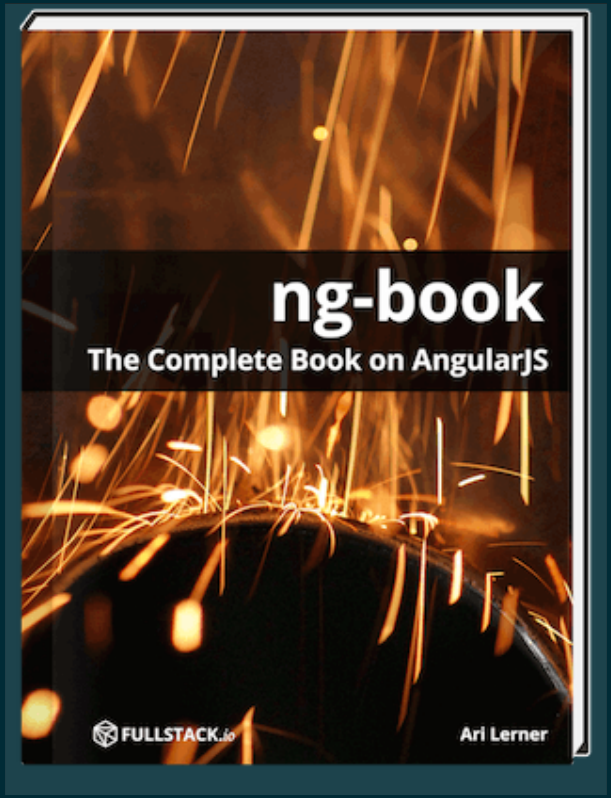
BEGINNER WORKSHOP TO ANGULARJS

WHO AM I?

ARI LERNER, FULLSTACK.IO

- Author of ng-book and ng-newsletter
- Author of a few others (D3 on Angular, Riding Rails with AngularJS)
- Teacher at HackReactor, General Assembly
- Co-founder of Fullstack.io
- Background in distributed computing and infrastructure

NG-BOOK.COM


ng-book
The Complete Book on AngularJS
FULLSTACK.io
Ari Lerner

TOOLS

TEXT EDITOR

- Sublime Text 2/3
- Textmate
- Vim
- Emacs

WEB BROWSER

- Chrome
- Firefox
- Safari

WEB SERVER

Python

```
 $ python -m SimpleHTTPServer 8000
# or
$ twistd -no web --path .
```

WEB SERVER

NodeJS

```
$ npm install http-server -g
$ http-server ./ -p 8000
```

WEB SERVER

Golang

```
package main

import (
    "fmt"; "log"; "net/http"
)

func main() {
    fmt.Println("Serving the current directory on port 8080")
    http.Handle("/", http.FileServer(http.Dir(".")))
    err := http.ListenAndServe(":8080", nil)
    if err != nil {
        log.Fatal("ListenAndServe: ", err)
    }
}
```

WEB SERVER

Serve the current directory

FLICKR API KEY

Flickr Services

www.flickr.com/services/api/

Mail News Sports Finance Weather Games Groups Answers Screen Flickr Mobile More

flickr Sign Up Explore Upload Sign In

The App Garden

Create an App | API Documentation | Feeds | What is the App Garden?

The Flickr API is available for non-commercial use by outside developers. Commercial use is possible by prior arrangement.

Read these first:

- Developer Guide
- Overview
- Encoding
- User Authentication

- Dates
- Tags
- URLs
- Buddyicons

- Flickr APIs Terms of Use

- API Keys
- Developers' mailing list

Photo Upload API

- Uploading Photos
- Replacing Photos
- Example Request
- Asynchronous Uploading

Request Formats

API Methods

activity

- flickr.activity.userComments
- flickr.activity.userPhotos

auth

- flickr.auth.checkToken
- flickr.auth.getFrob
- flickr.auth.getFullToken
- flickr.auth.getToken

auth.oauth

- flickr.auth.oauth.checkToken
- flickr.auth.oauth.getAccessToken

blogs

- flickr.blogs.getList
- flickr.blogs.getServices
- flickr.blogs.postPhoto

cameras

- flickr.cameras.getBrandModels

Open "http://www.flickr.com/services/api/keys/" in a new tab

BUILDING OUR FIRST APP

Enter a name here

HELLO, !

DEMO

```
<html ng-app>
  <head>
    <title>Test</title>
  </head>
  <body>
    <input type="text" ng-model="yourName" placeholder="Enter a name here">
    <h3>Hello {{ yourName }}!</h3>
    <script
      src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.12/angular.min.js">
    </script>
  </body>
</html>
```

- `ng-app`
- `ng-model="yourName"`
- `{{ yourName }}`

DEFINE AN APPLICATION

DEFINE A MODULE

```
// Setter
angular.module('myApp', []);
// Getter
angular.module('myApp')
```

INVOKING OUR APPLICATION

ng-app

INVOKING OUR APPLICATION

```
<html>
  <head>
    <script
      src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.12/angular.min.js">
    </script>
  </head>
  <body ng-app="myApp">
    <!-- Application runs from here -->
  </body>
</html>
```

INVOKING OUR APPLICATION

```
<html>
  <head>
    <script
      src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.12/angular.min.js">
    </script>
  </head>
  <body ng-app="myApp">
    <!-- Application runs from here -->
  </body>
</html>
```

```
body.ngApp('myApp');
```

DIRECTIVE?!??

DIRECTIVES ARE FUNCTIONS

```
function say(msg) {
  alert("Hi " + msg);
}
```

...RUN ON ELEMENTS

```
function say(msg) {
  alert("Hi " + msg);
}
```

```
<div say="world"></div>
```

Say

BUILT-IN DIRECTIVES

NG-CLICK

```
<h4>State is {{ state }}</h4>
<button ng-click="state = !state">Change state</button>
```

STATE IS

Change state

NG-SHOW/NG-HIDE

```
<button ng-click="state = !state">Change state</button>
<div ng-show="state">
  <h4>I'm out of hiding when the state is true</h4>
  <h4>Weeeee</h4>
</div>
<div ng-hide="state">
  <h4>BOO!</h4>
  <h4>I'm only showing when the state is not true</h4>
</div>
```

Change state

BOO!

I'M ONLY SHOWING WHEN THE STATE IF NOT TRUE

NG-MODEL

```
<input ng-model="name" type="text" placeholder="Enter your name" />
<h4>{{ name }}</h4>
```

Enter your name

NG-REPEAT

```
<ul ng-init="names=['Ari', 'Anand', 'Q', 'Colby']">
  <li ng-repeat="name in names">{{ name }}</li>
</ul>
```

- Ari
- Anand
- Q
- Colby

EXPRESSIONS

Angular expressions are similar to JavaScript and can be thought of like it.

```
<div>1 + 2 = {{ 1 + 2 }}</div>
<a ng-click="count = count + 1">Add one to the count</a>
```

EXPRESSIONS

Angular expressions are similar to J can be thought of li

```
<div>1 + 2 = {{ 1 + 2 }}</di
<a ng-click="count = cou che count</a>
```

They don't throw errors, which is a good thing.

SCOPES

Expressions have access to the variables inside the parent scope

DOING STUFF

AND WHAT ABOUT THE JAVASCRIPT?

SCOPES

The `$scope` object is the *glue* between our JavaScript and our view (HTML).

```
<h2>Welcome back {{ user.name }}</h2>
```

SCOPES

The `$scope` object is the *glue* be vaScript and our vie

```
<h2>Welcome back {{ user
```

Where is `user.name` defined?

$SCOPE

```
$scope.user = {
  name: 'Ari'
};
```

SCOPES ARE JUST POJOS

```
// Create variables
$scope.message = 'Hello';
// or objects
$scope.user = {
  name: 'Ari'
};
// Define functions
$scope.say = function(msg) {
  alert($scope.message + " " + msg);
}
```

$?

Just a name...

What's in a name?

name...

VIEW

```
<h2>{{ message }}</h2>
<h3>Welcome back {{ user.name }}</h3>
<buttton ng-click='say("World")'>Say hello</buttton>
```

HELLO

WELCOME BACK ARI

Say hello

HOW TO GET ACCESS TO THE $SCOPE

CONTROLLERS

A controller is a piece of code that defines functionality for a part of the page. Controllers are like mediators of functionality for portions of the page.

```
<div ng-controller="AddController">
  <h3>Count is at: {{ count }}</h3>
  <a ng-click="addOne()">Add one</a>
</div>
```

DEFINE A CONTROLLER

```
angular.module('myApp', [])
.controller('AddController', function($scope) {
  // We have access to the HomeController's $scope
  $scope.count = 0;
  $scope.addOne = function() {
    $scope.count += 1;
  }
});
```

DATA

Client-side applications are only as exciting as the data they contain.

GETTING DATA

`$http` is a wrapper on the browser's XMLHttpRequest API.

```
$http({
  method: 'GET',
  url: 'http://api.flickr.com/services/rest',
  params: {
    // Flickr API parameters
    method: 'flickr.interestingness.getList',
    api_key: apiKey,
    format: 'json',
    nojsoncallback: 1
  }
});
```

XHR REQUESTS ARE ASYNC

CALLBACKS

```
$.ajax({
  url : 'example.com',
  type: 'GET',
  success : function(data) {
    // Success :)!
  },
  error: function(reason) {
    // Failure :(
  }
})
```

CALLBACK HELL

```
fs.readdir(source, function(err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function(filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function(err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function(width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(destination + 'w' + width + '_' + filename, fu
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
});
```

PROMISES

Rather than passing a callback function into the `$http` method to get called when the data returns, the `$http` object returns a *promise*.

PROMISE API

```
promise
.then(function(data) {
  // Called when no errors have occurred with data
})
.catch(function(err) {
  // Called when an error has occurred
})
.finally(function(data) {
  // Called always, regardless of the output result
})
```

USING $HTTP

```
angular.module('myApp', [])
.controller('HomeController', function($scope, $http) {
  $scope.getPhotos = function() {
    $http({
      method: 'GET',
      url: 'http://api.flickr.com/services/rest',
      params: {
        // Flickr API parameters
        method: 'flickr.interestingness.getList',
        api_key: apiKey,
        format: 'json',
        per_page: 3,
        nojsoncallback: 1
      }
    }).then(function(data) {
      $scope.photos = data.data.photos.photo;
    });
  }
});
```

Get photos

BUT WAIT A MINUTE

How come we can even call the `$http` object?

```
angular.module('myApp', [])
.controller('HomeController', function($scope, $http) {
});
```

BUT WAIT A MINUTE

How come we can even c object?

```
angular.module('myApp', [])
.controller('HomeController'  $http) {
});
```

Magic?!??!

DEPENDENCY INJECTION

Anytime we rely on a library (%99.9999999999 of all code we write), it either:

- needs to find the dependency itself or
- needs to be handed the dependency

DEPENDENCY INJECTION

Angular handles this ugly process for us by making objects and *injecting* them for us when invoking the objects.

```
angular.module('myApp', [])
.controller('HomeController', function($scope, $http) {
  // Because we've named them
  //  $scope and $http
  // angular will provide these to our controller
});
```

REMEMBER OUR MODULE?

```
// the [] is a list of module dependencies
angular.module('myApp', []);
```

DEPENDENCY INJECTION

- naming matters
- order does not matter

DEPENDENCY INJECTION

- [illegible]ers
[illegible]es not matter

How to handle minification?

DEPENDENCY INJECTION

- [illegible]ers
[illegible]es not matter

I know a book that can help...

TEMPORARY `$scope` OBJECTS

Controllers are temporary objects and hang around only while they are needed.

SO HOW DO WE STORE DATA?

For instance, how do we keep a user logged in through the life-cycle of our application?

SERVICES

- Singleton objects that persist for the life-cycle of the application
- A container for like methods and data

BUILT-IN SERVICES

Like directives, Angular comes packed with services and providers (a special type of service)

- `$http`
- `$timeout`
- `$sce` (security)
- `$log`
- `$q`
- and more

CREATING OUR OWN SERVICE

```
angular.module('myApp')
.service('Flickr', function($http) {
  this.getPhotos = function() {
    // Define getPhotos() function
  }
});
```

CREATING OUR OWN SERVICE

```
angular.module('myApp')
.factory('Flickr', function($http) {
  return {
    getPhotos: function() {}
  }
});
```

CREATING OUR OWN SERVICE

```
angular.module('myApp')
.provider('Flickr', function() {
  var apiKey = '';
  this.setApiKey = function(key) {
    return apiKey = key || apiKey;
  }
  this.$get = function($http) {
    this.getPhotos = function() {};
    return this;
  }
});
```

CREATING OUR OWN SERVICE

The `provider()` is the only type of service we can use in the `config()` block as `[Name]Provider`

```
angular.module('myApp')
.config(function(FlickrProvider) {
  FlickrProvider.setApiKey('KEY');
});
```

CONFIG()

The `config()` function runs before our app is running and let's us *set up* the app.

RUN()

The `run()` function is the first function to get run before any other part of our app.

USING OUR SERVICE

Just like using Angulars!

```
angular.module('myApp')
.controller('HomeController', function($scope, Flickr) {
  Flickr.getPhotos()
  .then(function(data) {
    $scope.photos = data.photos;
  });
});
```

NEVER USE THE `$HTTP` IN A CONTROLLER

WHY?

WHAT ABOUT MULTIPLE PAGES?

ROUTING

INSTALLATION

```
<html ng-app>
  <head></head>
  <body>
    <script
      src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.12/angular.min.js">
    </script>
    <script
      src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.12/angular-route.min.js">
    </script>
  </body>
</html>
```

INSTALLATION

```
// Tell our Angular app of the new
// ngRoute module dependency
angular.module('myApp', ['ngRoute'])
```

LAYOUTS

We can now define our routes, but where how will they show up on the page?

LAYOUTS

We can now define our routes, but where how will they show up on the page?

```
<body>
  <header>Header</header>
  <div ng-view=""></div>
  <footer>Footer</footer>
</body>
```

LAYOUTS

`ngRoute` switches the child element of the `ngView` directive.

DEFINING ROUTES

The `ngRoute` module provides us with a new provider where we'll define routes called the `$routeProvider`.

```
angular.module('myApp')
.config(function($routeProvider) {
  // Configure our routes here
});
```

DEFINING ROUTES: WHEN()

The `when()` method allows us to define a route and a route configuration object.

```
angular.module('myApp')
.config(function($routeProvider) {
  $routeProvider
    .when('/', {
      templateUrl: 'templates/home.html',
      controller: 'HomeController'
    })
});
```

DEFINING ROUTES: OTHERWISE()

The `otherwise()` method defines a catch-all route if no other route matches:

```
angular.module('myApp')
.config(function($routeProvider) {
  $routeProvider
    .otherwise({
      redirectTo: '/'
    })
});
```

TESTING

TESTING

- Gives us assurance our code works as expected
- Allows for confidence with application deployment
- Knowledge transfer and maintainability
- ...

UNIT TESTING

Based off **Jasmine** syntax, Angular's Karma test runner runs our tests in a headless browser in several different browsers, such as Chrome, Safari, and PhantomJS.

UNIT TESTING

Focuses on testing small, atomic pieces of functionality.

```
describe('Unit controllers: ', function(){
  // Mock the myApp module
  beforeEach(module('myApp'));
  describe('HomeController', function() {
    // Local variables
    var HomeController, scope;
    beforeEach(inject(
      function($controller, $rootScope) {
        // Create a new child scope
        scope = $rootScope.$new();
        HomeController = $controller('HomeController', {
          $scope: scope
        });
    }));

    it('should have name set', function() {
      expect(scope.name).toBeDefined();
    });

  });
});
```

UNIT TESTING

Do it.

UNIT ng

o it.

Seriously, it will save your life.

END-TO-END TESTING

Instead of clicking the browser trying to mimic our user's actions...

E2E TESTING

Karma's test runner

E2E

test runner

Protractor

PROTRACTOR

Protractor is an end-to-end test framework built off the WebDriverJS browser automation framework.

END-TO-END TESTING OUR PAGE

```
describe('page load', function() {
  var link;
  beforeEach(function() {
    link = element(by.css('.header ul li:nth-child(2)'));
    link.click();
  });

  it('should navigate to the /about page', function() {
    expect(browser.getCurrentUrl()).toMatch(/\/about/);
  });
});
```

THANK YOU

NG-BOOK.COM

630+ page book with all this information and much much more.

The **only** constantly updating book available for Angular today.

ADDENDUM (JUST IN CASE)

CONTROLLERS HANDLE BUSINESS LOGIC

```
<div ng-controller="AddController">
  <h3>Count is at: {{ count }}</h3>
  <a ng-click="addOne()">Add one</a>
</div>
```

COUNT IS AT: 0

Add one