SERVICES & SERVICE FACTORIES

SERVICES

Services are reusable components that are separated from views and can be used across multiple controllers and views.

CREATING A SERVICE

There are several methods for creating services. Two are:

- `service()` method
- Service `factory()` method

THE SERVICE METHOD

```
var app = angular.module('ourApp', []);

app.controller('ourCtrl', function($scope, ourService) {
  ourService.ourFunction();
});

app.service('ourService', function() {
  this.ourFunction = function() {
    // awesome code
  };
});
```

THE SERVICE FACTORY METHOD

A factory is another way to create a service.

With a factory, you create an object, add properties, and then return that object literal. Once injected into the controller, the properties of that object are accessable to the controller.

THE SERVICE FACTORY METHOD

```
var app = angular.module('ourApp', []);

// The service is used exactly the same way in controllers.
app.controller('ourCtrl', function($scope, ourService) {
  ourService.ourFunction();
};

// The resulting service is the same, but the method
// of creating it is different.
app.factory('ourService', function() {
  return {
    ourFunction: function() {
      // awesome code
    }
  };
});
```

A NOTE ON FACTORIES AND SERVICES

There are subtle differences between services and factories. There is no right or wrong choice. The debate on whether to use a factory or service has been on-going for a few years now.

MAKING THE DECISION

There are a handful of great style guides and best practice guides out there for Angular. The common theme among many high-end developers of Angular is to just stick with factories.

BUILT-IN SERVICES

There are also a number of built-in services. One of the most useful is `$http`. It works similarly to jQuery's `$.get()` to make AJAX calls.

Two other common services are `$timeout` and `$location`. This is a list of build-in services.

$HTTP ...THEN

We're going to be using `$http` a lot. Tomorrow you'll have a chance to try it.

But here's an preview.

YOU KNOW WHAT TIME IT IS...

DEMO!

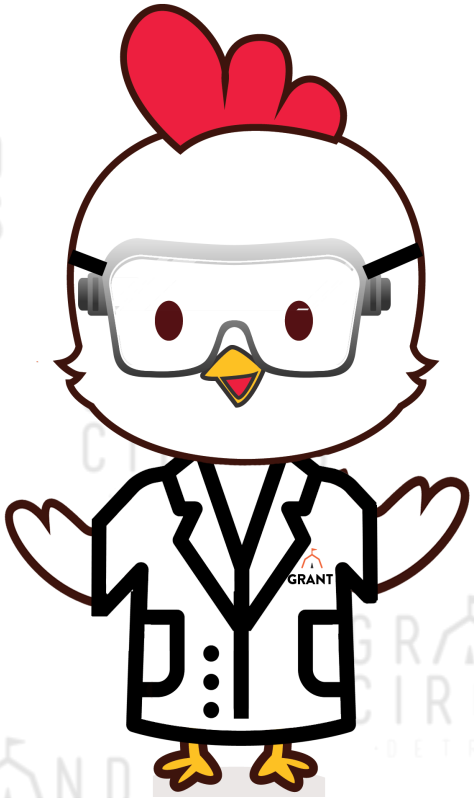
HOMEWORK

From ng-book:

- Introduction to Direcives: 64-80
- Directives Explained: 104-134

LAB 15

MAD LIBS



MAD LIBS

Make a Mad Libs-type web application that allows a user to enter information in one view and see all of that information displayed and formatted in another.

INSTRUCTIONS

- Create two views: A form with inputs, and a display page.
- Set up routes for both views, provide some navigation solution to move between each.
- In the form view, ask the user for at least 7 strings (e.g. adjective, noun, verb).
- Each view should have its own controller.
- When the user submits the form data, persist it so that it may be used for multiple controllers.
- In the display view, format and display the data entered in the form view.

PRE-GAME

Another Mad Lib!

form | viewer

But we are all in the hands of the pigeons; and The Fonz jumped again. It was under very similar circumstances to the first chair; but this time he did not boogie board out the line; and hence, when the whale started to run, The Fonz was left behind on the sea, like a hurried shade. Alas! Waldorf was but too true to his word. It was a chunky, sparkling, boorish day; the spangled sea calm and cool, and flatly stretching away, all round, to the horizon, like gold-beater's skin hammered out to the extremest.

DEMOS!

BONUS EXERCISE

WORKING WITH EXTERNAL DATA

AJAX

One of the most common operations for web applications is to make API calls to external sources. This is commonly called AJAX requests, though that name is not entirely accurate anymore (we're using JSON mostly now).

INSTRUCTIONS

- Expand the last lab to include external data.
- Use the `$http` service to pull in external data.
- Use the data from https://twlaas.herokuapp.com/.
- Display the data somewhere in your information view.
- You may need to use ng-repeat to display multiple things.
- You'll need to get used to parsing through the body of the json to get to the data you want.

PRE-GAME