AngularJS + API Data

Let’s build a search engine!

About Tim

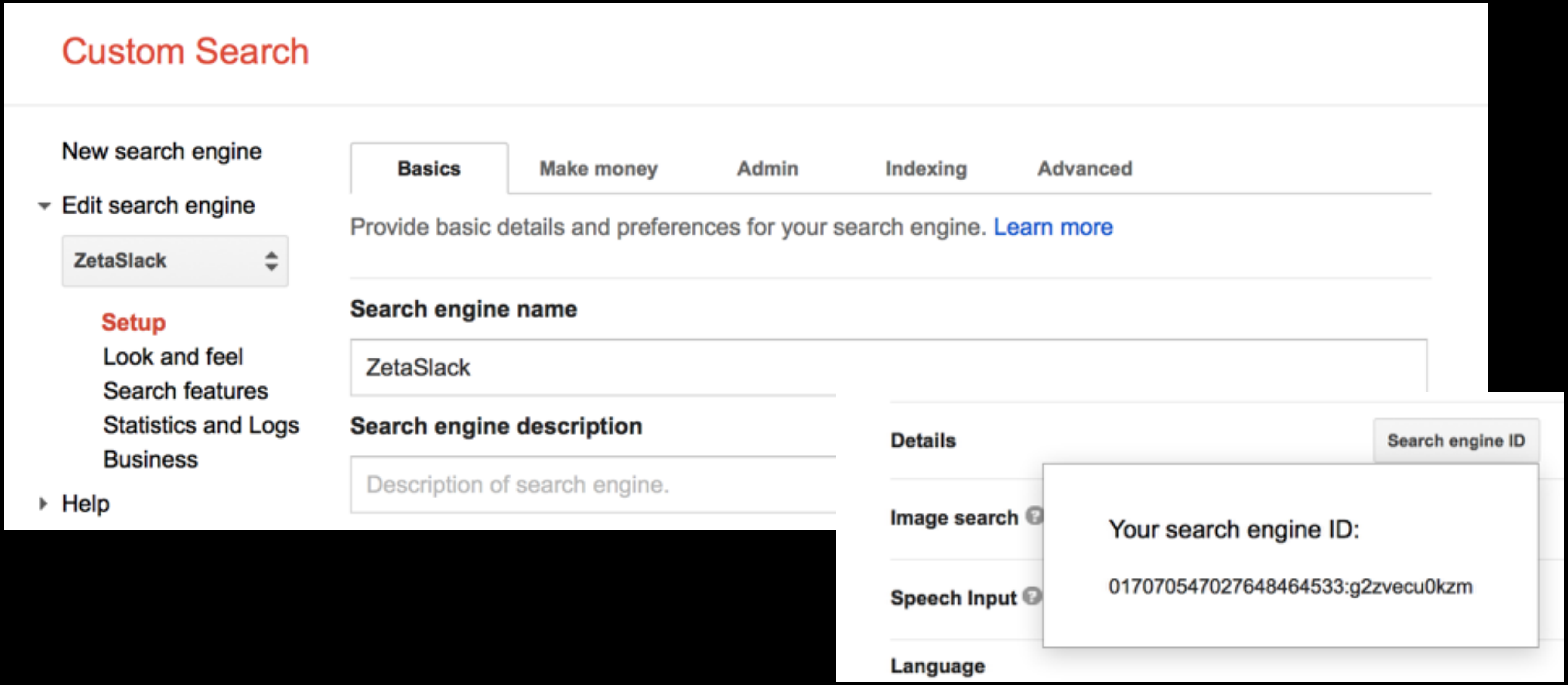
- Been in the industry since 2003 as a designer, developer, and aspiring writer.
- Currently working for CSG building sites for companies like Redbox and Paramount
- We’re using Angular every day.

Let's start with Angular

It's a really cool platform

```
<!doctype html>
  <html ng-app>
    <head>
      <script src="https://ajax.googleapis.com/ajax/libs/
angularjs/1.5.0/angular.min.js"></script>
    </head>
    <body>
      <div>
        <label>Name:</label>
        <input type="text" ng-model="yourName"
placeholder="Enter a name here">
        <hr>
        <h1>Hello {{yourName}}!</h1>
      </div>
    </body>
  </html>
```

ng-something is a built in Angular directive (or component)
ng-app is like ‘hey I’m an Angular app’.
ng-model binds an element to a piece of data.
The **{{dual brackets}}** are pretty much ng-model.


Custom Search

New search engine
Edit search engine
ZetaSlack
Setup
Look and feel
Search features
Statistics and Logs
Business
Help

Provide basic details and preferences for your search engine. Learn more

Search engine name: ZetaSlack

Search engine description: Description of search engine.

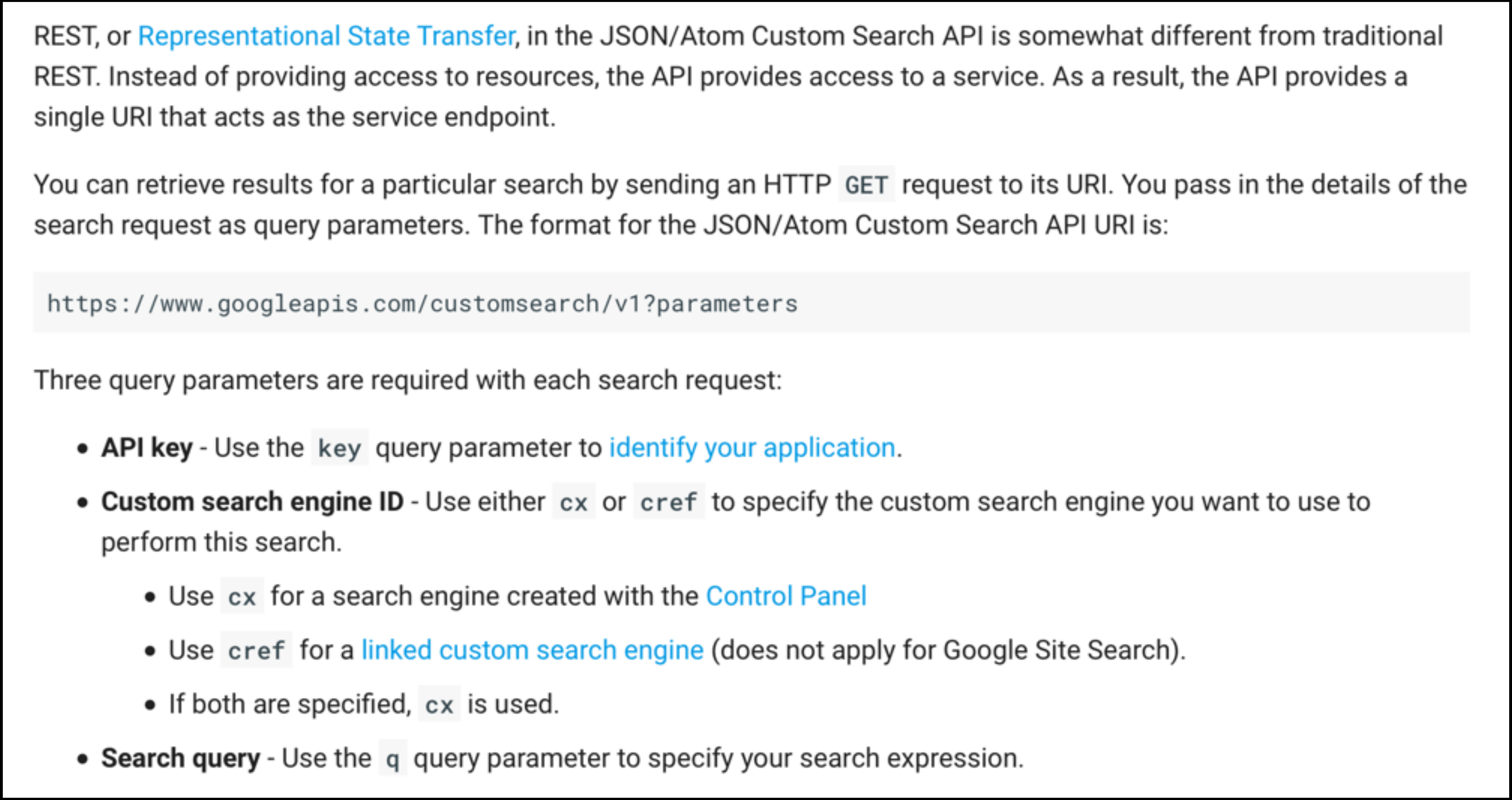
Details
Search engine ID
Image search
Speech Input
Language

Your search engine ID:

017070547027648464533:g2zvecu0kzm

For data, Google Custom Search

We're going to use Angular to tap into their API. It's a JSON feed, which Angular loves. Nice.



REST, or Representational State Transfer, in the JSON/Atom Custom Search API is somewhat different from traditional REST. Instead of providing access to resources, the API provides access to a service. As a result, the API provides a single URI that acts as the service endpoint.

You can retrieve results for a particular search by sending an HTTP `GET` request to its URI. You pass in the details of the search request as query parameters. The format for the JSON/Atom Custom Search API URI is:

```
https://www.googleapis.com/customsearch/v1?parameters
```

Three query parameters are required with each search request:

- **API key** - Use the `key` query parameter to identify your application.
- **Custom search engine ID** - Use either `cx` or `cref` to specify the custom search engine you want to use to perform this search.
    - Use `cx` for a search engine created with the Control Panel
    - Use `cref` for a linked custom search engine (does not apply for Google Site Search).
    - If both are specified, `cx` is used.
- **Search query** - Use the `q` query parameter to specify your search expression.

We can retrieve data by signing up for an API key, creating a search engine, and calling it from our app. Here's what our response looks like…

```
app/
----- app.js
----- index.html
----- controllers/
---------- searchController.js
---------- resultsController.js
----- directives/
----- services/
----- views/
---------- searchView.html
---------- resultsView.html
```

Ok let's build our app folder!

We'll start with some controllers and views. https://scotch.io/tutorials/angularjs-best-practices-directory-structure

```
 <!DOCTYPE html>
<html ng-app="searchApp">
<head>
<title>Google Custom Search</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.9/angular.min.js"></
script>
<script src="app.js"></script>
<script src="controllers/searchController.js"></script>
<script src="controllers/resultsController.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/
bootstrap.min.css">
<link rel="stylesheet" href="css/styles.css">
</head>

<body>
<div class="container">
      <div ng-view></div>
</div>
</body>
</html>
```

Get started with index.html

All kinds of ng-tags and script files. Woot.

The angular.module is a global place for creating, registering and retrieving Angular modules. All modules (angular core or 3rd party) that should be available to an application must be registered using this mechanism.

app.js

```
(function(){
      angular.module("searchApp", []);
})();
```

Initialize the ng-app

We’ll use app.js to declare our Angular application *searchController.js*

```
(function(){
	angular.module("searchApp", [])
		.controller("searchController", ['$scope', function($scope){

		}]);
})();
```

resultsController.js

```
(function(){
	angular.module("searchApp", [])
		.controller(“resultsController",['$scope',function($scope){

		}]);
})();
```

Add our ng-controllers

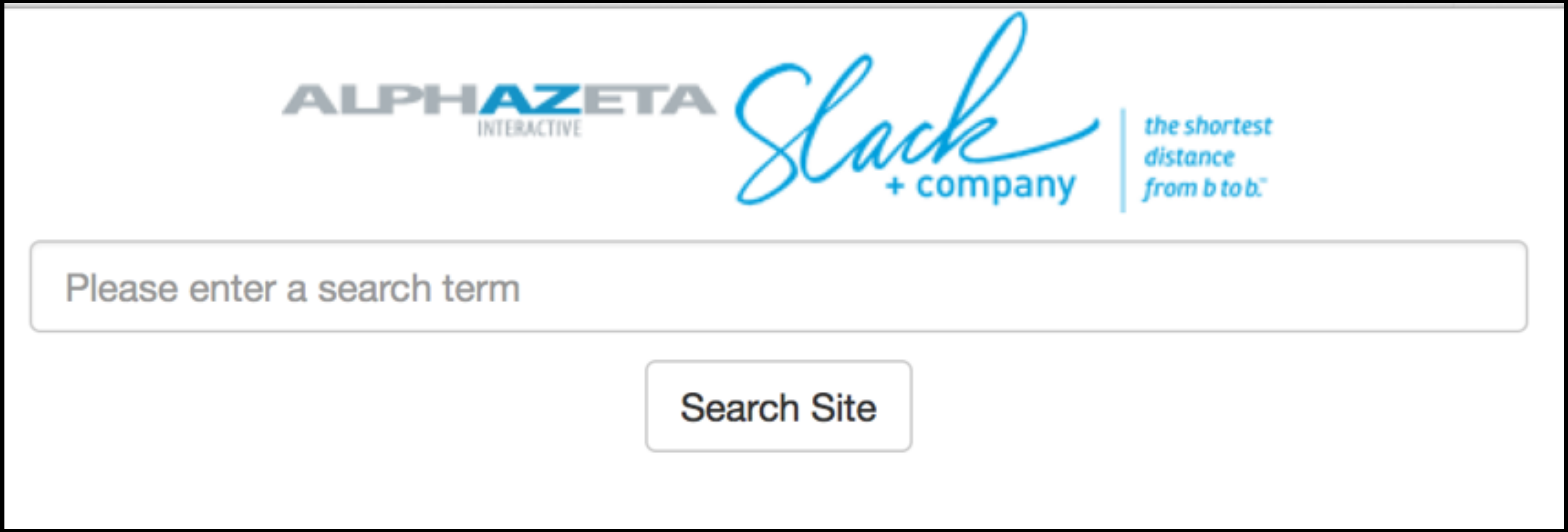
This is the C in MVC. We’re going to need two controllers - one for search, and one for results.

searchView.html

```
<div class="search">
	<p class="text-center">
		<img src="http://www.alphazeta.com/files/0/3/logo-Alphazeta.gif">
		<img src="http://www.slackandcompany.com/portals/_default/skins/Slack-
and-Company2/img/top_logo.png">
	</p>
	<form>
		<fieldset>
			<input type="text" name="searchField" id="search" class="form-
control" placeholder=“Please enter a search term" required>
			<input type="submit" name="submit" id="submit-search" class="btn
btn-default" value="Search Site" />
		</fieldset>
	</form>
</div>
```

Now let’s create our first view

This is the V in MVC. It’s what you see in the browser.


ALPHAZETA INTERACTIVE
Slack + company
the shortest distance from b to b.
Please enter a search term
Search Site

This is the initial search view

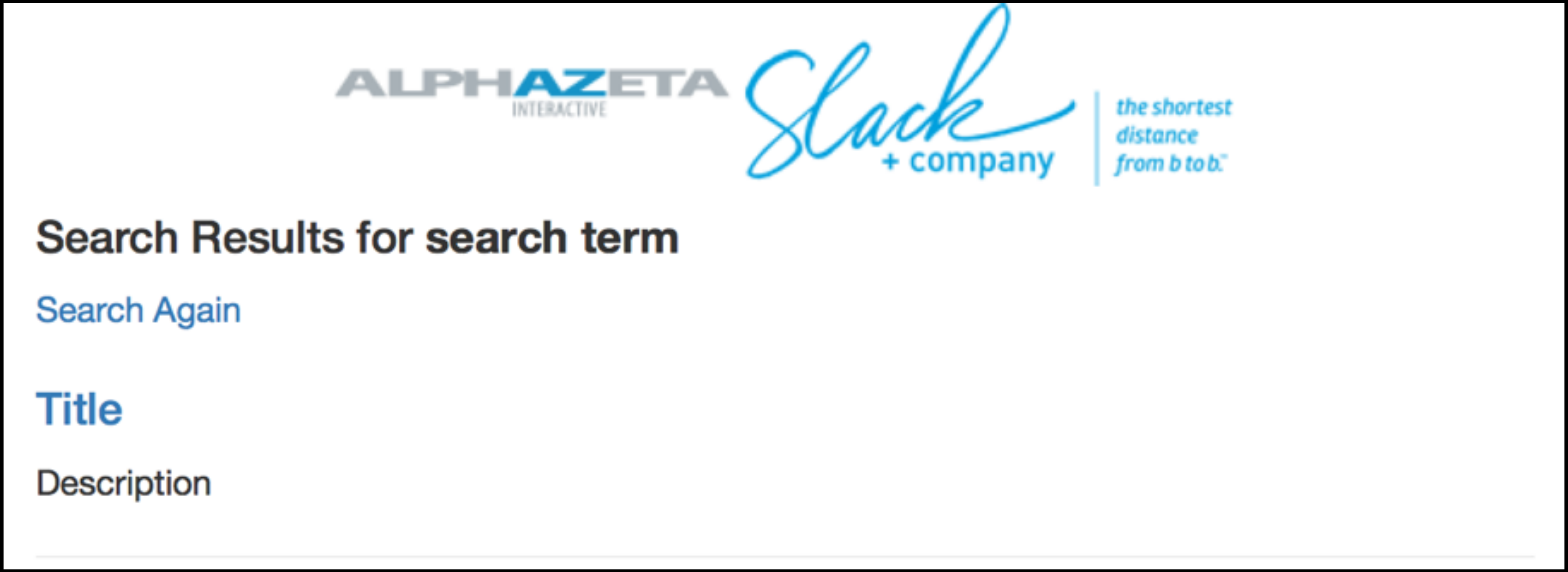
It’s not really doing anything yet

resultsView.html

```
<p class="text-center">
	<img src="http://www.alphazeta.com/files/0/3/logo-Alphazeta.gif">
	<img src="http://www.slackandcompany.com/portals/_default/skins/Slack-
and-Company2/img/top_logo.png">
</p>
<div class="search-results" id="top">
	<h4>Search Results for <strong class=“string">search term</strong></h4>
	<p><a href="#/">Search Again</a></p>
	<div class="row">
		<div class="col-sm-12">
			<h4><a href=“#" target="_blank">Title</a></h4>
			<p>Description</p>
			<hr />
		</div>
	</div>
</div>
```

Now let’s create our second view

This is the results screen with one sample result



Search Results for search term

Search Again

Title

Description

And here's our initial results view

It's also really not doing anything yet, just test data *resultsView.html*

```
<p class="text-center">
	<img src="http://www.alphazeta.com/files/0/3/logo-Alphazeta.gif">
	<img src="http://www.slackandcompany.com/portals/_default/skins/Slack-and-
Company2/img/top_logo.png">
</p>
<div class="search-results" id="top">
	<h4>Search Results for <strong>search term</strong></h4>
	<p><a href="#/">Search Again</a></p>
	<div class="row">
		<div class="col-sm-12">
			<h4><a href="#" target="_blank">Title</a></h4>
			<p>Description</p>
			<hr />
		</div>
	</div>
	<ul class="pagination">
		<li><a href="#">Previous</a></li>
		<li><a href="#">Next</a></li>
	</ul>
</div>
```

Now let’s create our second view

This is a sample result with sample paging *index.html*

```
 <!DOCTYPE html>
<html ng-app="searchApp">
<head>
<title>Google Custom Search</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.9/
angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.9/angular-
route.min.js"></script>
<script src="app.js"></script>
<script src="controllers/searchController.js"></script>
<script src="controllers/resultsController.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/
bootstrap.min.css">
</head>


.....
```

To finish our setup, let's add in some routing - so we know where we're going!

app.js

```
(function(){
	angular.module("searchApp", ['ngRoute'])
	.config(['$routeProvider', function($routeProvider) {
	  $routeProvider.when('/search',{
	    templateUrl: "views/searchView.html",
	       controller: "searchController"
	  }).when('/search/:term',{
	       templateUrl: "views/resultsView.html",
	       controller: "resultsController"
	  }).otherwise({
	       redirectTo: "/search",
	  });
	 }]);
})();
```

I like adding routes right into app.js

Our app will load at this point. It will be very static.

And now for some live coding!

Get the final code at:

https://github.com/timpalac/angularcustomsearch