MAKE TODOS PERSISTENT WITH LOCAL STORAGE

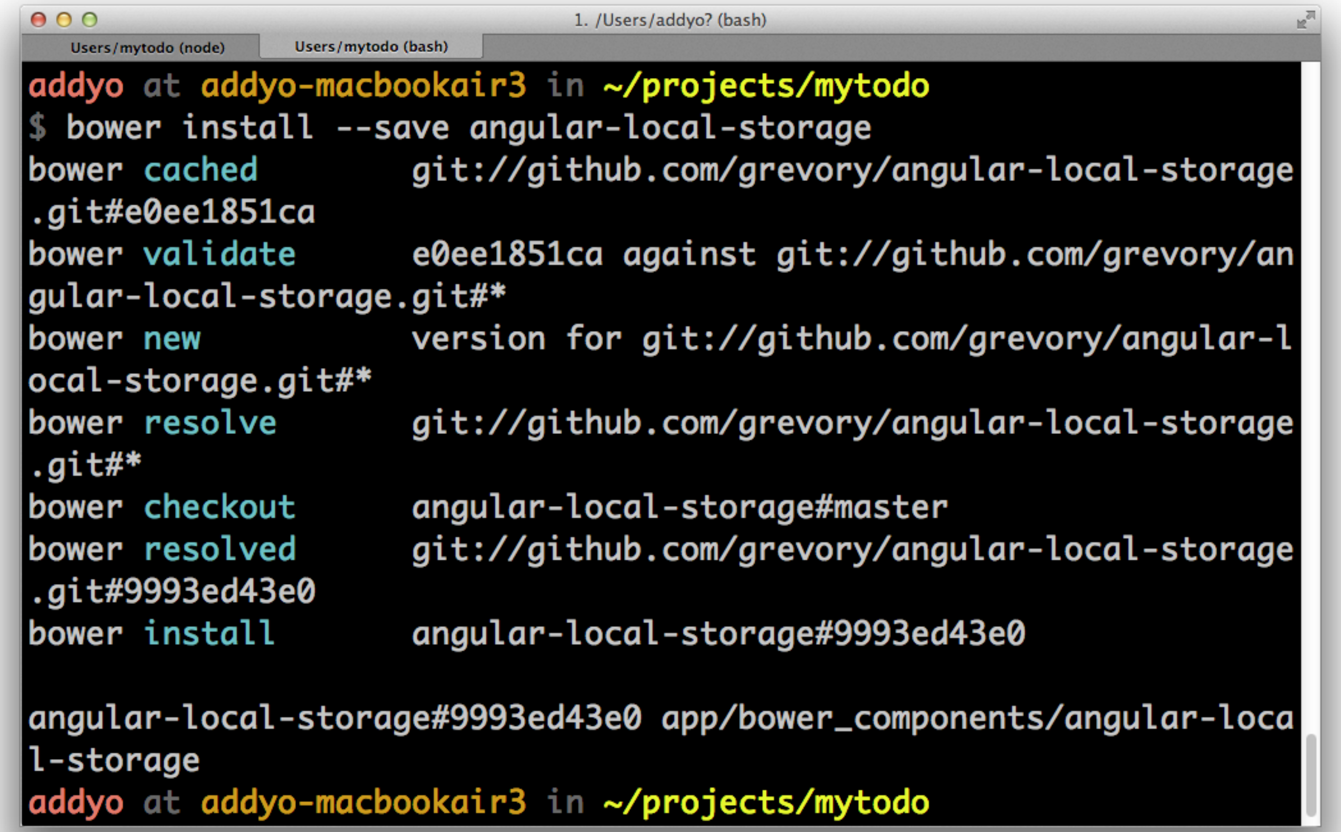
Let’s revisit the issue of items not persisting when the browser refreshes.

INSTALL BOWER PACKAGE

To easily achieve this we can use another Angular module called "angular-local-storage" that will allow us to quickly implement local storage. Again, Bower comes to the rescue.

Run the following command:

```
$ bower install --save angular-local-storage
```



```
addyo at addyo-macbookair3 in ~/projects/mytodo
$ bower install --save angular-local-storage
bower cached        git://github.com/grevory/angular-local-storage
.git#e0ee1851ca
bower validate      e0ee1851ca against git://github.com/grevory/an
gular-local-storage.git#*
bower new           version for git://github.com/grevory/angular-l
ocal-storage.git#*
bower resolve       git://github.com/grevory/angular-local-storage
.git#*
bower checkout      angular-local-storage#master
bower resolved      git://github.com/grevory/angular-local-storage
.git#9993ed43e0
bower install       angular-local-storage#9993ed43e0

angular-local-storage#9993ed43e0 app/bower_components/angular-loca
l-storage
addyo at addyo-macbookair3 in ~/projects/mytodo
```

ADD LOCAL STORAGE

Similar to how we added jQueryUI and AngularUI Sortable in Step 7 to make todos sortable, we need to add a reference to the *angular-local-storage.js* file in *index.html*.

Since we're using *bower.json* to keep track of our modules, Ctrl + C to exit the current command line process, then re-run `grunt serve` to get some automated magic in your *index.html*.

At the bottom of *index.html*, this should have been added:

```
<script src="bower_components/angular-local-storage/angular-local-storage.js">
</script>
```

Your *index.html* scripts block should now look like this:

```
<!-- build:js scripts/vendor.js -->
<!-- bower:js -->
<script src="bower_components/jquery/jquery.js"></script>
```

```
<script src="bower_components/angular/angular.js"></script>
<script src="bower_components/bootstrap/dist/js/bootstrap.js"></script>
<script src="bower_components/angular-resource/angular-resource.js"></script>
<script src="bower_components/angular-cookies/angular-cookies.js"></script>
<script src="bower_components/angular-sanitize/angular-sanitize.js"></script>
<script src="bower_components/angular-route/angular-route.js"></script>
<script src="bower_components/jquery-ui/ui/jquery-ui.js"></script>
<script src="bower_components/angular-ui-sortable/sortable.js"></script>
<script src="bower_components/angular-local-storage/angular-local-storage.js">
</script>
<!-- endbower -->
<!-- endbuild -->
```

Edit the `mytodoApp` application module (*scripts/app.js*) to include the `LocalStorageModule` adapter:

```
angular.module('mytodoApp', [
  'ngCookies',
  'ngResource',
  'ngSanitize',
  'ngRoute',
  'ui.sortable',
  'LocalStorageModule'
])
```

While you're in *app.js*, also configure `localStorageServiceProvider` to use `"ls"` as a localStorage name prefix so your app doesn't accidently read todos from another app using the same variable names:

```
.config(['localStorageServiceProvider', function(localStorageServiceProvider){
  localStorageServiceProvider.setPrefix('ls');
}])
```

Our application module should now look like this:

```
'use strict';

angular.module('mytodoApp', [
  'ngCookies',
  'ngResource',
  'ngSanitize',
  'ngRoute',
  'ui.sortable',
  'LocalStorageModule'
])
  .config(['localStorageServiceProvider', function(localStorageServiceProvider){
    localStorageServiceProvider.setPrefix('ls');
  }])
  .config(function ($routeProvider) {
    $routeProvider
      .when('/', {
        templateUrl: 'views/main.html',
        controller: 'MainCtrl'
      })
      .otherwise({
        redirectTo: '/'
      });
  });
```

You will also need to update your controller (*main.js*) to declare a dependency on the localStorage service. Add `localStorageService` as the second parameter in the callback function.

```
'use strict';

angular.module('mytodoApp')
  .controller('MainCtrl', function ($scope, localStorageService) {
    // (code hidden here to save space)
  });
```

So now, rather than reading our todos from a static array, we'll be reading it from local storage and then storing it in `$scope.todos` instead.

We'll also use the angular `$watch` listener to watch for changes in the value of `$scope.todos`. If someone adds or removes a todo, it will then keep our local storage `todos` datastore in sync.

Therefore, we need to remove the current `$scope.todos` declaration:

```
$scope.todos = ['Item 1', 'Item 2', 'Item 3'];
```

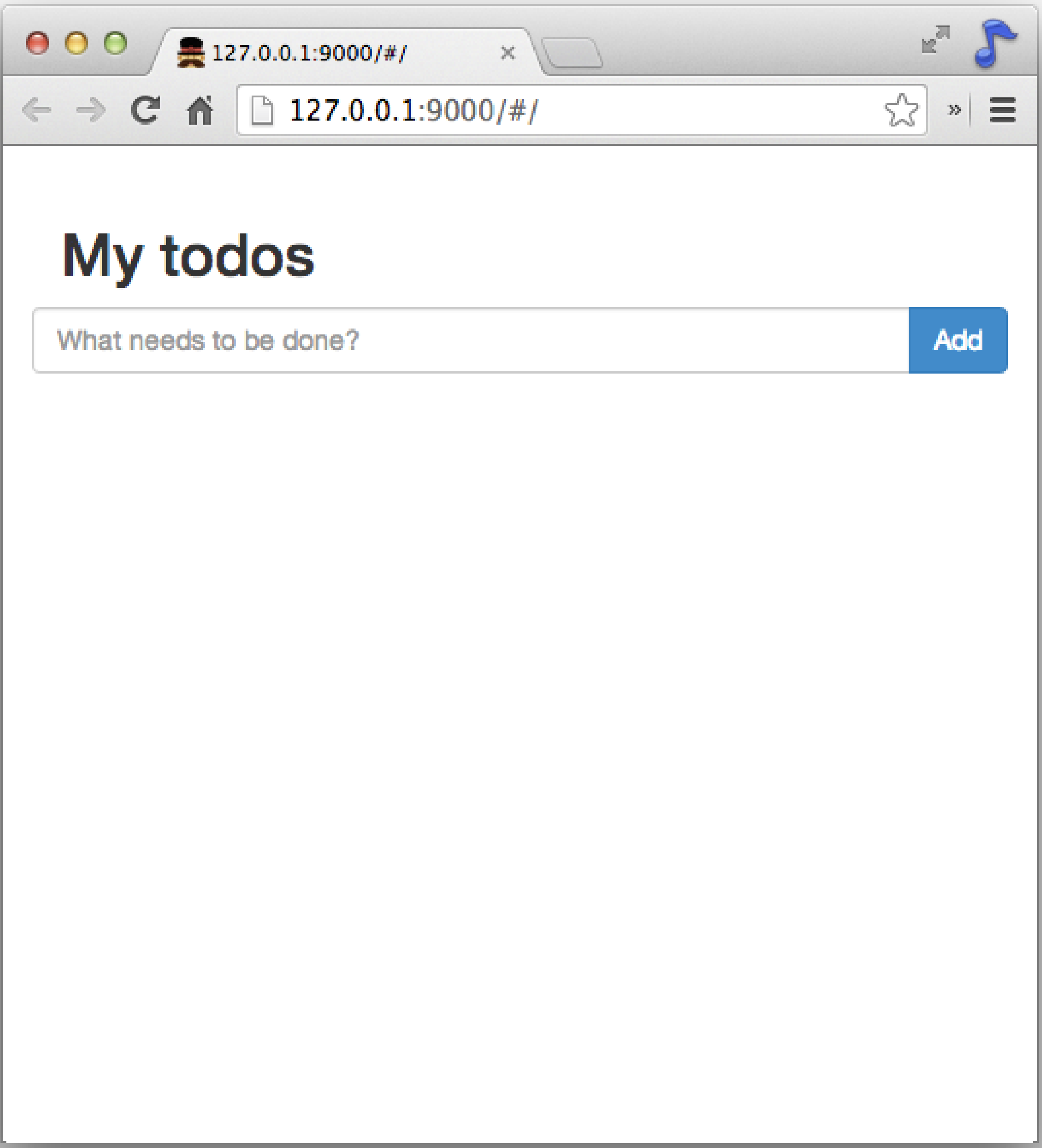
And replace it with this:

```
var todosInStore = localStorageService.get('todos');

$scope.todos = todosInStore && todosInStore.split('\n') || [];

$scope.$watch('todos', function () {
  localStorageService.add('todos', $scope.todos.join('\n'));
}, true);
```

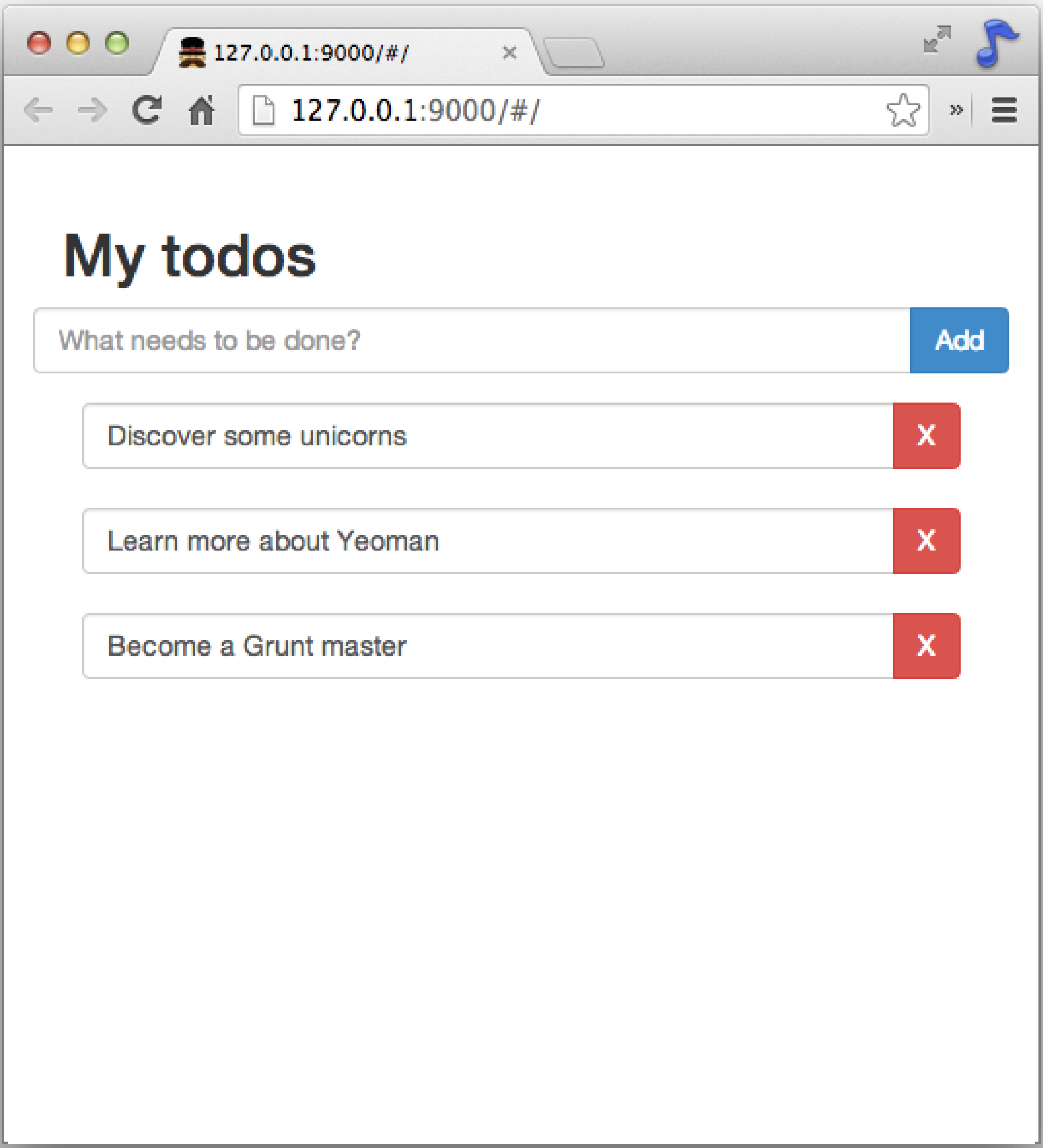
We now have a controller that is as follows:

```
'use strict';

angular.module('mytodoApp')
  .controller('MainCtrl', function ($scope, localStorageService) {

    var todosInStore = localStorageService.get('todos');

    $scope.todos = todosInStore && todosInStore.split('\n') || [];
    $scope.$watch('todos', function () {
      localStorageService.add('todos', $scope.todos.join('\n'));
    }, true);

    $scope.addTodo = function () {
      $scope.todos.push($scope.todo);
      $scope.todo = '';
    };

    $scope.removeTodo = function (index) {
      $scope.todos.splice(index, 1);
    };

  });
```

If you look at your app in the browser now you'll see that there are no items in the todo list. The app is initialising the todos array from local storage and we haven't given it any todo items yet.

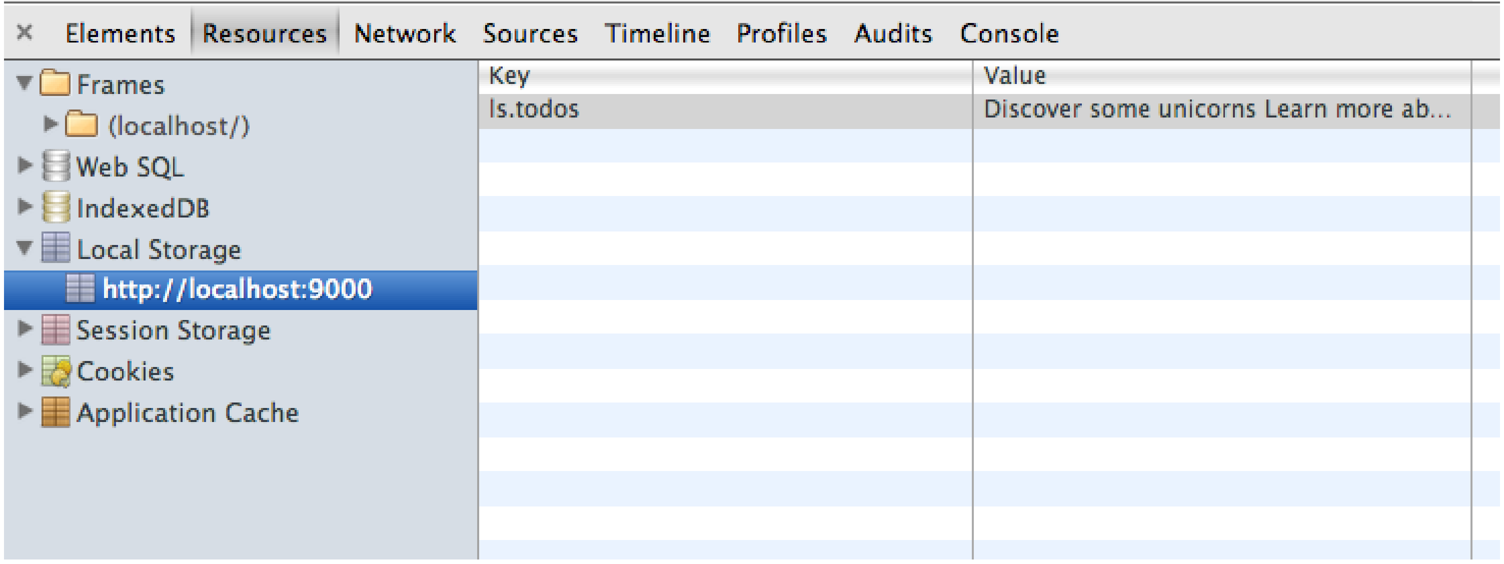


Go ahead and add a few items to the list:



Now when we refresh our browser the items persist. Hooray!

We can confirm whether our data is being persisted to local storage by checking the **Resources** panel in Chrome DevTools and selecting **Local Storage** from the lefthand side:



Key	Value
ls.todos	Discover some unicorns Learn more ab...

WRITE UNIT TESTS

For an extra challenge, revisit unit testing in Step 8 and consider how you might update your tests now that the code is using local storage.

Tip: It's not a straight forward answer and involves knowing about mock services. Check out Unit Testing Best Practices in AngularJS, specifically the *Mocking Services and Modules in AngularJS* section.

« Return to overview or **Go to the next step »**