` service , a `HeadController` controller and a `MainController` controller.

- `ThemeService` holds the theme state
- `HeadController` exposes getTheme() to change the stylesheet link in the `<head>` tag
- `MainController` acts as an example for a controller which can change the theme.

```
angular.module('cookbookApp', [])
  .factory('ThemeService', function(){
    var themes = { available: ['default', 'greenish'], active: 'default' };

    function getTheme() { return themes.active; }

    function setTheme(name) {
      if (themes.available.indexOf(name) === -1) { return; }
      themes.active = name;
    }

    return {
      getTheme: getTheme,
```

```
      setTheme: setTheme
    };
  })
  .controller('HeadController', function($scope, ThemeService) {
    $scope.getTheme = ThemeService.getTheme;
  })
  .controller('MainController', function($scope, ThemeService) {
    $scope.setTheme = ThemeService.setTheme;
  });
```

In the `index.html` we use `<link rel="stylesheet" ng-href="theme-{{getTheme()}}.css"/>` to replace the stylesheet on the fly.

```
<html ng-app="cookbookApp">
<head ng-controller="HeadController">
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="theme-{{getTheme()}}.css"/>
</head>
<body ng-controller="MainController">
  <h1>Hello World</h1>
  <p>
    Select theme:
    <a href ng-click="setTheme('default')">Default</a>
    <a href ng-click="setTheme('greenish')">Greenish</a>
  </p>
</body>
</html>
```

The style are just examples to have a working version.

theme-default.css

```
body { background-color: #ccc; }
h1   { color: #333; }
```

theme-greenish.css

```
body { background-color: #eeffee; }
h1   { color: #005500; }
```

Read more about communication between controller in Recipe [controller-to-controller].

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-theme-support/

27 Use the inner html of a directive in your template

Problem

You write a directive and want to get the inner html and use it in your template.

Solution

The solution is to activate and use transclusion. You activate transclusion in your directive if you use `transclude: true` in your returning object. Now AngularJS captures what was inside your directive. You can then use the contents with `<div ng-transclude></div>` which is replaced with the captured content.

```
.directive('...', function() {
    return {
        transclude: true,
        template: '<div id="wrapper"><div ng-transclude></div></div>'
    }
});
```

28 Write a blacklist validator

Problem

You want to make sure that an input field or textarea does not contain a set of words from a blacklist.

Solution

For this we have to write a custom validator. A validator is nothing else but a directive which includes a special controller.

As an example we take the following excerpt from the `index.html`:

```
<input type="text" name="name" ng-model="name" blacklist="blacklistValues" />
```

The attribute `blacklist` is our custom validator which you pass an array of your blacklist. Most of the work is done by the `ngModelController`. The interesting part is how to get the array from the attribute. AngularJS has a service called `$parse` which can evaluate expressions. If we had an isolated scope, this is what the `@` sign does.

So we get the blacklist with $parse. We then can use one of the new [ES5 array functions] (#big-picture-es5-array-functions) to check if the model contains one of the element in the blacklist array. If it does we set the field invalid.

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body ng-controller="MainController">
  <p>Try <strong>hello</strong> or <strong>bye</strong>.</p>
  <form name="form" novalidate>
    <input type="text" name="name"
           ng-model="name" blacklist="blacklistValues" />
  </form>

  <div ng-show="form.name.$dirty">
```

```
    <span ng-show="form.name.$error.blacklist">
      Your name contains a word from the blacklist.
    </span>
  </div>
</body>
</html>
```

```
angular.module('cookbookApp', [])
  .directive('blacklist', function ($parse) {
    return {
      require:'ngModel',
      link: function (scope, element, attrs, ngModelCtrl) {
        var badWords = $parse(attrs.blacklist)(scope) || [];
        ngModelCtrl.$parsers.push(function (value) {
          if (value) {
            var containsBadWord = badWords.some(function(str) {
              return value.indexOf(str) >= 0;
            });
            ngModelCtrl.$setValidity('blacklist', !containsBadWord);
          }
        });
      }
    };
  })
  .controller('MainController', function($scope) {
    $scope.blacklistValues = ['hello', 'bye'];
  });
```

```
input.ng-invalid.ng-dirty {
  background-color: #ffc4d0;
}

input.ng-valid.ng-dirty {
  background-color: #d8ffd0;
}
```

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-validator-blacklist/

29 General purpose uniqueness validator

Problem

You want to check if the input in an form field is unique, e.g. a login box.

Solution

For this we have to write a custom validator. A validator is nothing else but a directive which includes a special controller.

As an example, we take the following excerpt from the `index.html`:

```
<input type="text" name="login" ng-model="login"
       unique="checkUniqueLogin" required />
```

The attribute `unique` will be our directive and `checkUniqueLogin` is a function that has to exist on the current scope. This function is called with the value of `login` and returns a promise with either `true` or `false` depending on the result of the check.

The interesting part here is the `unique` directive. The first thing is that we require `ngModel`. This means that we expect a `ng-model` attribute in the same tag and want to share its controller. We then can access the controller in the link function as the 4th parameter which we call `ngModelCtrl`. The `ngModelCtrl` has a set of helper function for form validation (Learn more about the ng-model controller here[1]). In the `$parsers` function we check if the current value has been changed. If it has changed, we call the function which we passed to the `unique` attribute with value as parameter and expect a promise a return value.

We can call a method from the controller here because the scope is not isolated. Now we just have to make sure that the `checkUniqueLogin` in the controller returns a promise which is either true or false.

[1] TODO

```html
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body ng-controller="MainController">
  <p>Try <strong>bloodymary</strong> as login.</p>

  <form name="form" novalidate>
    <label>Login:</label>
    <input type="text" name="login" ng-model="login"
           unique="checkUniqueLogin" required/>
  </form>

  <div ng-show="form.login.$dirty && form.login.$invalid">Invalid:
    <span ng-show="form.login.$error.required">Choose a login.</span>
    <span ng-show="form.login.$error.unique">Login already taken.</span>
  </div>
</body>
</html>
```

```javascript
angular.module('cookbookApp', [])
  .directive('unique', function () {
    return {
      require:'ngModel',
      link: function (scope, element, attrs, ngModelCtrl) {
        var original;
        ngModelCtrl.$formatters.unshift(function(modelValue) {
          original = modelValue;
          return modelValue;
        });
        ngModelCtrl.$parsers.push(function (value) {
          if (value && value !== original) {
            scope[attrs.unique](value).then(function(result) {
              ngModelCtrl.$setValidity('unique', result);
            });
            return value;
          }
        });
      }
    };
```

```
  })
  .controller('MainController', function($scope, $http) {
    $scope.checkUniqueLogin = function(value) {
      return $http.get('users.json').then(function(usersResponse) {
        return !usersResponse.data.filter(function(user){
          return user.login === value;
        }).length;
      });
    };
  });
```

```
[
  { "name": "John", "login": "johndoe" },
  { "name": "Anne", "login": "awesomeanne" },
  { "name": "Phil", "login": "justphil" },
  { "name": "Mary", "login": "bloodymary" }
]
```

```
input.ng-invalid.ng-dirty {
  background-color: #ffc4d0;
}

input.ng-valid.ng-dirty {
  background-color: #d8ffd0;
}
```

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-validator-uniqueness/

30 Forms with view / edit mode

Problem

You have a form but don't want to allow editing directly. You need something like a view mode which can be switched to edit mode.

Solution

The solution is to solve it mainly through styling. We change the styling for the disabled state of an element like it's normal text. Then we create a directive `editMode` which toggles the styling class and adds / removes the disabled attribute.

```
angular.module('cookbookApp', [])
  .directive('editMode', function() {
    return function(scope, element) {
      scope.$watch('editMode', function() {
        if (scope.editMode) {
          element.removeClass('viewable')
            .addClass('editable')
            .removeAttr('disabled');
        } else {
          element.removeClass('editable')
            .addClass('viewable')
            .attr('disabled','disabled');
        }
      });
    };
  })
  .controller('MainController', function($scope) {
    $scope.editMode = false;
    $scope.name = 'AngularJS';
  });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body ng-controller="MainController">
  <label><input type="checkbox" ng-model="editMode"> edit mode</label>
  <p>
    <input type="text" ng-model="name" edit-mode />
  </p>
</body>
</html>
```

```
input.viewable { border-color: transparent; }
input.editable { color: #000; }
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/directives-view-edit-mode

Online demo: http://sbrink.github.io/angularjs-cookbook-code/directives-view-edit-mode/

II Controller Recipes

31 All / none / invert for a list of checkboxes

Problem

You have a list of check boxes and you want to give the user an easy way to select all, none or invert the states. Buttons for these options should be disabled if not possible.

Solution

For this example, we create an array of objects. An object consists of a boolean field `completed` which represents the state of the checkbox and a `title` field (`[{ completed: ..., title: ...}, ...]`).

For the buttons, we need the count of all completed tasks. We create a new variable `completedTasksCount` on the scope and update it on every change of the array. To do this, we create a watcher with `$scope.$watch('tasks', ..., true);`. The `true` as a third parameter is important to signal AngularJS to do a deep watch and inspect all the object in the arrays.

With the count of the all completed tasks, we can disable the button for select all if all tasks already checked with `ng-disabled="completedTasksCount == tasks.length"`. The implementation for select none works similarly.

For `selectAllTasks()`, `deselectAllTasks()` and `invertAllTasks()`, we use the each function from AngularJS to iterate over the array and modify the state of completed.

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope) {
    $scope.tasks = [
      { completed: true,  title: 'Wash dishes' },
      { completed: false, title: 'Tidy up' },
      { completed: false, title: 'Do laundry' }
    ];

    $scope.$watch('tasks', function(tasks) {
      $scope.completedTasksCount = tasks.filter(function(task) {
        return task.completed;
      }).length;
```

```
13      }, true);
14
15      $scope.selectAllTasks = function() {
16        angular.forEach($scope.tasks, function(task) {
17          task.completed = true;
18        });
19      };
20
21      $scope.deselectAllTasks = function() {
22        angular.forEach($scope.tasks, function(task) {
23          task.completed = false;
24        });
25      };
26
27      $scope.invertAllTasks = function() {
28        angular.forEach($scope.tasks, function(task) {
29          task.completed = !task.completed;
30        });
31      };
32    });
```

```
1  <html ng-app="cookbookApp">
2  <head>
3    <script src="../vendor/angular.js"></script>
4    <script src="application.js"></script>
5  </head>
6  <body ng-controller="MainController">
7    <button ng-click="selectAllTasks()"
8        ng-disabled="completedTasksCount == tasks.length">Select all</button>
9    <button ng-click="deselectAllTasks()"
10       ng-disabled="completedTasksCount == 0">Select none</button>
11   <button ng-click="invertAllTasks()">Invert</button>
12
13   <ul>
14     <li ng-repeat="task in tasks">
15       <input type="checkbox" ng-model="task.completed"/> {{ task.title }}
16     </li>
17   </ul>
18
19   <p>Completed: {{ completedTasksCount }}</p>
20 </body>
21 </html>
```

32 Controller to controller communication

Problem

You have two controllers which are not nested and want to exchange data between them.

Solution

The solution is to use a service for communication. This way you can use all the benefits of two-way-databinding.

We use the piece of html as an example:

```
<div ng-controller="FirstController">
  <ul>
    <li ng-repeat="item in items" ng-bind="item"></li>
  </ul>
</div>

<div ng-controller="SecondController">
  <ul>
    <li ng-repeat="item in items" ng-bind="item"></li>
  </ul>
  <button ng-click="addItem()">Add item</button>
</div>
```

We have two controllers here: `FirstController` and `SecondController` which are siblings. We use the variable `items` for demonstration purposes here.

To share data, we create a service `ItemsService`. `items` is an internal variable which we expose through a function `getItems()`. After we injected the `ItemsService` in both controllers, we can now begin to use it on the scope. After writing `$scope.items = ItemsService.getItems();` in both controller, we can use the items on both controller views and they stay in sync. Open the link to the demo and try to add items.

You see another working version in Theme support and Learn more about [structuring services](#services-how-to-structure).

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/controllers-communication

Online demo: http://sbrink.github.io/angularjs-cookbook-code/controllers-communication/

33 Use your view filters in your controller (quick)

Problem

You want to localize a string in your controller. Because there's no extra service for this, you can you use the date filter.

Solution

You can just inject the `$filter` service and use the view filter. You call the filter with the name of the filter and it will return a function. The first parameter of the returning function is always the input. If the filter has options, they are the following parameters.

Example:

```
$filter('date')(input, options)
```

34 Reset a form

Problem

You have a form and want to reset it to the initial state.

Solution

We have to save the initial form state ourselves. We do this by creating a new variable `var initialFormData;` and assign the original data with `initialFormData = formResponse.data`. If we would just use `scope.form = initialFormData`, we would hand over the reference and it a change in `$scope.form` would also result in a direct change in `initialFormData`. To prevent this, we have to separate them and work with a copy. AngularJS brings a convenient method we can use called `angular.copy`. So every time we need the initial state, we just use `$scope.form = angular.copy(initialFormData);`

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <input type="text" ng-model="form.firstName" />
  <input type="text" ng-model="form.lastName" />

  <button ng-click="reset()">Reset</button>
</body>
</html>
```

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope, $http) {
    var initialFormData;

    $http.get('person.json').then(function(formResponse) {
      initialFormData = formResponse.data;
      $scope.form = angular.copy(initialFormData);
    });

    $scope.reset =function() {
      $scope.form = angular.copy(initialFormData);
    };

  });
```

```
{
  "firstName": "John",
  "lastName": "Doe"
}
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/controllers-form-reset

Online demo: http://sbrink.github.io/angularjs-cookbook-code/controllers-form-reset/

35 How to use the same function for multiple watchers

Problem

You want to recalculate some function on the changing of different watches. You don't want to copy it several times.

Solution

AngularJS has a not so obvious solution for it. You can specify multiple watchers in an array like syntax. Instead of returning single variables for the new and old value, you get arrays.

Here's the code:

```
$scope.$watch('[first, second, third]', function(newArray, oldArray){
    console.log(newArray[0]); // new value of first
    console.log(oldArray[0]); // old value of first

    console.log(newArray[1]); // new value of second
    console.log(oldArray[1]); // old value of second

    console.log(newArray[2]); // new value of third
    console.log(oldArray[2]); // old value of third
},true);
```

III Service Recipes

36 Get current app name (quick)

Problem

You use several `ng-app` on your page and need the current app you're in.

Solution

You just have to inject `$rootElement`. This represents the root element of application so you just have to read out the `ng-app` attr. Works with multiple apps on a page.

Working version:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  Your application name is: {{appName}}
</body>
</html>
```

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope, $rootElement) {
    $scope.appName = $rootElement.attr('ng-app');
  });
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/services-current-app-name

Online demo: http://sbrink.github.io/angularjs-cookbook-code/services-current-app-name/

37 Prevent heavy computing operations from making your app sluggish

Problem

You have an operation which takes some time. Your app is sluggish oder the tab kills itself.

Solution

The first thing is to understand why apps become sluggish in the browser.

1. JavaScript is single-threaded. This means a single event queue. New events are appended at the end of this queue.
2. JavaScript is 'greedy'. This means it tries to execute as much code as possible.

We use the following example with bogosort (inefficient sorting algorithm) as an example and create two versions. One version is blocking and the other one is not)

```
angular.module('cookbookApp', [])
  .service('Bogosort', function($rootScope){
    function shuffleArray(v) {
      // http://jsfromhell.com/array/shuffle
      for(var j, x,
            i = v.length; i;
            j = parseInt(Math.random()*i, 10), x=v[--i], v[i]=v[j], v[j]=x
      );
    }
    function isSorted(v){
      for(var i=1; i<v.length; i++) {
        if (v[i-1] > v[i]) { return false; }
      }
      return true;
    }
```

```
16      function blockingSort(input) {
17        while(!isSorted(input)){
18          shuffleArray(input);
19        }
20      }
21      function nonBlockingSort(input) {
22        function next() {
23          var counter = 0;
24          console.log('Shuffle...');
25
26          while(!isSorted(input) && counter++ < 1000){
27            shuffleArray(input);
28          }
29
30          if (isSorted(input)) {
31            $rootScope.$apply();
32          } else {
33            setTimeout(next, 0);
34          }
35        }
36        setTimeout(next, 0);
37      }
38      return {
39        nonBlockingSort: nonBlockingSort,
40        blockingSort: blockingSort
41      };
42    })
43    .controller('MainController', function($scope, Bogosort) {
44      $scope.items = [];
45      for(var i=1; i<11; i++) {
46        $scope.items.push(Math.random());
47      }
48      $scope.sortBlocking = Bogosort.blockingSort;
49      $scope.sortNonBlocking = Bogosort.nonBlockingSort;
50    });
```

```html
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <button ng-click="sortBlocking(items)">Sort blocking</button>
  <button ng-click="sortNonBlocking(items)">Sort non blocking</button>
  <ul>
    <li ng-repeat="item in items" ng-bind="item"></li>
  </ul>
</body>
</html>
```

If we now do the following:

- Mouse enter button
- Click button
- Mouse leave button

Blocking version

The event queue looks like:

- highlight button
- heavy computation
- unhighlight button

Non-blocking version

The event queue looks like:

- highlight button
- heavy computation (999x shuffle)
- unhighlight button
- heavy computation (999x shuffle)
- …

Code

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

Online demo: http://sbrink.github.io/angularjs-cookbook-code/services-heavy-computations/

38 How to structure your services

Problem

JavaScript has no build in concept of private/public methods. Your services should have a public api but helper methods should be private. Here we show how to do it.

Solution

Here we take a factory service `Task` as an example for serveral important design decisions.

Return an object and get privacy

In JavaScript you can't mark methods as private/public, but you can simulate this behaviour through closures. Inside the factory, we have three methods: `inRange`, `find` and `all`. But `inRange` is just a helper method and we don't want to expose it. The way to accomlish this, is to return an object where we create properties with references to the methods we want to expose. Through closures we have access to the methods in the factory.

```
return {
    all: all,
    find: function(i) {
        return find(i);
    }
};
```

Expose the API 1

Instead of

```
return {
    all: all,
    find: function(i) {
        return find(i);
    }
};
```

We could have also done this:

```
var srv = {};
srv.all = function() { ... }
srv.find = function(i) { ... }
return srv;
```

This would remove some duplication, but also has a disadvantace. You don't have an overview over the api in one place . Because of this a new developer has first to go through all the noise of the code, scroll up/down to identify the interface.

Expose the API 2

In this example, we see two possiblities to expose the inner methods.

```
return {
    all: all,
    find: function(i) {
        return find(i);
    }
};
```

For `all` we set a direct reference the function. For `find`, we first create an anonymous function and inside it, we return the find method. Why two different possibilities and which one is correct?

The answer is: Both are correct. But both have strengths and weaknesses.

If you choose the first one, you expose the method but not the parameters. So if you want to use the method, you have to look up the parameters seperately.

If you choose the second one, you can see how to use the function immediately but have to type some more and change it if you change the function definition. There's also one caveat. If you ask for `arguments.length` inside your function, the second method will break.

Don't expose data structures directly

To get the tasks array, you see that we don't return a reference to the task array, but a copy. This is good practice and urges the developer to use the defined methods in the service instead of operating on the data structure directly.

```
var tasks = ['Tidy up'];
function all() {
    return angular.copy(tasks);
}
```

Full demo code

```
angular.module('cookbookApp', [])
  .factory('Task', function () {
    var tasks = ['Tidy up'];
    function inRange(i) {
      return i <= tasks.length-1;
    }
    function all() {
      return angular.copy(tasks);
    }
    function find(i) {
      return inRange(i)? tasks[i] : 'Not in range';
    }
    return {
      all: all,
      find: function(i) {
        return find(i);
      }
    };
  })
  .controller('MainController', function($scope, Task) {
    $scope.tasks = Task.all();
    $scope.task = Task.find(0);
  });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <p>{{tasks}}</p>
  <p>{{task}}</p>
</body>
</html>
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/services-how-to-structure

Online demo: http://sbrink.github.io/angularjs-cookbook-code/services-how-to-structure/

39 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/

40 Notification service delayed / sticky

Problem

You want to give your user feedback about success / failure of an operation with notifications which close themselves after a short period.

Solution

The adding/removing of notifications is solved by a service. The service holds an array of objects which are displayed by `ng-repeat` in the view. If you add a notification, there's also a close event after a certain timeout added.

For the implementation, we need a unique identifier for each list item. We can't rely on this position inside the array because of the dynamic nature of the list. So we use a global counter which is always incremented by one with every added notification.

```
angular.module('cookbookApp', [])
  .factory('NotificationService', function($timeout) {
    var globalCounter = 0, list = [];

    function getCounter() { return globalCounter += 1; }
    function getList() { return list; }

    function add(text, sticky, timeout) {
      var counter = getCounter();
      list.unshift({ id: counter, text: text});
      if (!sticky) {
        $timeout(
          function(){ remove(counter); },
          (timeout || 3000)
        );
      }
    }

    function remove(id){
```

```
      for (var i=0; i<list.length; i++) {
        if (list[i].id === parseInt(id, 10)) {
          return list.splice(i, 1);
        }
      }
    }

    return {
      add: add,
      remove: remove,
      getList: getList
    };
  })
  .controller('NotificationsController', function($scope, NotificationService) {
    $scope.notifications = NotificationService;
  })
  .controller('MainController', function($scope, NotificationService) {
    $scope.addNotification = function(sticky) {
      NotificationService.add(new Date(), sticky, 1000);
    };
  });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
  <link rel="stylesheet" ng-href="style.css"/>
</head>
<body>
  <div ng-controller="MainController">
    <button ng-click="addNotification(false)">Timed notification</button>
    <button ng-click="addNotification(true)">Sticky notification</button>
  </div>
  <ul class="notifications" ng-controller="NotificationsController">
    {{notificationList}}
    <li ng-repeat="notification in notifications.getList()"
        class="animate-repeat">
      <a href ng-bind="notification.text"
         ng-click="notifications.remove(notification.id)"></a>
    </li>
  </ul>
```

```
</body>
</html>
```

```
.notifications {
  margin: 0;
  position: absolute;
  right: 0; bottom: 0;
  list-style: none;
}
.notifications li a {
  display: block;
  border: 1px solid #ccc;
  background-color: #f9f9f9; color: #333;
  font-size: 80%;
  text-decoration: none;
  padding: 20px 10px;
  text-align: center;
  width: 200px;
}
```

Discussion

Of course it's also possible to solve it with a directive.

Code

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

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

41 Why is there a Provider at the end of some services ($route and $routeProvider)?

Problem

You're confused why sometimes there exist two services with the difference that one has `Provider` at the end of the name.

Solution

In short: If you take $route and $routeProvider as example, `$route` is the service and `$routeProvider` is provider function. You use `$routeProvider` to configure the service.

In AngularJS, the injector is responsible for instanciating services. But sometimes you want to set some options before the injector does its work and instanciate it.

We take a simplified version for the $route service as an example:

```
angular.module('ngRoute', ['ng']).provider('$route', $RouteProvider);

function $RouteProvider(){

    this.when = function(path, route) { ... }
    this.otherwise = function(params) { ... }

    this.$get = function() { ... }
};
```

To configure routes, you use `when` and `otherwise` in the config block of your module. This is done before the injector instanciated the route service. You can't change these settings later in a controller for example.

The contents defined in the `$get` is what you can actually access later in controllers and services.

42 Replace history path

Problem

You use location.path('/') and got stuck in a 'redirection loop'.

Solution

This is an easy one to fix. You just have to append `.replace()`.

```
$location.path('/').replace();
```

IV Filter Recipes

43 Filter an exact match (quick)

Problem

If you use the normal filter `| filter:search` with an input field, you also getting substrings of your search. But it's easy to do an exact matching.

Solution

The filter filter has a third argument which allows for exact matching

We have an array of objects like `[{ name: 'John', gender: 'male' }, { name: 'Anne', gender: 'female' }]`. If we would filter for `male` with the default filter filter we would get also all females.

When we set a third parameter to true we get exact matching, e.g. `... | filter:personFilter:true`.

Here's a complete example:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">

  <a href ng-click="personFilter.gender = undefined">Show all</a>
  <a href ng-click="personFilter.gender = 'male'">Show males</a>
  <a href ng-click="personFilter.gender = 'female'">Show females</a>
  <ul>
    <li ng-repeat="person in people | filter:personFilter:true">
      {{person.name}} ({{person.gender}})</li>
  </ul>

</body>
</html>
```

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope) {
    $scope.people = [
      { name: 'John', gender: 'male' },
      { name: 'Bill', gender: 'male' },
      { name: 'Anne', gender: 'female' }
    ];
  });
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/filters-exact-match

Online demo: http://sbrink.github.io/angularjs-cookbook-code/filters-exact-match/

44 Get last element(s) in a collection (quick)

Problem

You have a collection and only want to display the last element(s).

Solution

You can use the `limitTo` filter for this. The filter allows negative values which limit the elements starting from the last one. `limitTo:-1` would only output the last element.

A working example:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <ul>
    <li ng-repeat="item in items | limitTo:-2" ng-bind="item.name"></li>
  </ul>
</body>
</html>
```

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope) {
    $scope.items = [{ id: 1, name: 'First'  },
                    { id: 2, name: 'Second' },
                    { id: 3, name: 'Third'  },
                    { id: 4, name: 'Forth'  }];
  });
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/filters-get-last-element

Online demo: http://sbrink.github.io/angularjs-cookbook-code/filters-get-last-element/

45 How to highlight a search

Problem

You want to highlight a search string within your ng-repeat.

Solution

For this to work, we have to first create a filter. This filter wraps all occurrences of the search string with a span with a class `.highlight`. This is done with a replace with a regular expression. `RegExp('('+ search + ')', 'gi');` has the second parameters g and i which mean 'all occurrences' and don't care for the case. You need the parentheses because the part inside will be contained in $1 in the replace statement. See (TODO) Regular expressions for more information.

The next thing is the module `ngSanitze`. The default behaviour of AngularJS is to replace all < and > with `< and > in an expression. Because we want to output the raw html, we include` ngSanitze `and get a new directive with it` ng-bind-html'. This directive allows html but tries to sanitize it to prevent XSS attacks.

Tip: You're using Webstorm? There is a Regex tester plugin here (TODO link)

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope) {
    $scope.people = [
      { id:1, name:'John' },
      { id:2, name:'Bill' },
      { id:3, name:'Phil' }
    ];

    $scope.selected = [1,2];
  });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <select multiple ng-model="selected"
          ng-options="person.id as person.name for person in people">
  </select>
  {{ selected }}
</body>
</html>
```

For the styling we add a `.highlight` class which highlight the part in red.

<<(code/directives-select-multiple/style.css)

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/filters-search-highlight

Online demo: http://sbrink.github.io/angularjs-cookbook-code/filters-search-highlight/

46 Easy filtering with the filter filter

Problem

In your view you need a simple filter which only concerns your current view/controller.

Solution

This one confuses a lot of people. AngularJS has a concept called filters. With that you can transform a string or a collection by using a pipe `|`. But one of the default filters which AngularJS brings with it, is also called filter. This filter filter is what we use here.

The filter filter is a general purpose filter which can take string, options and functions as parameters. It has an optional third argument which does exact matches for strings and objects.

In the following examples we use this list (in JSON notation) as example:

```
[
  { "name": "Bill", "age": 42, "gender": "male" },
  { "name": "John", "age": 52, "gender": "male" },
  { "name": "Anne", "age": 19, "gender": "female" },
  { "name": "Phil", "age": 21, "gender": "male" },
  { "name": "Mary", "age": 23, "gender": "female" }
]
```

String

If you use a string as input field, you're matching all object containing this string. If you take the list of uses you can type `il` and and match B*il*l and Ph*il*. If you want exact matches you would use `true` as third argument.

```
<tr ng-repeat="user in users | filter:withString">
```

Object

If you don't want to match every attribute in an object, you can use object notation and specify a subset of fields. Here we only filter for the gender *male*. We use the second parameter to do an exact match. Otherwise we would get all rows with fe*male*, too.

```
<tr ng-repeat="user in users | filter:{ gender: 'male' }:true">
```

There is one special in object notation - the $ attribute. This is an wilcard like the normal string.

```
<tr ng-repeat="user in users | filter:{ $: withWildcard, gender: 'female' }">
```

Function

If you need a more complex filter which is only used in this controller and you don't want to create a filter on its own, you can also pass a function.

The function is evaluated for each element in the collection. So in our example you would test each single user for a given condition. Here we filter all users which are under 40.

```
$scope.underForty = function(user) {
    return user.age < 40;
};

<tr ng-repeat="user in users | filter:underForty">
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/filters-the-filter-filter

Online demo: http://sbrink.github.io/angularjs-cookbook-code/filters-the-filter-filter/

V Promise Recipes

47 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/

48 Convert a 3rd party promise with $q.when

Problem

You have a 3rd party promise and need to convert it into and $q promise

Solution

Why would you need to convert a 3rd party promise if the api is compatible? Because AngularJS needs a hint when to do an $apply and start the dirty checking cycle. See more about dirty checking in [How to use the scope right] (#big-picture-use-the-scope-right).

Promise is api compatible

If the promise you want to convert is compatible to the promise API `.then(successCallback, errCallback)` it's really easy. You just have to use `$q.when(foreignPromise)` and AngularJS converts it for you.

Look at the following example where we convert a jQuery promise:

```
$q.when($.ajax({url: 'users.json'})).then(function(users) {
    $scope.users = users;
});
```

See the full in the code section.

Convert an incompatible api

If you have a function with callbacks like success or error, the api is not compatible. But you can also use `$q .when` in this situation. `$q.when` can take three optional arguments like:

```
function when(value, callback, errback, progressback)
```

So if you have a function like `ajaxLib` with has callback functions like `.success`, `.failure` and `.progress`, you can convert it like so:

$q.when(ajaxLib, ajaxLib.success, ajaxLib.failure, ajaxLib.progress)

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/promises-convert-3rd-party

Online demo: http://sbrink.github.io/angularjs-cookbook-code/promises-convert-3rd-party/

49 TODO: How to wait for several async events

Problem

You have several async events, e.g. calls to serveral external apis. You cannot work if one failed, so you have to wait for them all. Here we show an easy way to do it.

Solution

With callbacks, it's hard to start several asynchronous tasks at once and synchronize the results later. With promises, it is really easy.

In this example, we use two promises called `firstPromise` and `secondPromise`. The first promise we create with a defer and have use two functions on the scope to either resolve or reject it. The second promise we create with the `$q.when`. This function creates a promise which is resolved immediately with the string we pass to it ('Resolved another promise').

After that, we can use `$q.all` to call several promises in parallel. To do that, we pass an array of promises to `$q.all`. The result itself is again a promise. The success function contains an array with the result of all promises. The error callback is only the error result of the promise that failed first.

Again in short:

- **success**: returns an array with the result of every promise
- **error**: returns the result of the first failing promise

```
angular.module('cookbookApp', [])
  .controller('MainController', function($scope, $q) {
    var defer = $q.defer();
    var firstPromise = defer.promise;
    var secondPromise = $q.when('Resolved second promise');

    $scope.resolve = function() {
      defer.resolve('Resolved first promise');
    };
    $scope.reject = function() {
```

```
      defer.reject('Error in first promise');
    };

    $q.all([firstPromise, secondPromise]).then(function(messages) {
      $scope.resultAll =  messages.join(' and ');
    }, function(reason) {
      $scope.resultAll = reason;
    });
  });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <fieldset>
    <legend>Promise</legend>
    <button ng-click="resolve()">Resolve</button>
    <button ng-click="reject()">Reject</button>{{resultOne}}
  </fieldset>

  <fieldset>
    <legend>Result all</legend> {{resultAll}}
  </fieldset>
</body>
</html>
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/promises-start-events-in-parallel

Online demo: http://sbrink.github.io/angularjs-cookbook-code/promises-start-events-in-parallel/

50 How to transform every callback into a promise

Problem

You have a 3rd party library which uses callbacks. You have some async mechanism in your app and need to wait for the result of several events. Maybe an $http promise and the result of external api which uses normal callbacks. Now you want to chain them like in [How to wait for several async events] or want to have an unified api.

Solution

As an example, we use the camera feature of phonegap. The original definition looks like this:

```
navigator.camera.getPicture(cameraSuccess, cameraError, [ cameraOptions ]);
```

As an promise we want sth. like this:

```
phonegapCamera.getPicture([ cameraOptions ]).then(success, failure);
```

The necessary steps are:

1. Make sure that you injected `$q`
2. Create a deferred object with `$q.defer()`
3. Define the library function with callbacks
4. Use `deferred.resolve(data)` in the success function
5. Use `deferred.reject(error)` in the error function
6. Return `deferred.promise`

Result:

```
function getPicture(options) {
    var deferred = $q.defer()

    navigator.camera.getPicture(onSuccess, onFail, options);

    function onSuccess(imageData) {
        deferred.resolve(imageData);
    }

    function onFail(message) {
        deferred.reject(message);
    }

    return deferred.promise;
}
```

Because we can pass functions as arguments and callback and resolve/reject, take both exactly one parameter. We can also write a much simpler version (complete example):

```
app.factory('phonegapCamera', function($q) {
    function getPicture(options) {
        var deferred = $q.defer()
        navigator.camera.getPicture(deferred.resolve, deferred.reject, options);
        return deferred.promise;
    }

    return {
        getPicture: getPicture
    }
}
```

Unfamilar with the declaration used in the factory? Learn more here:

VI Testing Recipes

51 Testing focus directive

Problem

You want to test focus on an element but don't know how.

Solution

The trick here is to append the element you want to test with the body. You can only test focus on elements which are bound to the DOM.

In this example, we take simple directive which just gets the focus. We create a new scope and create a template with our focus directive. After we compiled it, we attach it to the body. Checks on focus now work.

```
it('should focus the input field', inject(function ($rootScope, $compile) {
    var scope = $rootScope.$new();
    var template = '<input type="text" focus-me />';
    var element = $compile(template)(scope);
    element.appendTo(document.body);
    scope.$apply();

    expect(element.find('input').is(':focus')).toBe(false);
    element.find('input').focus();
    expect(element.find('input').is(':focus')).toBe(true);
}));
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/testing-focus-in-directive

Online demo: http://sbrink.github.io/angularjs-cookbook-code/testing-focus-in-directive/

52 Mocking http requests

Problem

You want to test your service which itself uses $http. You want to isolate your tests a mock the returned data.

Solution

We take the following factory as an example:

```
angular.module('mockHttpApp', [])
  .factory('Task', function($http) {
    return {
      all: function() {
        return $http.get('/tasks').then(function(tasksResponse){
          return tasksResponse.data;
        });
      }
    };
  });
```

```
.factory('Task', function($http) {
  return {
    all: function() {
      return $http.get('/tasks').then(function(tasksResponse){
        return tasksResponse.data;
      });
    }
  };
});
```

The Task factory should just return a list of the task and return the data directly instead of an response object. What's problematic here, that if we test it, we want to isolate it without a backend. And even if we could isolate it, how do we manage to resolve the promise, returned be the service.

The way to do it, is to use the `$httpBackend` service. This is a fake backend service, whom we can tell which data it should return. It is also capable of resolving the promises.

The way to use it, you see in the following code:

```
describe('Task Factory', function () {
  beforeEach(angular.mock.module('mockHttpApp'));

  beforeEach(inject(function (_$httpBackend_, _Task_) {
    $httpBackend = _$httpBackend_;
    Task = _Task_;
  }));

  it('should just return the task array without response object', function () {
    var result,
      sample = ['Tidy up', 'Clean the dishes'];

    $httpBackend.when('GET', '/tasks', {}).respond(sample);

    Task.all().then(function(response){
      result = response;
    });

    $httpBackend.flush();

    expect(result).toEqual(sample);
  });

});
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/testing-mock-http-requests

Online demo: http://sbrink.github.io/angularjs-cookbook-code/testing-mock-http-requests/

53 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.

VII Big Picture Recipes

54 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');
  };
});
```

55 Spreading route definitions among modules

Problem

You have several modules and want each module to have its own route definitions.

Solution

AngularJS allows this out of the box. Every module can have its own config function with an injected `$routeProvider` service. The final result is a merged version of all routes. If you define the same route twice, the latter is taken.

```
angular.module('cookbookRecipes', []).config(function ($routeProvider) {
  $routeProvider.when('/recipes', { ... });
  $routeProvider.when('/recipes/new', { ... });
  $routeProvider.when('/recipes/:recipeId', { ... });
});

angular.module('cookbookIngredients', []).config(function ($routeProvider) {
  $routeProvider.when('/ingredients', { ... });
  $routeProvider.when('/ingredients/new', { ... });
  $routeProvider.when('/ingredients/:recipeId', { ... });
});

angular.module('cookbookApp', ['cookbookRecipes', 'cookbookIngredients']);
```

56 Stop timers before a scope is removed

Problem

You have a timer which is still running after switching the url. You want to stop it up before the scope is removed.

Solution

AngularJS has its own eventing mechanism. Before a scope is removed from its parent, AngularJS will send an event called `$destroy`. This happens before tearing down the scope.

To give an example, we look at the following controller. The controller continuously updates a date in one second intervals. This timer would also continue to run after we removed the controller. With every instance of a new controller we would create a new timer which will then run forever. Thus we have to remove it manually.

```
.controller('DateController', function($scope, $interval) {
  function refreshDate() { $scope.now = new Date(); };
  $interval(refreshDate, 1000);
});
```

To solve this, we have to listen for the `$destroy` event and stop the timer. To do this, we need to hold a reference to timer (`dateTimer`). That way, we can cancel the timer on the `$destroy` event like shown here:

```
.controller('DateController', function($scope, $timeout) {
  var dateTimer;

  function refreshDate() { $scope.now = new Date(); };
  dateTimer = $interval(refreshDate, 1000);

  $scope.$on('$destroy', function() {
    if (dateTimer) { $interval.cancel(dateTimer); }
  });
});
```

57 What all the extra .js files are doing?

Problem

AngularJS consists not only of a single file *angular.js* but several files. You want to know which are there and what do they do?

Solution

Considering you downloaded the AngularJS zip file from http://angularjs.org, here are the files and folders with their function.

i18n folder

The `i18n` Contains localization files for date / time / number formatting. You have to include the language file in your index file. The files are named like `angular-locale_de-de.js`.

docs folder

The whole documentation as offline version. To get this to work, you have to start an http server in the root and navigate your browser to `/docs`. Remember that AngularJS does a location push. If you navigate through the documentation, you can't do a browser reload. You always have to start a `/docs` again or set up special rewrite rules for your http server.

angular-animate.js

Include with `angular.module('myApp', ['ngAnimate']).`

`ngAnimate` is an optional module that provides CSS and JavaScript animation hooks.

angular-cookies.js

Include with `angular.module('myApp', ['ngCookies']).`

Services included in this module:

- `$cookies`: This is a wrapper around browser cookies
- `$cookieStore`: Objects you put or retrieve from the cookie store are automatically (de)serialized.

angular-loader.js

A module loader for AngularJS modules.

If you are loading multiple script files containing AngularJS modules, you can load them asynchronously and in any order. You only have to make sure that you load this file first.

It's a good idea to put the contents of this file is into your `index.html` to save the initial request.

angular-mocks.js (Testing)

This file contains an implementation of mocks that makes is easier to test your app. It includes the `$httpBackend`, which you need to for proper testing the `$http` service. Additionally it contains overwrites for existing services like `$exceptionHandler`, `$interval`, `$log` and `$timeout`.

angular-resource.js

The `ngResource` module contains just one service: `$resource`. This service is build on top of `$http` and can abstract a rest api.

angular-route.js

The route provider has now it's own file / module. Usually you want to include it in every application.

angular-sanitize.js

The `ngSanitize` module includes:

- `ng-bind-html` directive, which allows you to output sanitized html
- linky filter which turns text links into hyperlinks
- $sanitize service which is whitelist filter for html

angular-scenario.js (Testing)

This file helps you writing and executing end-to-end tests for angular applications.

angular-touch.js

Handles touch events in AngularJS. Implements Fast click which eliminats the 300ms delay between a tap and the actual firing of a click event on mobile browsers.

angular.js

The core of AngularJS. You always have to include this.

58 How to debug your application with the browser

Problem

You're building your application and something you defined isn't showing up in the view. You then try to find the cause of the problem.

Solution

Batarang

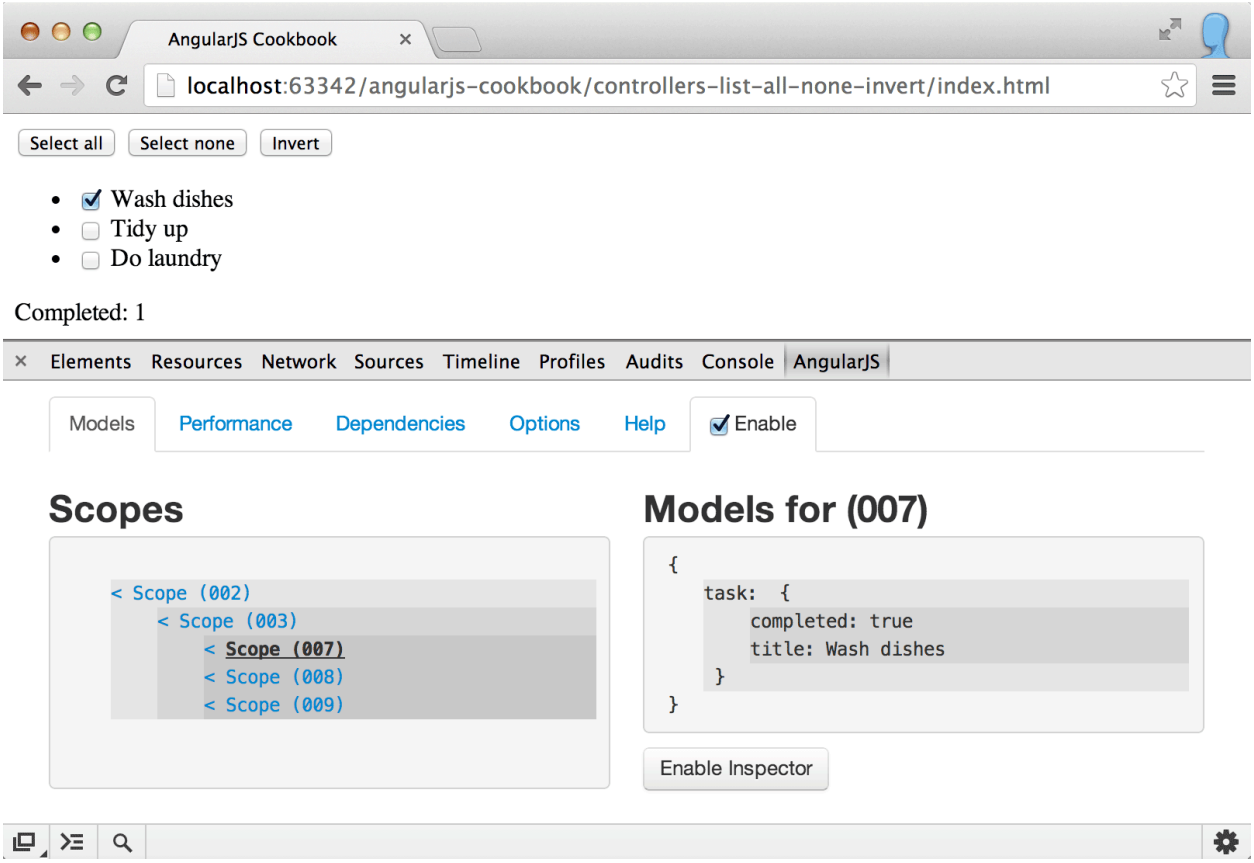
A method for smaller applications is to use Batarang. Batarang is a chrome extension which was specifically designed for AngularJS developers.

To use it, you have to start Goole Chrome[1] as a browser and install the Batarang extension[2] from the chrome store.

If you now open the chrome developer tools, you get a new tab called *AngularJS* (see screenshot). After you checked *Enable*, you can can use the *Models* tab to inspect you scopes.

[1] http://www.google.com/chrome/â€Ž

[2] https://chrome.google.com/webstore/detail/angularjs-batarang/ighdmehidhipcmcojjgiloacoafjmpfk



Batarang Scopes

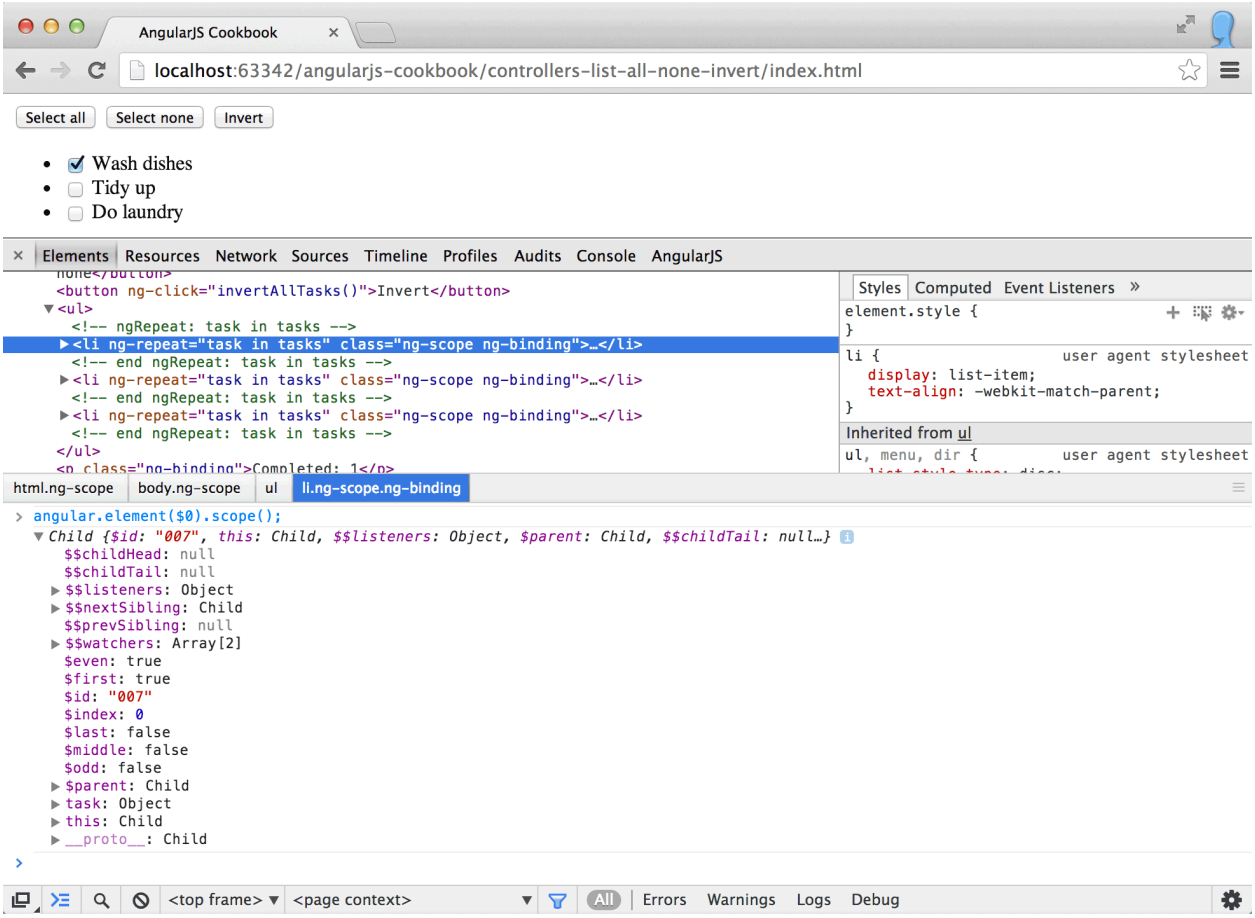
In this example, we see the list recipe. What you see:

- Scope (002): the root scope
- Scope (003): the scope of ng-repeat
- Scope (007-008): the individual list elements

You can now click on each scope and see its contents.

Chrome inspector

If your application is bigger and you have problems with batarang or you need more detailed information, there's a second way. `angular.element` has a `scope()` function which returns all available information about a scope. This includes watchers and internal variables like `$index`. To make the selection of scopes really easy, we combine `angular.element` with a nice trick from the chrome developer tools. With the chrome developer, you get the last selected element with `$0`. So we just have to inspect an element and write `angular.element($0).scope();` into the console. This returns all information about the scope.



Inspect

Here we inspected the *Wash dishes* list element. We then pressed Escape which opened the chrome console and entered `angular.element($0).scope();`. As you can see, there are all the internal variables created by `ng-repeat` like `$index`, `$first`, `$last` etc.

59 EcmaScript 5 array functions you should know and use

Problem

You're building complex loops where you don't need it or you're including other libraries to ease manipulating arrays but don't know about EcmaScript 5.

Solution

Until EcmaScript 5, working with arrays in JavaScript was no fun and usually ended in including underscore or lodash to ease the pain of working with arrays.

If you have to support older browsers, you can use a polyfill like array-generics[1] which emulates the functions if not available. See Compatibility Matrix[2] for brower support.

forEach

Reference[3]

```
[1, 5, 9].forEach(function(element, index, array) {
    console.log("a[" + index + "] = " + element);
}));
// => a[0] = 1
// => a[1] = 5
// => a[2] = 9
```

This is similar to angular.forEach(). So which should you use? At the moment you should use `angular.forEach()`. If you look at this comparison[4], you see that the ES5 forEach implementation is not the fastest. The AngularJS version in this comparison uses ES5 forEach if it's available. This is changed by this commit[5]. Now it's always using the fastest for loop.

[1] https://github.com/plusdude/array-generics
[2] http://kangax.github.io/es5-compat-table/
[3] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach
[4] http://jsperf.com/foreach-vs-loop/20
[5] https://github.com/angular/angular.js/issues/3221

every

Reference[6]

Checks if every element meets a certain condition in the array.

```
[1, 2, 3, 4, 5].every(function(element, index, array){
    return element < 4;
});
// => false
```

Returns true or false.

some

Reference[7]

Checks if at least one element meets a condition.

```
[1, 2, 3, 4, 5].some(function(element, index, array){
    return element >= 3;
});
// => true
```

Returns true or false.

filter

Reference[8]

Creates a new array with only the elements that meet the condition.

```
[1, 2, 3, 4, 5].filter(function(element, index, array){
    return element % 2 === 0;
});
// => [2, 4]
```

Returns an array (can be smaller than the original array).

map

Reference[9]

Creates a new array where every element is transformed by the function.

[6] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/every
[7] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/some
[8] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter
[9] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

```
[1, 2, 3].map(function(element, index, array){
    return element * element;
});
// => [1, 4, 9]
```

Returns an array (is of the same length as the original array).

indexOf

Reference[10]

Returns the index of the element.

The following example finds all occurrences of an element in the array:

```
var indices = [];
var i = array.indexOf(element);
while (i != -1) {
    indices.push(i);
    i = array.indexOf(element, i + 1);
}
```

It also works with simple objects which do not contain functions.

```
var a = { a: 1 }, b = { b: 2 };
[a, b].indexOf(b)
// => 1
```

Returns the index of the element or -1 if not found.

reduce

Reference[11]

Walks through an array and applies the function to an one element called accumulator.

[10] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/indexOf
[11] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce

```
[0,1,2,3,4].reduce(function(previousValue, currentValue, index, array){
  return previousValue + currentValue;
});
// => 10
```

Returns the value, which is saved in previousValue.

Here is a more useful example to convert an array of arrays to an object:

a = [['a', 1], ['b', 2]] a.reduce(function(map, arrayElement) { map[arrayElement[0]] = arrayElement[1]; return map; }, {}) // ⇒ { a: 1, b: 2 }

60 Execute code at startup

Problem

You want to execute some code when AngularJS start. It should run before any controllers and directives and should not be tied to any view.

Solution

You can use the `run` method. This method is invoked when the injector loaded all modules.

```
angular.module('myApp',[])
    .run(function()
        // Your code here
    });
```

61 Finding Bottlenecks with Batarang

Problem

You have a lagging application and need a way to investigate which part causes your performance issues.

Solution

You can do some benchmarking with Batarang[1].

Batarang is a chrome extension which was specifically designed for AngularJS developers.

With batarang you get a list of all watchers and their relative time spent. To show you how it works, we will sort a list with bogosort. Bogosort is a very inefficient sorting algorithm which randomizes the order of the elements and checks if the array is now sorted. If not, it randomizes the elements again.

The demo application implements bogosort as a filter:

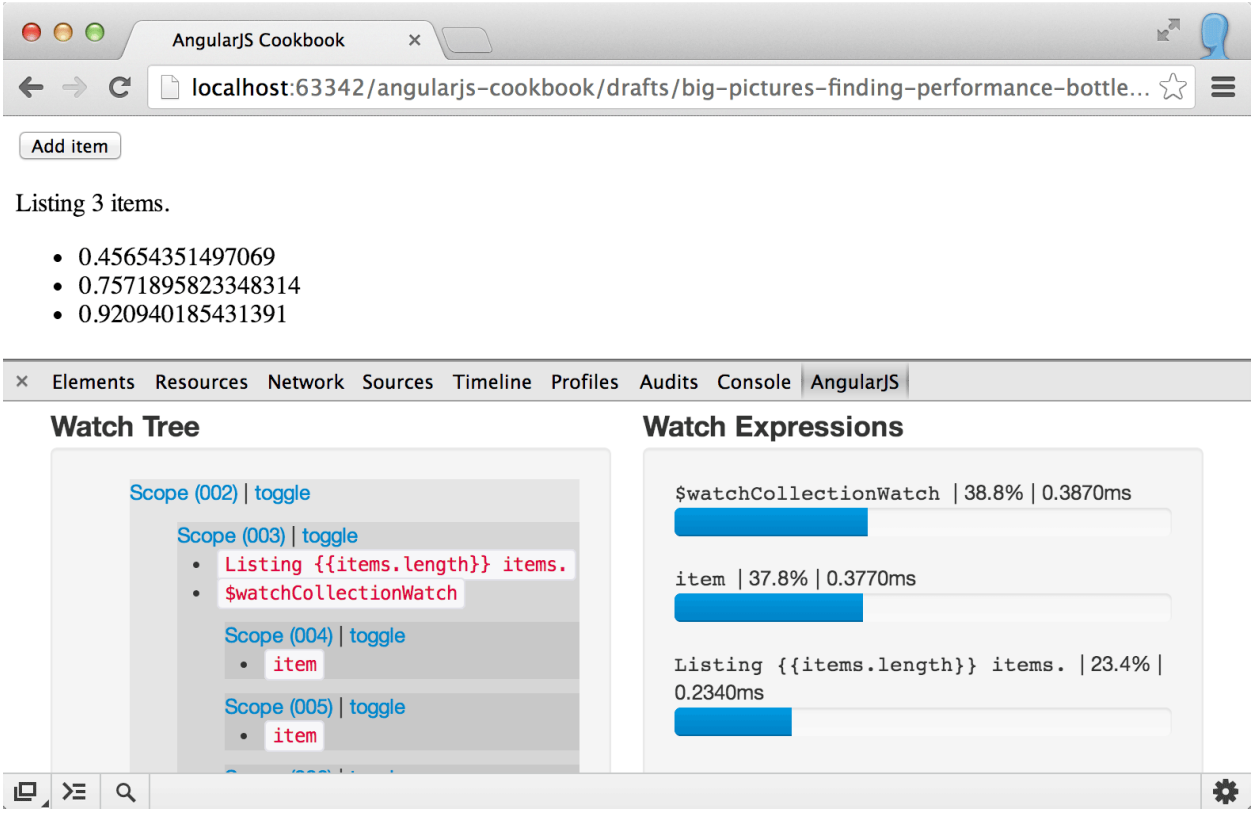
```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <button ng-click="addItem()">Add item</button>
0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567
  <p>Listing {{items.length}} items.</p>

  <ul>
    <li ng-repeat="item in items | bogosort" ng-bind="item"></li>
  </ul>
</body>
</html>
```

[1] https://chrome.google.com/webstore/detail/angularjs-batarang/ighdmehidhipcmcojjgiloacoafjmpfk

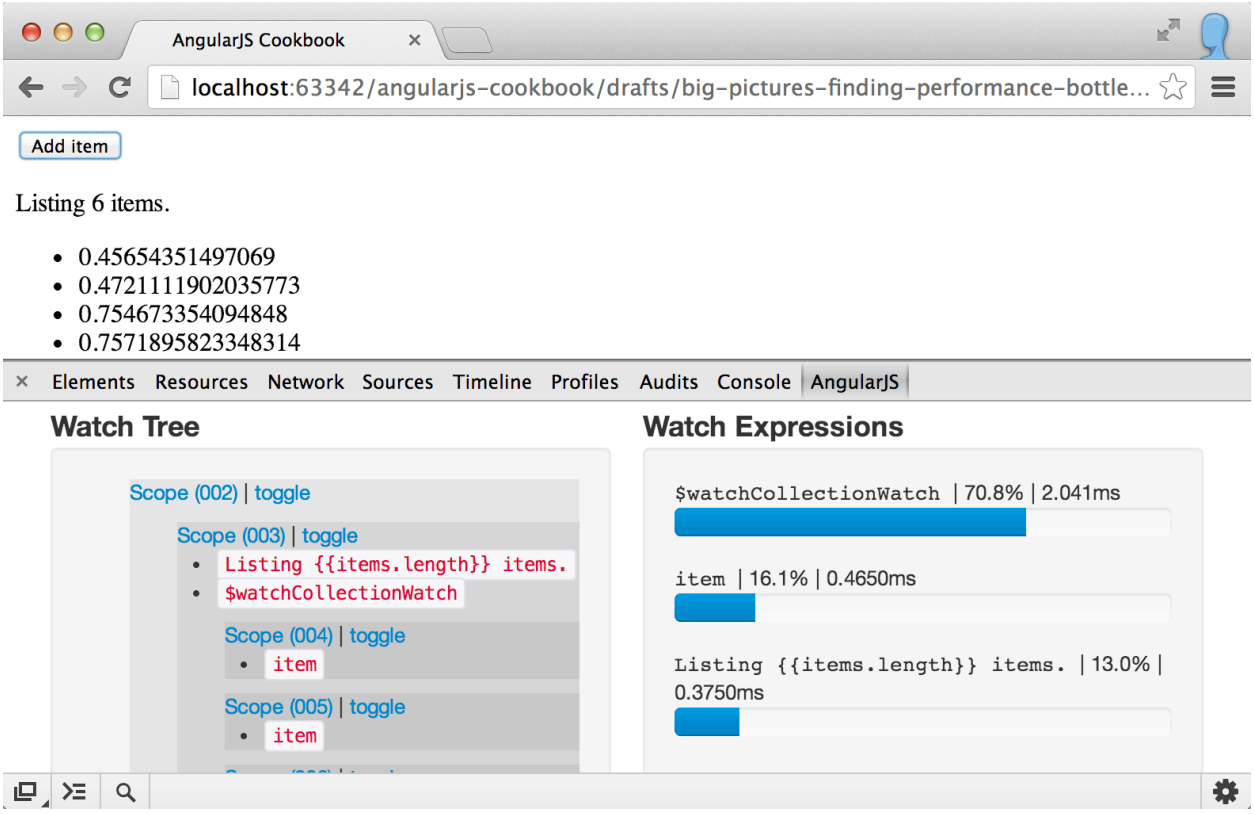
```
angular.module('cookbookApp', [])
  .filter('bogosort', function(){
    function shuffle(v) {
      // http://jsfromhell.com/array/shuffle
      for(var j, x,
            i = v.length; i;
          j = parseInt(Math.random() * i, 10), x = v[--i], v[i] = v[j], v[j] = x
        );
      return v;
    }
    function isSorted(v){
      for(var i=1; i<v.length; i++) {
        if (v[i-1] > v[i]) { return false; }
      }
      return true;
    }
    return function(input) {
      var sorted = false;
      while(sorted === false){
        input = shuffle(input);
        sorted = isSorted(input);
      }
      return input;
    };
  })
  .controller('MainController', function($scope) {
    $scope.items = [Math.random(), Math.random(), Math.random()];

    $scope.addItem = function() {
      $scope.items.push(Math.random());
    };
  });
```

Now to measure the performance, you open the performance tab in Batarang and you use the features of your application, which are slow. Batarang now sums the time up for each watcher used.

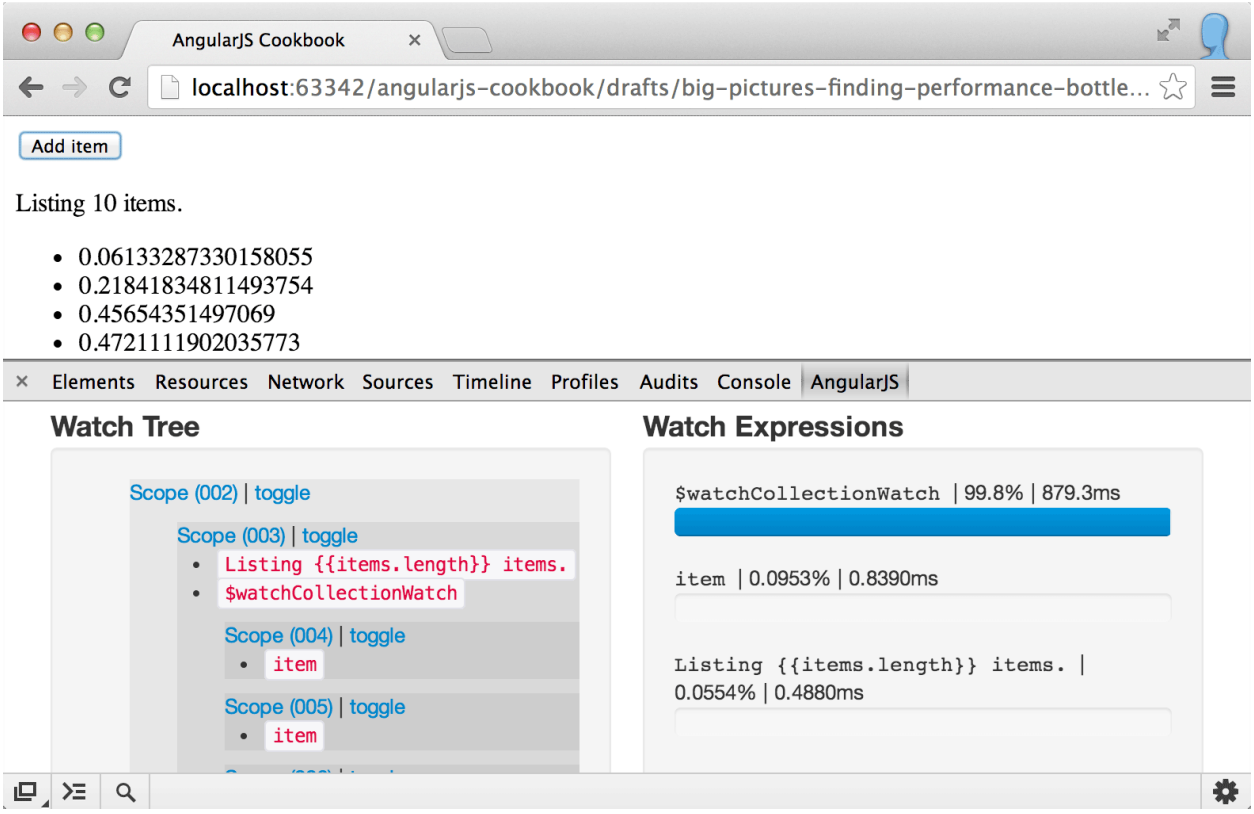
In the following screenshot, you the result after we added some items to the array.



Bogosort with 3 items



Bogosort with 6 items



Bogosort with 10 items

In the screenshot, you see how the time for the collection increases. With 10 items, it's 99.8% for the collection watcher. So here's definitely the bottleneck. Of course, most of the time, it's not that obvious, but it gives you a clue of where to start.

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/big-picture-finding-performance-bottlenecks

Online demo: http://sbrink.github.io/angularjs-cookbook-code/big-picture-finding-performance-bottlenecks/

62 How to use regular urls without the hash

Problem

By default, AngularJS uses the `#` for urls to write urls like `#/posts/1`. You want to have nice urls and drop the hash.

Solution

The first part is to change the default behavior of AngularJS and remove the `#`. To do this, you have to inject the `$locationProvider` into the config block of your application and set `$locationProvider.html5Mode(true);`.

```
angular.module('cookbookApp', [])
    .config(function($locationProvider) {
        $locationProvider.html5Mode(true);
    }
)
```

The second part concerns your webserver. If you use the urls with the hash, it always the your index page which is opened. If we look at `http://example.com/#/posts/1`, usually the http://example.com/index.html is used. #/posts/1 is just an anchor tag. So, the webserver only needs to place the index.html at the root path and everything is fine. If we now switch to *html5Mode*, the url would look like this: `http://example.com/posts/1`. Now the webserver assumes that there is a folder with a file like http://example.com/posts/1/index.html. Of course, we don't want to place an index.html for every new page create. And of course, we can't predict a lot of urls. The solution is to define rewrite rules for your webserver. So every url is automatically rewritten to `http://example.com/`. To do this, you need a specific configuration for your webserver. Here we give examples for apache and nginx.

Nginx

Here we rewrite every url except urls starting with /images.

```
rewrite ^/(?!images) / last;
```

Apache

For apache it's a little bit longer

```
<ifModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_URI} !index
    RewriteCond %{REQUEST_URI} !.*\.(css ¦js|html|png)
    RewriteRule (.*) index.html [L]
</ifModule>
```

63 Report backend errors

Problem

You want to catch your backend errors and display them directly in the frontend to help you as a developer.

Solution

The solution consists of two parts. A directive and an http interceptor.

We define the the http interceptor as a service and transform the error into an event and broadcast it with `$rootScope.$broadcast('responseError', responseError);`.

The directive `errorOutput` just waits for the error events and appends a new one at the end.

```
angular.module('cookbookApp', [])
  .config(function($httpProvider) {
    $httpProvider.interceptors.push('httpErrorInterceptor');
  })
  .factory('httpErrorInterceptor', function ($q, $rootScope) {
    return {
      'responseError': function(responseError) {
        $rootScope.$broadcast('responseError', responseError);
        return responseError;
      }
    };
  })
  .directive('errorOutput', function() {
    return {
      restrict: 'E',
      link: function(scope, element, attrs) {
        scope.$on('responseError', function(event, response) {
          var status  = response.status;
          var url     = response.config.url;
          var headers = JSON.stringify(response.headers());
          element.append(status + ' ' + url + ' ' + headers + '<br>');
        });
```

```
      }
    };
  })
  .controller('MainController', function($scope, $http) {
    $http.get('fail1.json');
    $http.get('fail2.json');
  });
```

Here is the really simple view which has just the element for displaying data:

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body ng-controller="MainController">
  <h3>Error messages:</h3>
  <error-output></error-output>
</body>
</html>
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/big-picture-report-backend-errors

Online demo: http://sbrink.github.io/angularjs-cookbook-code/big-picture-report-backend-errors/

64 Optional params and wildcards in Router

Problem

You want to use optional params in an url for language setting or match a part of the url including slashes.

Solution

The router has options for optional params. A part of the url is called group.

Here we have a little example which demonstrates both possibilities:

Don't forget to include `ngRoute`!

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="../vendor/angular-route.js"></script>
  <script src="application.js"></script>
</head>
<body>
  <div ng-view></div>
  <p><a href="#/pages/products/topseller">Demo wildcard</a></p>
  <p>
    <a href="#/admin">With optional group</a>
    <a href="#/en/admin">Witout Optional group</a>
  </p>
</body>
</html>
```

```
angular.module('cookbookApp', ['ngRoute'])
  .config(function($routeProvider) {
    $routeProvider
      .when('/pages/:pages*', { templateUrl: 'demo.html' })
      .when('/:lang?/admin',  { templateUrl: 'demo.html' });
  })
  .controller('RouteController', function($scope, $location, $routeParams) {
    $scope.locationPath = $location.path();
    $scope.routeParams = $routeParams;
  });
```

```
<div ng-controller="RouteController">
  <p>Path: {{locationPath}}</p>
  <p>Params: {{routeParams}}</p>
</div>
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/big-picture-router-optionals-and-wildcards

Online demo: http://sbrink.github.io/angularjs-cookbook-code/big-picture-router-optionals-and-wildcards/

65 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();
```

66 How to use the dot correctly

Problem

You're sharing variables in the scope hierarchy and sometimes they don't update or behave like you expect.

Solution

Oftentimes you have the following situation (try it here[1]):

```
<div ng-controller="ParentController">
    <input type="text" ng-model="name">

    <div ng-controller="ChildController">
        <input type="text" ng-model="name">
    </div>
</div>
```

If you change `name` in the `ParentController`, the change is reflected in the `ChildController`. This is the expected behaviour because scopes use prototypal inheritance. This is expected because at this point there is not `name` attribute on the `ChildController`, so it's looked up it the prototype chain.

The unexpected behavior comes when you now edit the `name` of `ChildController`. If you try it in the example, you'll see that both are now out of sync. So there are now two independent `name` variables on each scope.

If you use `person.name` instead of `name`, you get a differenct result. If you again first change `name` of the `ParentController` and then `name` of the `ChildController`, they stay in sync.

** Why? **

This is not angular's fault. It is the way prototypal inheritance in JavaScript works. We differentiate here between reading and writing variables.

- reading: Reading does everything as you expect. If a variable on the current object isn't found, it goes through the prototypal chain and tries to find it on another object.

[1] http://sbrink.github.io/angularjs-cookbook-code/big-picture-use-the-dot-correctly/index.html

- writing: This is where the problem arises. If you write a simple property like a string or number, the prototype chain is never consulted. Only where you have an array, object or function it is. So where you use the latter, JavaScript goes the prototype chain up, looks for an occurence and writes the value there.

** How to solve this:**

- use a dot in your variable names (*preferred*)
- use `$parent.myVariableName` in the child (*workaround*)

** Here's a full demo: **

```
angular.module('cookbookApp', [])
  .controller('ParentController', function() { })
  .controller('ChildController', function() { });
```

```
<html ng-app="cookbookApp">
<head>
  <script src="../vendor/angular.js"></script>
  <script src="application.js"></script>
</head>
<body>
  <h2>Without dot</h2>
  <div ng-controller="ParentController">
    Parent: <input type="text" ng-model="name" placeholder="name">

    <div ng-controller="ChildController">
      Child: <input type="text" ng-model="name" placeholder="name">
    </div>
  </div>

  <h2>With dot</h2>
  <div ng-controller="ParentController">
    Parent: <input type="text"
                   ng-model="person.name" placeholder="person.name">

    <div ng-controller="ChildController">
      Child: <input type="text"
                    ng-model="person.name" placeholder="person.name">
    </div>
```

```
  </div>

  <h2>Without dot (workaround)</h2>
  <div ng-controller="ParentController">
    Parent: <input type="text" ng-model="name" placeholder="name">

    <div ng-controller="ChildController">
      Child: <input type="text"
                    ng-model="$parent.name" placeholder="$parent.name">
    </div>
  </div>
</body>
</html>
```

Code

Complete source: https://github.com/sbrink/angularjs-cookbook-code/tree/gh-pages/big-picture-use-the-dot-correctly

Online demo: http://sbrink.github.io/angularjs-cookbook-code/big-picture-use-the-dot-correctly/

67 What belongs on the scope

Problem

Your app gets slower and slower or you're wondering when to place variables and functions on the scope.

Solution

This one should be a no-brainer but it seems that it isn't.

Only variables and functions that you need in your view belong on the scope.

Discussion

Why is this important? Remember that the scope is a link between data structures and the view. For the two-way data binding to work, AngularJS has to find out when the model changed so that it can update the view.

AngularJS does this through dirty checking. If we make it simple, it means: Compare the current scope with an old version of the scope. So everything on the scope, whether it can be be updated in the view or not, is compared.

You may look at the following example where we only print one random user on the screen.

The view:

```
<body ng-controller="MainController">
   {{getRandomUser()}}
</body>
```

What you can often see is a controller like this:

```
.controller(function() {
    $scope.users = ['Bill', 'John', ... ]:

    $scope.getRandomUser = function() {
      return $scope.users[Math.floor(Math.random()*$scope.users.length)];
    };
}
```

What's wrong with this? In the view, only one random user is ever shown. But the dirty checking of AngularJS checks `$scope .users` every time because it is on the scope. It checks for a change of `$scope.users` so that it can update the view accordingly, so that the view doesn't use `users`. So the checking has no use. This can be a huge costs if it's a large array or complex object.

The right solution:

```
.controller(function() {
    var users = ['Bill', 'John', ... ];

    $scope.getRandomUser = function() {
      return users[Math.floor(Math.random()*users.length)];
    };
}
```

Here we store users as a normal variable. The dirty checking only checks for changes on `$scope.getRandomUser` which is right because it's visible in the view.