HACK YALE

< ADVANCED JAVASCRIPT />

WWW.HACKYALE.COM

HACK YALE

< ADVANCED JAVASCRIPT />

DAY 3

ONWARD TO ANGULAR

HAPPY HUMP WEEK

Halfway done (almost)! The agenda:

- Feedback
- Homework review
- **Angular JS!**

FEEDBACK

http://goo.gl/LMY91x

HACK YALE

HOMEWORK REVIEW

HACK YALE

COMPARE NOTES

CHECK IN WITH THE PEOPLE NEAR YOU —
WHAT DID YOU DO SIMILARLY?
WHAT WAS DIFFERENT?

MY SOLUTION: SAMPLE OBJ

```
21  var rafi = {
22    fname: "Rafi",
23    lname: "Khan",
24    favoriteCereal: "Special K ;)",
25    interests: ["coding", "sleeping"],
26    fullname: function() {
27      return this.fname + " " + this.lname;
28    },
29    miniBio: function() {
30      toPrint = "Hi my name is " + this.fullname();
31      toPrint += " and my favorite cereal is " + this.favoriteCereal;
32      toPrint += " In my free time, I like to do ";
33      for (var i in this.interests) {
34        toPrint += this.interests[i] + " ";
35      }
36      console.log(toPrint);
37      return toPrint;
38    }
39  }
```

MY SOLUTION: SAMPLE OBJ

```
Hi my name is Rafi Khan and my favorite cereal is Special K ;) In my free time, I like to do coding sleeping
```

MY SOLUTION: MOUSE MOVE

```
$(window).mousemove(function(e) {
  // Ex. 3: My Shadow code here
  setTimeout(function() {
    $("#follow-dot").css({left: e.pageX + 10, top :e.pageY + 10 });
  }, 200);
});
```

MY SOLUTION: FACEBOOK

```
var getFB = function() {
  $.ajax({
    method: "get",
    url: "https://graph.facebook.com/me",
    data: {
      fields: "name,picture",
      // Access token obtained at https://d
      access_token: "CAACEdEose0cBAHyMA26VA
    },
    success: function(response) {
      console.log(response);
      picture = response.picture.data.url;
      $("body").prepend(
        $("<p>").html(response.name),
        $("<img>").attr("src", picture)
      );
    }
  });
};
```

ANGULAR JS

HACK YALE

HOW EXCITING!

Angular is a “hot” framework!

- Launched by Google in 2009
- Leverages AJAX and two-way data binding to easily create single-page apps
- Router - Controller - View - Model for the front-end

BUT ALSO BEWARE

Angular is a hard framework

- Easy to initially pick up
- Then gets *very* difficult
  - Documentation is shaky :(
- Mostly because there are a lot of new concepts to learn!
  - scopes, data binding, the digest cycle, HTTP promises, dependency injection…

KEEP THAT IN MIND

Take the time to try to understand as much as possible

- Look up things you don't understand
- Be patient!

3 PROBLEMS

THAT ANGULAR SOLVES

1: KEEPING FRONT-END UP-TO-DATE

Most user interaction cycles go like this:

- Get data from somewhere
- Show user data
- Get data from user
- Send (save) data somewhere

1: KEEPING FRONT-END UP-TO-DATE

In jQuery:

- Get data: $.get(…)
- Show data: $(“#list”).append(…)
- Get user data: $(“button”).click (…), $(“input”).val()
- Send user data: $.post(…)

1: KEEPING FRONT-END UP-TO-DATE

In Angular:

- Get: $http.get(…)
- Send: $http.post(…)

- The displaying to and getting from user happens automatically!
  - Thanks to data-binding and scopes

FREEDOM!

No more:

- Callbacks
- Manipulating the DOM programmatically

Instead:

- Templates - define where data goes
- Angular directives - define user interactions

2: THINKING ABOUT DATA

Rails has “models” because they help programmers conceptualize data.

Same for Angular! (but they’re called resources)

3: WHERE DOES MY CODE GO?

jQuery:

- Usually in one file, or in one file per controller (i.e. Rails)
- All the code is run always

Angular:

- In modules defined by functionality, not by page, that are loaded using dependency injection
- Only the code you need gets run!


THE ANGULAR WAY
KEY CONCEPTS

THE ANGULAR ZEN

- The DOM and app logic should be separate
- Testing code is just as important as writing code
- Client and server side code should be separate
- Frameworks should guide programmers through the entire journey: UI, logic, testing
- Make common tasks trivial and difficult tasks possible

KEY CONCEPT 1:

HTML defines reactivity

- Using directives (ng-…)
- Some are like “callbacks”
  - ng-click, ng-mouseover
- Others show data
  - ng-repeat, ng-show

KEY CONCEPT 2:

Functions go in controllers

- Controllers contain all logic
  - What *happens* when you click or hover?
- Are connected to elements by a *scope*

KEY CONCEPT 3:

Data is made visible with templates

- Once you put data in the DOM, it stays *automatically updated!*
  - If the value changes, the user will immediately see
- This is called *data-binding*


ENOUGH CHIT CHAT
LET'S CODE!

THE FRUITS OF OUR LABOR

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.cc
       min.js"></script>
5  </head>
6  <body ng-app>
7    <input type="text" ng-model="myText">
8    {{myText}}
9  </body>
10 </html>
```

THE FRUITS OF OUR LABOR

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.co
       min.js"></script>
5  </head>
6  <body ng-app>
7    <input type="text" ng-model="myText">
8    {{myText}}
9  </body>
10 </html>
```

Include Angular: usually you'll pull in Angular from the Google CDN (or a gem, if you're using Rails)

THE FRUITS OF OUR LABOR

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.co
       min.js"></script>
5  </head>
6  <body ng-app>
7    <input type="text" ng-model="myText">
8    {{myText}}
9  </body>
10 </html>
```

ng-app: lets Angular know that we want to use Angular here.

Note: you can have only parts of your page use Angular!

THE FRUITS OF OUR LABOR

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.co
       min.js"></script>
5  </head>
6  <body ng-app>
7    <input type="text" ng-model="myText">
8    {{myText}}
9  </body>
10 </html>
```

ng-model: *binds* the value of this input to the variable myText.

Whenever the value changes, myText changes, in this scope

THE FRUITS OF OUR LABOR

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.cc
       min.js"></script>
5  </head>
6  <body ng-app>
7    <input type="text" ng-model="myText">
8    {{myText}}
9  </body>
10 </html>
```

{{}} - a template directive that injects the data into the DOM.

It's bound to the value of myText, so whenever myText changes, the screen automatically updates.


MORE COMPLEXITY
CONTROLLERS!

THE JOYS OF CONTROLLERS

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.com/ajax/libs
       min.js"></script>
5    <script src="./script.js"></script>
6  </head>
7  <body ng-app="HelloAngular">
8    <div ng-controller="HelloCtrl">
9      <input type="text" ng-model="myText">
10     {{myText}}
11     <button ng-click="logText()">click me</button>
12   </div>
13 </body>
14 </html>
```

When we want custom code, we'll have to add our own script file

THE JOYS OF CONTROLLERS

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.com/ajax/libs
       min.js"></script>
5    <script src="./script.js"></script>
6  </head>
7  <body ng-app="HelloAngular">
8    <div ng-controller="HelloCtrl">
9      <input type="text" ng-model="myText">
10     {{myText}}
11     <button ng-click="logText()">click me</button>
12   </div>
13 </body>
14 </html>
```

Similarly, because there may be other Angular apps on the page, it's good practice to name ours

THE JOYS OF CONTROLLERS

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.com/ajax/libs
       min.js"></script>
5    <script src="./script.js"></script>
6  </head>
7  <body ng-app="HelloAngular">
8    <div ng-controller="HelloCtrl">
9      <input type="text" ng-model="myText">
10     {{myText}}
11     <button ng-click="logText()">click me</button>
12   </div>
13 </body>
14 </html>
```

ng-controller: specifies a controller *for this DOM element and its children only*.

THE JOYS OF CONTROLLERS

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.com/ajax/libs
       min.js"></script>
5    <script src="./script.js"></script>
6  </head>
7  <body ng-app="HelloAngular">
8    <div ng-controller="HelloCtrl">
9      <input type="text" ng-model="myText">
10     {{myText}}
11     <button ng-click="logText()">click me</button>
12   </div>
13 </body>
14 </html>
```

ng-click: What happens when this DOM element is clicked

THE JOYS OF CONTROLLERS

```
1  <!DOCTYPE html>
2  <head>
3    <!-- Include Angular -->
4    <script src="http://ajax.googleapis.com/ajax/libs
       min.js"></script>
5    <script src="./script.js"></script>
6  </head>
7  <body ng-app="HelloAngular">
8    <div ng-controller="HelloCtrl">
9      <input type="text" ng-model="myText">
10     {{myText}}
11     <button ng-click="logText()">click me</button>
12   </div>
13 </body>
14 </html>
```

A custom function we define in our controller

THE CONTROLLER SCRIPT

```
1  var helloAngular = angular.module("HelloAngular", []);
2
3  helloAngular.controller("HelloCtrl", function($scope) {
4    $scope.logText = function() {
5      console.log($scope.myText);
6    }
7  });
```

`angular.module`: gets the specified angular app

THE CONTROLLER SCRIPT

```
1  var helloAngular = angular.module("HelloAngular", []);
2
3  helloAngular.controller("HelloCtrl", function($scope) {
4    $scope.logText = function() {
5      console.log($scope.myText);
6    }
7  });
```

Other apps/modules your app depends on.

If none, **put an empty array!**

THE CONTROLLER SCRIPT

```
1  var helloAngular = angular.module("HelloAngular", []);
2
3  helloAngular.controller("HelloCtrl", function($scope) {
4    $scope.logText = function() {
5      console.log($scope.myText);
6    }
7  });
```

Add a controller with the .controller function

THE CONTROLLER SCRIPT

```
1  var helloAngular = angular.module("HelloAngular", []);
2
3  helloAngular.controller("HelloCtrl", function($scope) {
4    $scope.logText = function() {
5      console.log($scope.myText);
6    }
7  });
```

First parameter: the name

By convention, controllers start with capital letters and end with Ctrl

THE CONTROLLER SCRIPT

```
1  var helloAngular = angular.module("HelloAngular", []);
2
3  helloAngular.controller("HelloCtrl", function($scope) {
4    $scope.logText = function() {
5      console.log($scope.myText);
6    }
7  });
```

Second parameter: a function with *dependencies to inject*

All controllers will usually inject $scope. We'll cover dependency injection in more detail later.

THE CONTROLLER SCRIPT

```
1  var helloAngular = angular.module("HelloAngular", []);
2
3  helloAngular.controller("HelloCtrl", function($scope) {
4    $scope.logText = function() {
5      console.log($scope.myText);
6    }
7  });
```

Define all your behavior here.

THE CONTROLLER SCRIPT

```
1  var helloAngular = angular.module("HelloAngular", []);
2
3  helloAngular.controller("HelloCtrl", function($scope) {
4    $scope.logText = function() {
5      console.log($scope.myText);
6    }
7  });
```

Anything that should be accessible from the DOM **must be** attached to the $scope object.

A very common bug is to forget this point

THE CONTROLLER SCRIPT

```
1  var helloAngular = angular.module("HelloAngular", []);
2
3  helloAngular.controller("HelloCtrl", function($scope) {
4    $scope.logText = function() {
5      console.log($scope.myText);
6    }
7  });
```

Likewise, since myText is visible in the DOM, we have to access it with $scope.myText

SCOPES

DOM <-> CONTROLLER

SCOPES

SCOPES HELP MANAGE THE VARIABLES AND FUNCTIONS ACCESSIBLE TO A DOM ELEMENT OR DIRECTIVE

SCOPES

Must be created, but usually are created for you

- Directives that create them for you:
  - ng-controller, ng-repeat, ng-include, many others
- Then those scope variables are only attached to

SCOPES

In our case: all data in the DOM is attached to the $scope object of its corresponding controller

- This includes all methods and variables
- Remember: any time you're creating or accessing data or methods that need to be in both the DOM and controller, use $scope

OUR SCOPE

```
7  <body ng-app="HelloAngular">
8    <div ng-controller="HelloCtrl">
9      <input type="text" ng-model="myText">
10     {{myText}}
11     <button ng-click="logText()">click me</button>
12   </div>
13 </body>
```

The scope created by the ng-controller directive

OUR SCOPE

```
7  <body ng-app="HelloAngular">
8    <div ng-controller="HelloCtrl">
9      <input type="text" ng-model="myText">
10     {{myText}}
11   </div>
12     <button ng-click="logText()">click me</button>
13 </body>
```

Will this work?

OUR SCOPE

```
7  <body ng-app="HelloAngular">
8    <div ng-controller="HelloCtrl">
9      <input type="text" ng-model="myText">
10     {{myText}}
11   </div>
12   <button ng-click="logText()">click me</button>
13 </body>
```

No: the logText() method is outside the scope, so Angular has no idea what it is.

Note: no error thrown! Silently fails :(


A “REAL” APP
THE CLASSIC TODO LIST

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

TODO: GO OVER THIS

the ng-controller's scope

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

TODO: GO OVER THIS

ng-repeat creates new scope!

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

syntax

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

TODO: GO OVER THIS

Where does toggleDone live?

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

What does ng-submit do?

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

Why is it newItem.*text*?

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

What does ng-hide do?

TODO: GO OVER THIS

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

How does ng-class work?

TODO: GO OVER THIS

Where does $index come from?

```
16  <body ng-app="Todo">
17    <div ng-controller="TodoCtrl">
18      <ul id="list">
19        <li ng-repeat="item in todoList">
20          <span class="item" ng-class="{done: item.done}">{{$index + 1}}: {{item.
            text}}</span>
21          <span class="link" ng-click="toggleDone($index)" ng-hide="item.done">
            done!</span>
22          <span class="link" ng-click="toggleDone($index)" ng-hide="!item.done">
            undo</span>
23        </li>
24      </ul>
25      <form ng-submit="addItem()">
26        <input type="text" ng-model="newItem.text">
27        <input type="submit" value="Add item">
28      </form>
29    </div>
30  </body>
31  </html>
```

TODO: GO OVER THIS

```
1  angular.module("Todo", []).controller("TodoCtrl", function($scope) {
2    $scope.newItem = {}
3    $scope.todoList = [
4      { text: "Clean room" },
5      { text: "Do homework"},
6      { text: "Pump iron" }
7    ]
8    $scope.addItem = function() {
9      if ($scope.newItem.text) {
10       $scope.todoList.push($scope.newItem);
11       $scope.newItem = {};
12     }
13   }
14   $scope.toggleDone = function(index) {
15     $scope.todoList[index].done = !$scope.todoList[index].done;
16   }
17 });
```

TODO: GO OVER THIS

```
1  angular.module("Todo", []).controller("TodoCtrl", function($scope) {
2    $scope.newItem = {}
3    $scope.todoList = [
4      { text: "Clean room" },
5      { text: "Do homework"},
6      { text: "Pump iron" }
7    ]
8    $scope.addItem = function() {
9      if ($scope.newItem.text) {
10       $scope.todoList.push($scope.newItem);
11       $scope.newItem = {};
12     }
13   }
14   $scope.toggleDone = function(index) {
15     $scope.todoList[index].done = !$scope.todoList[index].done;
16   }
17 });
```

TODO: GO OVER THIS

```
1  angular.module("Todo", []).controller("TodoCtrl", function($scope) {
2    $scope.newItem = {}
3    $scope.todoList = [
4      { text: "Clean room" },
5      { text: "Do homework"},
6      { text: "Pump iron" }
7    ]
8    $scope.addItem = function() {
9      if ($scope.newItem.text) {
10       $scope.todoList.push($scope.newItem);
11       $scope.newItem = {};
12     }
13   }
14   $scope.toggleDone = function(index) {
15     $scope.todoList[index].done = !$scope.todoList[index].done;
16   }
17 });
```

TODO: GO OVER THIS

```
1  angular.module("Todo", []).controller("TodoCtrl", function($scope) {
2    $scope.newItem = {}
3    $scope.todoList = [
4      { text: "Clean room" },
5      { text: "Do homework"},
6      { text: "Pump iron" }
7    ]
8    $scope.addItem = function() {
9      if ($scope.newItem.text) {
10       $scope.todoList.push($scope.newItem);
11       $scope.newItem = {};
12     }
13   }
14   $scope.toggleDone = function(index) {
15     $scope.todoList[index].done = !$scope.todoList[index].done;
16   }
17 });
```

TODO: GO OVER THIS

```
1  angular.module("Todo", []).controller("TodoCtrl", function($scope) {
2    $scope.newItem = {}
3    $scope.todoList = [
4      { text: "Clean room" },
5      { text: "Do homework"},
6      { text: "Pump iron" }
7    ]
8    $scope.addItem = function() {
9      if ($scope.newItem.text) {
10       $scope.todoList.push($scope.newItem);
11       $scope.newItem = {};
12     }
13   }
14   $scope.toggleDone = function(index) {
15     $scope.todoList[index].done = !$scope.todoList[index].done;
16   }
17 });
```

TODO: GO OVER THIS

```
1  angular.module("Todo", []).controller("TodoCtrl", function($scope) {
2    $scope.newItem = {}
3    $scope.todoList = [
4      { text: "Clean room" },
5      { text: "Do homework"},
6      { text: "Pump iron" }
7    ]
8    $scope.addItem = function() {
9      if ($scope.newItem.text) {
10       $scope.todoList.push($scope.newItem);
11       $scope.newItem = {};
12     }
13   }
14   $scope.toggleDone = function(index) {
15     $scope.todoList[index].done = !$scope.todoList[index].done;
16   }
17 });
```

TODO: GO OVER THIS

```
1  angular.module("Todo", []).controller("TodoCtrl", function($scope) {
2    $scope.newItem = {}
3    $scope.todoList = [
4      { text: "Clean room" },
5      { text: "Do homework"},
6      { text: "Pump iron" }
7    ]
8    $scope.addItem = function() {
9      if ($scope.newItem.text) {
10       $scope.todoList.push($scope.newItem);
11       $scope.newItem = {};
12     }
13   }
14   $scope.toggleDone = function(index) {
15     $scope.todoList[index].done = !$scope.todoList[index].done;
16   }
17 });
```