# AngularJS: Lesson 8

## Forms, Promises, Futures and ngResource

The New York Code + Design Academy

AngularJS by Google

# To Form or not to form

nested forms use directive: ng-form

Forms get you:

- one place to attach your submit handler
- submit on “enter” keypress

# Syntax

## nested forms use directive: ng-form

```
<form name="ctrl.myForm" novalidate ng-submit="onMySubmitAction()">
     <input name="myInput" type="text" ng-model="ctrl.myModelValue">
     <input name="subInput" type="submit" value="submit">
</form>

<div ng-form name="myForm" novalidate>
     <input name="myInput" type="text" ng-model="myModelValue">
     <input type="submit">Submit
</div>
```

# Built-In Validators

```
<form name=”myForm” novalidate>
     <input name=”textInput” type=”text” required ng-model=”someText”>
     <input name=”myInput” type=”number” min=”0” max=”99” ng-model=”myModelValue”>
     <input name=”emailInput” type=”email” ng-model=”userEmail”>
     <input name=”subInput” type=”submit” value=”submit”>
</form>
```

Others: https://docs.angularjs.org/api/ng/input

# Promises vs Futures

Design Pattern that offers a way of dealing with asynchronous events.

# Promise

A function that is guaranteed to be called.

$q for creating promises.

# Let’s see an example

# Future

A function you define and pass in that is guaranteed to be called sometime in the future.

ex: $watch(‘property’, future)

# CRUD Operations and REST Calls

Create:  POST (put data in request body)
Retrieve: GET (everything goes in URL)
Update: PUT (put data in request body)
Delete: DELETE (url to be safe)

# ngResource

Angular Service that allows us to make REST calls

$resource(url, [paramDefaults], [actions], options)

# URL placeholders

```
$resource('/assets/JSON/:userId/profile.json',
```

# Param Defaults

An Object that serves two purposes:

1) Fills in URL placeholders
2) Adds query parameters to URL

# Actions

Custom sub-functions we can specify

https://docs.angularjs.org/api/ngResource/service/$resource