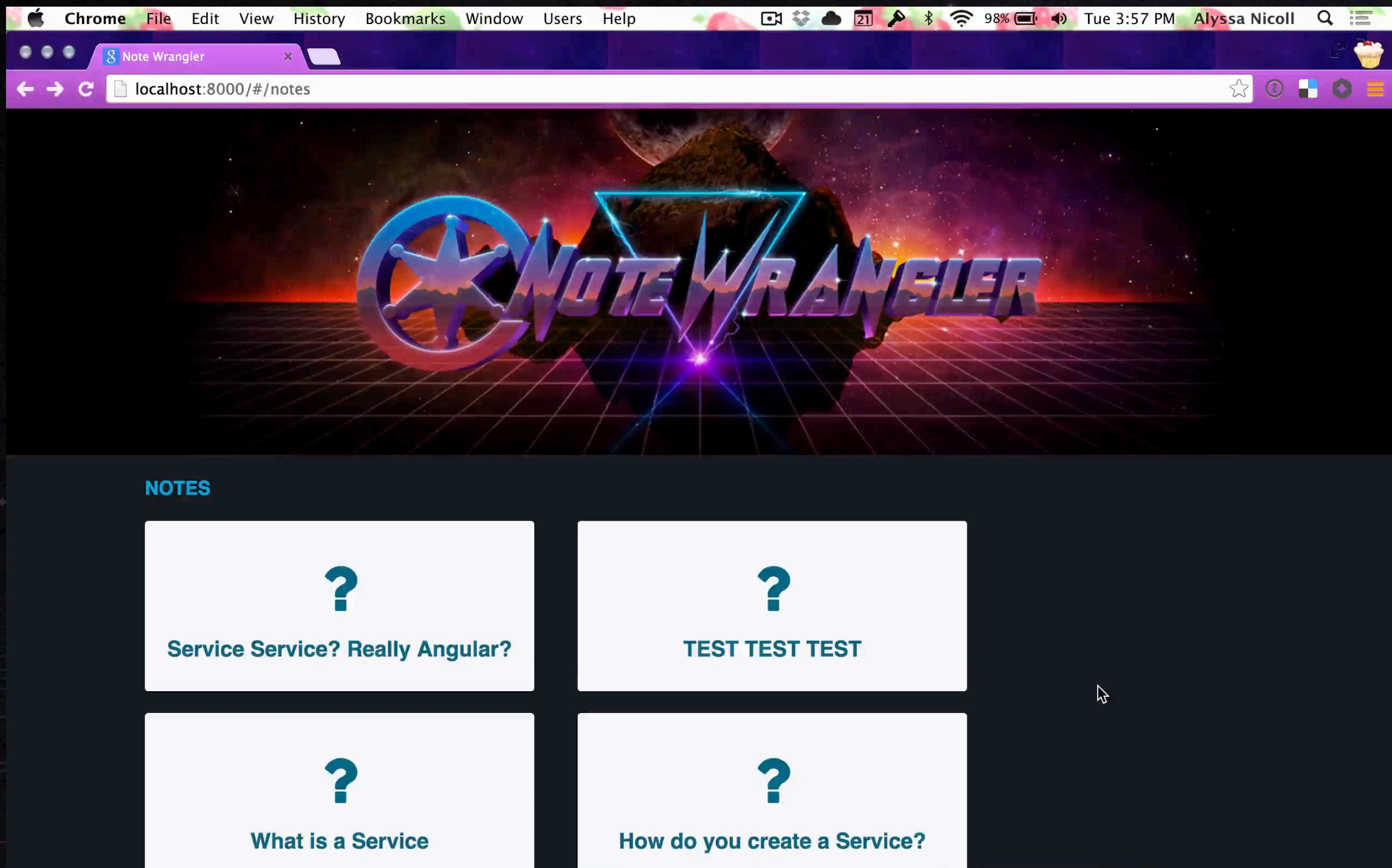


STARRYING
SHARP with
Angular.js

Routes

Wiring Together Views Section 1

Angular Note Wrangler



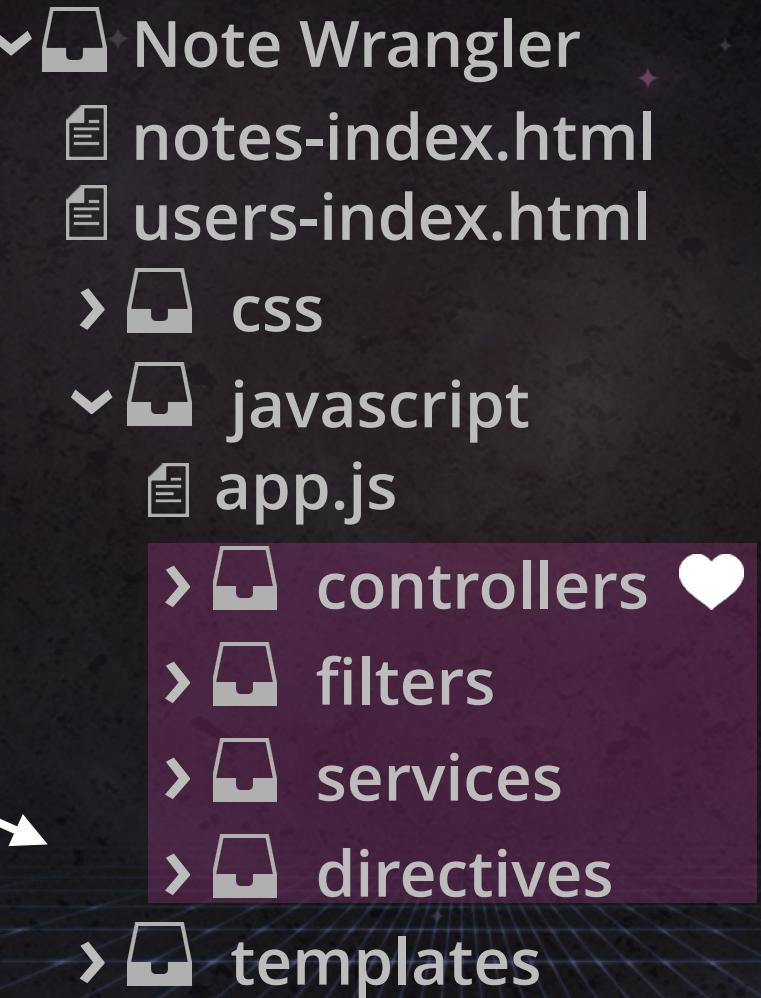
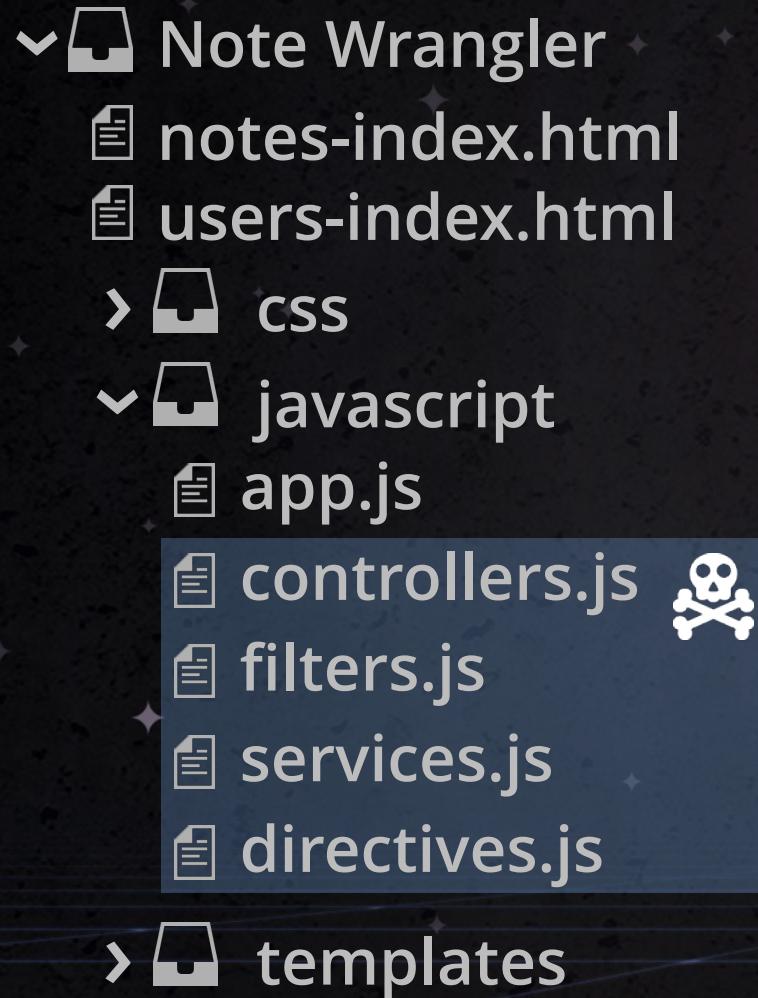
Starting With Static Pages

This was handed to us from our designer. All the pages are static.

- ✓  Note Wrangler
 - 📄 notes-index.html
 - 📄 users-index.html
 - ✓  css
 - 📄 application.css
 - >  images

Organizing: Separate Directories for All

Separate controllers, services, filters, and directives into individual directories.



Organizing: Separate Files for All

Inside the controllers directory we will have a file for each controller.

- ✓  Note Wrangler
 - 📄 notes-index.html
 - 📄 users-index.html
 -  css
 - ✓  javascript
 - 📄 app.js
 - ✓  controllers
 - 📄 notes-create-controller.js
 - 📄 notes-edit-controller.js
 - 📄 notes-index-controller.js
 - 📄 notes-show-controller.js
 -  filters
 -  services
 -  directives
 -  templates

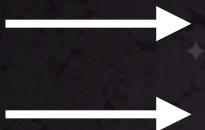
Keep things
encapsulated and bite-size

Back to Our Designer's Static Files

Both of these static files have boilerplate code that is exactly the same.

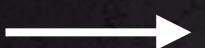
- ✓  Note Wrangler

-  notes-index.html



Both are fully fleshed out files with much redundancy <html>...</html>

-  users-index.html



- >  css

- >  javascript

- >  templates

Duplicated code in both users-index.html and notes-index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="NoteWrangler">

  <head>
    <meta charset="utf-8">
    <title>Note Wrangler</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/application.css" />
  </head>

  <body>
    <div class="nav-wrapper has-dropdown">
      <div class="nav-content">
        <div class="wrapper">
          <div class="nav-content-layout">
            <div class="nav-list">

              <a href="#" class="list-item" link="#/users">
```

```
<div class="main-wrapper">
  <div class="note-wrapper">
    <div class="note-content">
      <div class="notes-header">
        <h1 title="Notes">Notes</h1>
      </div>

      <div class="note-wrapper">
        <a class="card-notes" ng-repeat="note in notes">

          <div class="card" title="{{note.title}}>
            <h2 class="h3">{{note.title}}</h2>
            <p>{{note.description}}</p>
          </div>

        </a>
      </div>

    </div>
  </div>
</div>
```

unique chunk inside
the main-wrapper div

```
<div class="main-wrapper">  
  
  <div class="users-wrapper">  
    <h1>Users</h1>  
  
    <div class="users-wrapper">  
      <a class="card-users" ng-repeat="user in users">  
        <div class="card" title="{{user.name}}>  
          <h2 class="h3">{{user.name}}</h2>  
          <p>{{user.bio}}</p>  
        </div>  
      </a>  
    </div>  
  
  </div>  
  
</div>  
  
<!-- Load Js libs -->  
<script src=".js/vendor/jquery.js"></script>  
<script src=".js/vendor/angular.js"></script>  
<script src=".js/vendor/angular-route.js"></script>
```

Unique chunk inside
the main-wrapper div

Unique HTML Gets Its Own Template File

notes-index.html

```
<div class="note-wr
  <div class="note-
    <div class="not
      <h1 title="No
    </div>

    <div class="not
      <a class="car
        <div class=
          <h2 class
            <p>{{note
        </div>

      </a>
    </div>
```

users-index.html

```
<div class="users-wrapp
  <h1>Users</h1>

  <div class="users-wrapp
    <a class="card-users" ng-repeat="user in users">
      <div class="card" title="{{user.name}}>
        <h2 class="h3">{{user.name}}</h2>
        <p>{{user.bio}}</p>
      </div>
    </a>
  </div>
</div>
```

Moving Templates Into Their Own Folder

We will place our static files into our templates/pages directory.

✓ Note Wrangler

```
  ⏎ index.html  
  > ⏎ css  
  ✓ ⏎ templates  
    ✓ ⏎ pages  
      ✓ ⏎ notes  
        ⏎ notes-index.html  
      ✓ ⏎ users  
        ⏎ users-index.html
```

```
  ⏎ notes-index.html  
    <div class="note-wrapper"> ...  
  
  ⏎ users-index.html  
    <div class="users-wrapper"> ...
```

We can eliminate the duplication and remove the prefix on our files.

Moving Templates Into Their Own Folder

We will place our static files into our templates/pages directory.

✓ Note Wrangler

```
  ⏺ index.html  
  > ⏺ css  
  ✓ ⏺ templates  
    ✓ ⏺ pages  
      ✓ ⏺ notes  
        ⏺ index.html  
      ✓ ⏺ users  
        ⏺ index.html
```

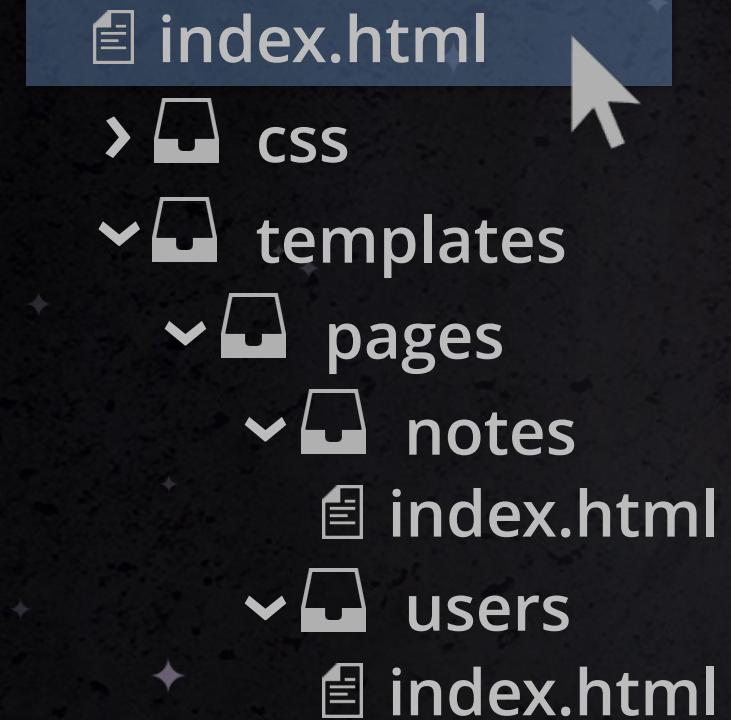
```
notes-index.html  
<div class="note-wrapper"> ...  
  
users-index.html  
<div class="users-wrapper"> ...
```

We can eliminate the duplication and remove the prefix on our files.

Creating the Main Index File

This is the first file that will load up our Angular app.

✓ Note Wrangler



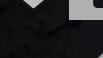
Unique content removed —

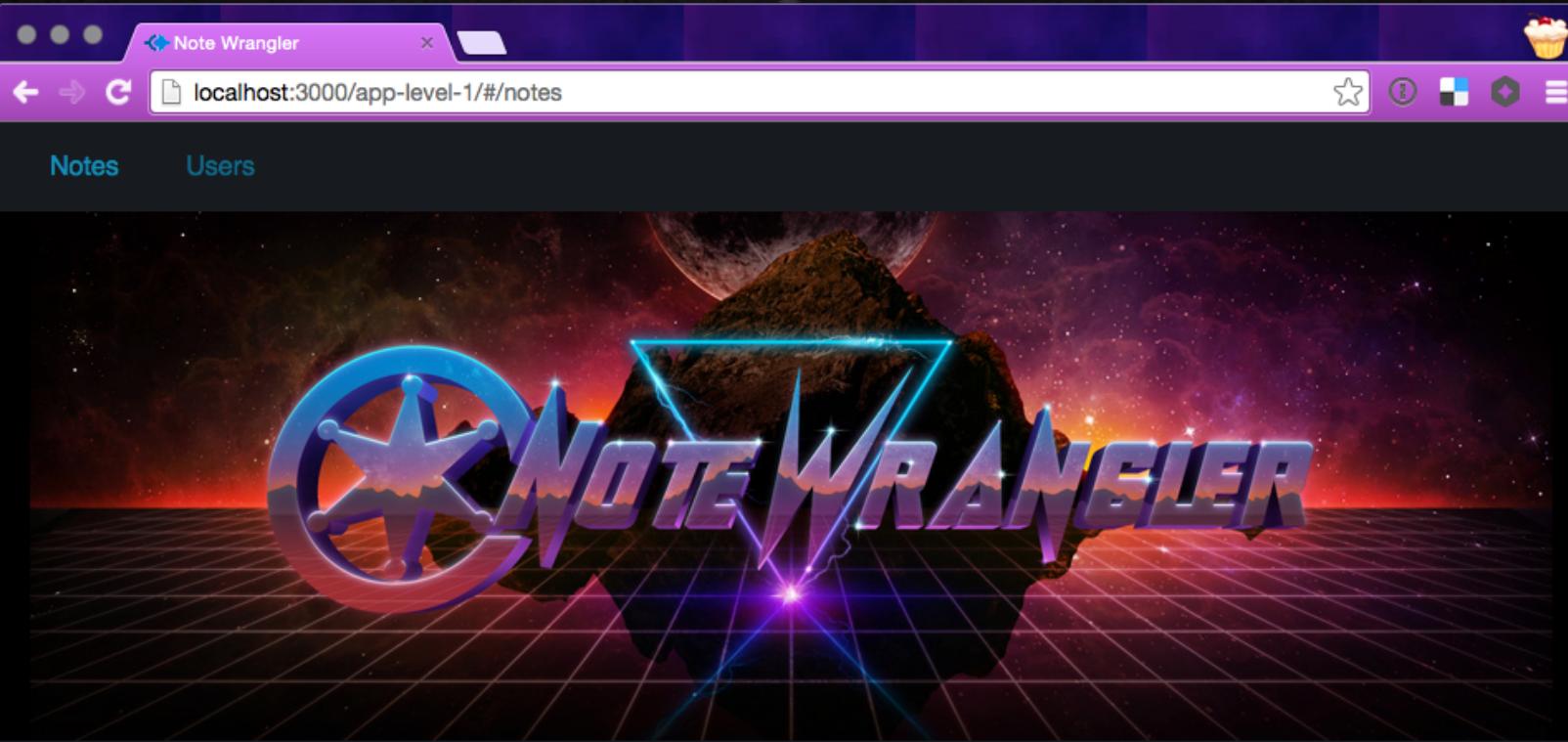
Add ng-app

```
<html lang="en" ng-app="NoteWrangler">
  <head>...</head>
  <body>
    <div class="nav-list">
      <a href="#/notes"> Notes </a>
      <a href="#/users"> Users </a>
    </div>

    <div class="hero-wrapper"> ... </div>
    <div class="main-wrapper">
      ...
    </div>
    <script src="/js/vendor/jquery.js"></script>
```

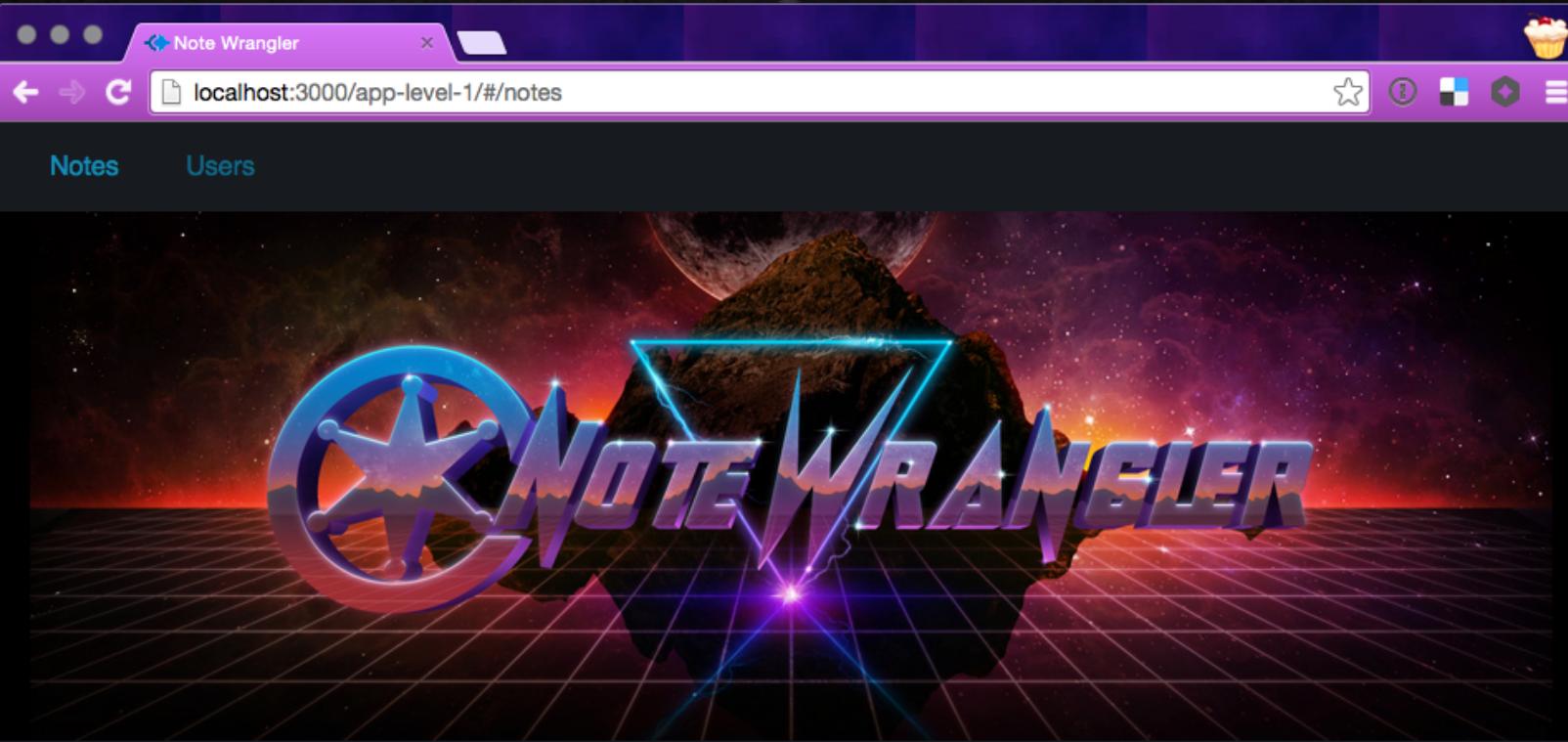
Current State of Our App

- ✓  Note Wrangler
 - 📄 index.html
 - >  css
 - ✓  templates
 - ✓  pages
 - ✓  notes
 - 📄 index.html
 - ✓  users
 - 📄 index.html
 - >  javascript



Current State of Our App

- ✓  Note Wrangler
 - 📄 index.html
 - >  css
 - ✓  templates
 - ✓  pages
 - ✓  notes
 - 📄 index.html
 - ✓  users
 - 📄 index.html
 - >  javascript

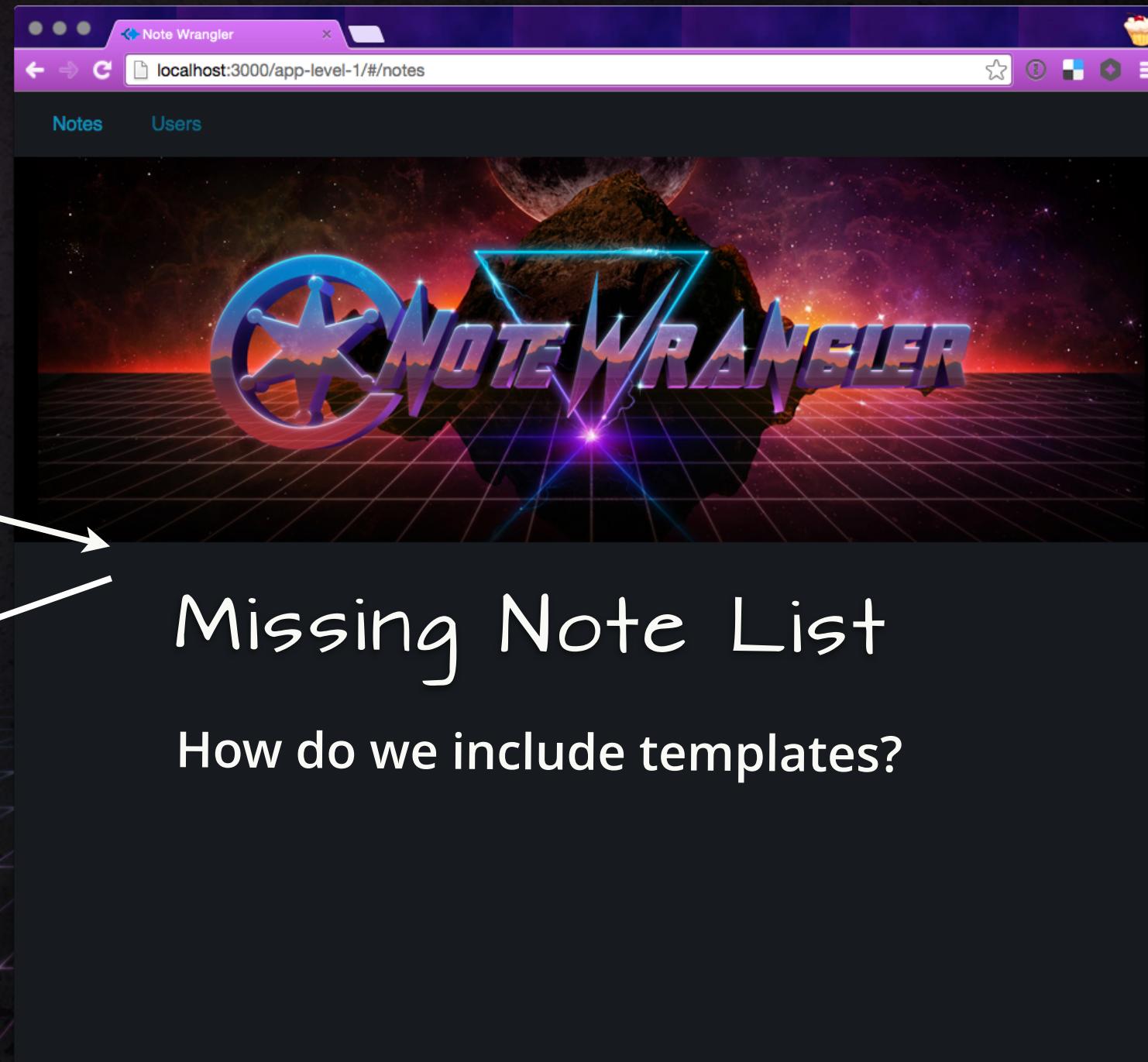




The Notes URL Should Load Notes

If the route is `http://example.com/#/notes`, we want `index.html` to load the `notes/index.html`.

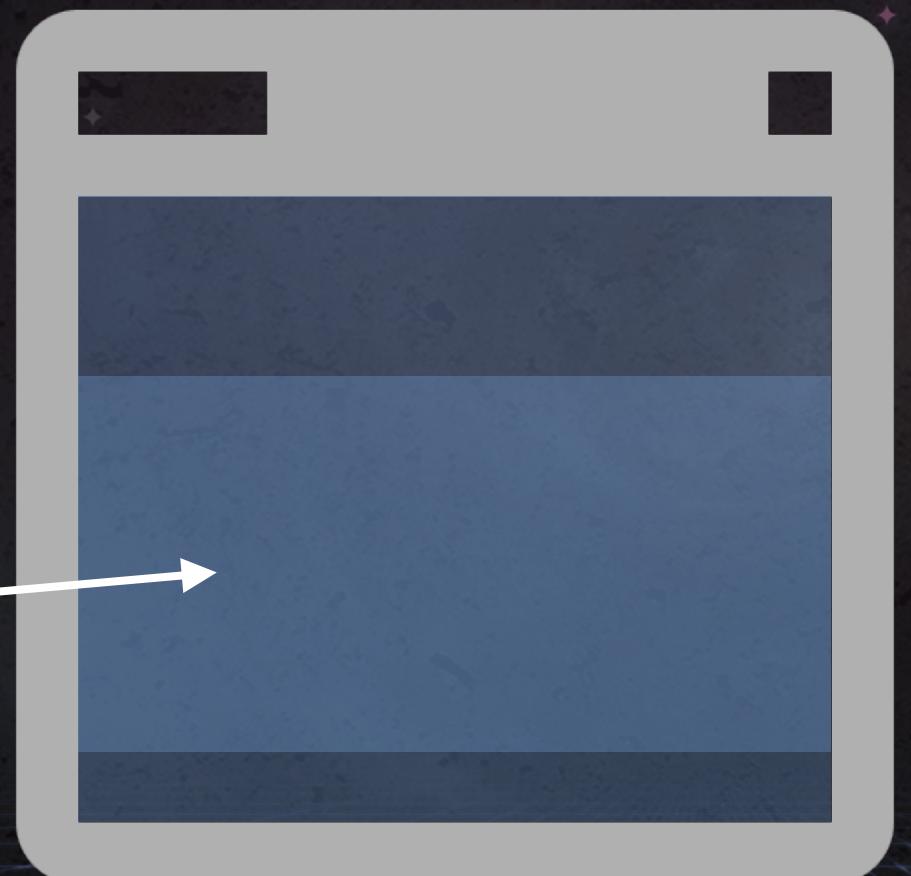
```
✓ └── Note Wrangler
    └── index.html
        >   └── css
    ✓ └── templates
        ✓ └── pages
            ✓ └── notes
                └── index.html
            ✓ └── users
                └── index.html
        >   └── javascript
```



Defining a Route

Angular routes allow us to map URLs to use templates so that every time the current route changes, the included view changes with it.

- ✓  Note Wrangler
 - 📄 index.html
 - >  css
 - ✓  templates
 - ✓  pages
 - ✓  notes
 - 📄 index.html
 - ✓  users
 - 📄 index.html
 - ✓  javascript
 - 📄 routes.js



routes.js fetches and loads notes/index.html

Four Steps to Route Happiness

 Using ngView

 Loading the ngRoute library

 Importing ngRoute module

 Defining routes



Using ngView

This allows us to tell our Angular app *where* to load in templates that we will wire together shortly.

index.html

```
<html lang="en" ng-app="NoteWrangler">
  <head>...</head>
  <body>
    <div class="nav-list">
      <a href="#/notes"> Notes </a>
      <a href="#/users"> Users </a>
    </div>

    <div class="hero-wrapper"> ... </div>
    <div class="main-wrapper">
      <div ng-view></div>
    </div>
    <script src="/js/vendor/jquery.js"></script>
```

notes/index.html

users/index.html

Loading the ngRoute Library

In order to slim the Angular core down, they put routing in a separate module. So you need to explicitly import ngRoute in your application.

 index.html

```
<div class="hero-wrapper"> ... </div>
<div class="main-wrapper">
  <div ng-view></div>
</div>

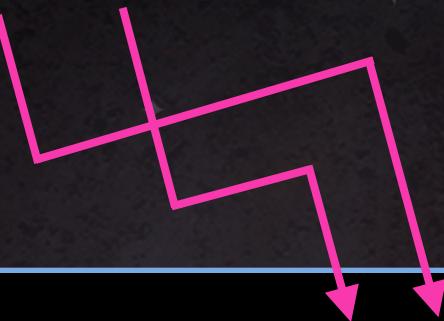
<script src="/js/vendor/jquery.js"></script>
<script src="/js/vendor/angular-route.js"></script>
```

Including ngRoute in Our App

We need to include ngRoute in our main application module so our whole app will have access to this service.

app.js

```
angular.module("NoteWrangler", ['ngRoute'])
```



Now we can create a routes.js file!

routes.js

```
angular.module('NoteWrangler')
.config(function($routeProvider){});
```

We'll use \$routeProvider in a minute to specify Routes.

Creating routes.js

Why aren't we doing it the way we learned in Shaping up with Angular.js?

 `var app = angular.module('store-products', []);
app.directive('productTitle', function(){ . . . });`

Setting your app module to a variable and reusing that variable is bad practice.

 routes.js

```
angular.module('NoteWrangler')  
.config(function($routeProvider){});
```

Re-declare your application module in *every new file*

Defining Routes With \$routeProvider

Inside module.config we can use one of \$routeProvider's methods to define routes.

`.when(path, route);`

Adds a new route definition to the \$route service.

`.otherwise(params);`

Sets route definition that will be used on route change when no other route definition is matched.

Declaring Our First Route

Let's go ahead and define a route for /notes to load in the notes/index.html using \$routeProvider's `.when()` method.

 routes.js

```
angular.module('NoteWrangler')
.config(function($routeProvider) {
  $routeProvider.when('/notes', {
    templateUrl: '/templates/pages/notes/index.html'
  })
});
```

Notes Route Now Loads In

The screenshot shows a web browser window titled "Note Wrangler". The address bar displays "localhost:3000/app-level-1/#/notes". The main content area features a large, stylized "Note Wrangler" logo with a star and a grid background. Below the logo, there are four cards:

- NOTES**
- Service Service? Really Angular?**
Clarify the confusion between Service the term and `service` the angular method and to explain the 5 different Service recipes in Angular.
- Define Service**
Angular services are objects that are used to organize and share code across your app."
- How do you Create a Service?**
Steps for Creating a Service
- ngModel BP**
NgModel Best Practice

STAYING SHARP with Angular.js

Adding the Users Route

 routes.js

```
angular.module('NoteWrangler')
.config(function($routeProvider) {
  $routeProvider.when('/notes', {
    templateUrl: 'templates/pages/notes/index.html',
  })
  .when('/users', {
    templateUrl: 'templates/pages/users/index.html',
  })
});
```

Notice the method chaining

Users Route Now Loads In

The screenshot shows a web browser window with a dark theme. The title bar reads "Note Wrangler". The address bar displays "localhost:3000/app-level-1/#/users". The main content area features a large, stylized logo for "Note Wrangler" with a blue circular icon containing a person silhouette and a mountain. Below the logo, the word "Note Wrangler" is written in a large, metallic, reflective font with a purple-to-blue gradient. The background of the page has a futuristic, space-themed aesthetic with a grid floor and a planet in the distance. A section titled "USERS" is present, containing four individual user cards:

- Jeffrey Zeldman**
Founder, Happy Cog studios. Author, Designing With Web Standards. Publisher, A List Apart, A Book Apart.
- Brad Green**
I work at Google where I manage AngularJS and Google's internal sales productivity applications.
I'm a dad.
- Eric A. Meyer**
Web standards | (X)HTML | CSS | microformats | community | writing | speaking | signing man.
- Gregg Pollack**
Founder of Envy Labs, Code School, Orlando Ruby Users Group, BarCamp Orlando, and the Orlando Tech Events newsletter.

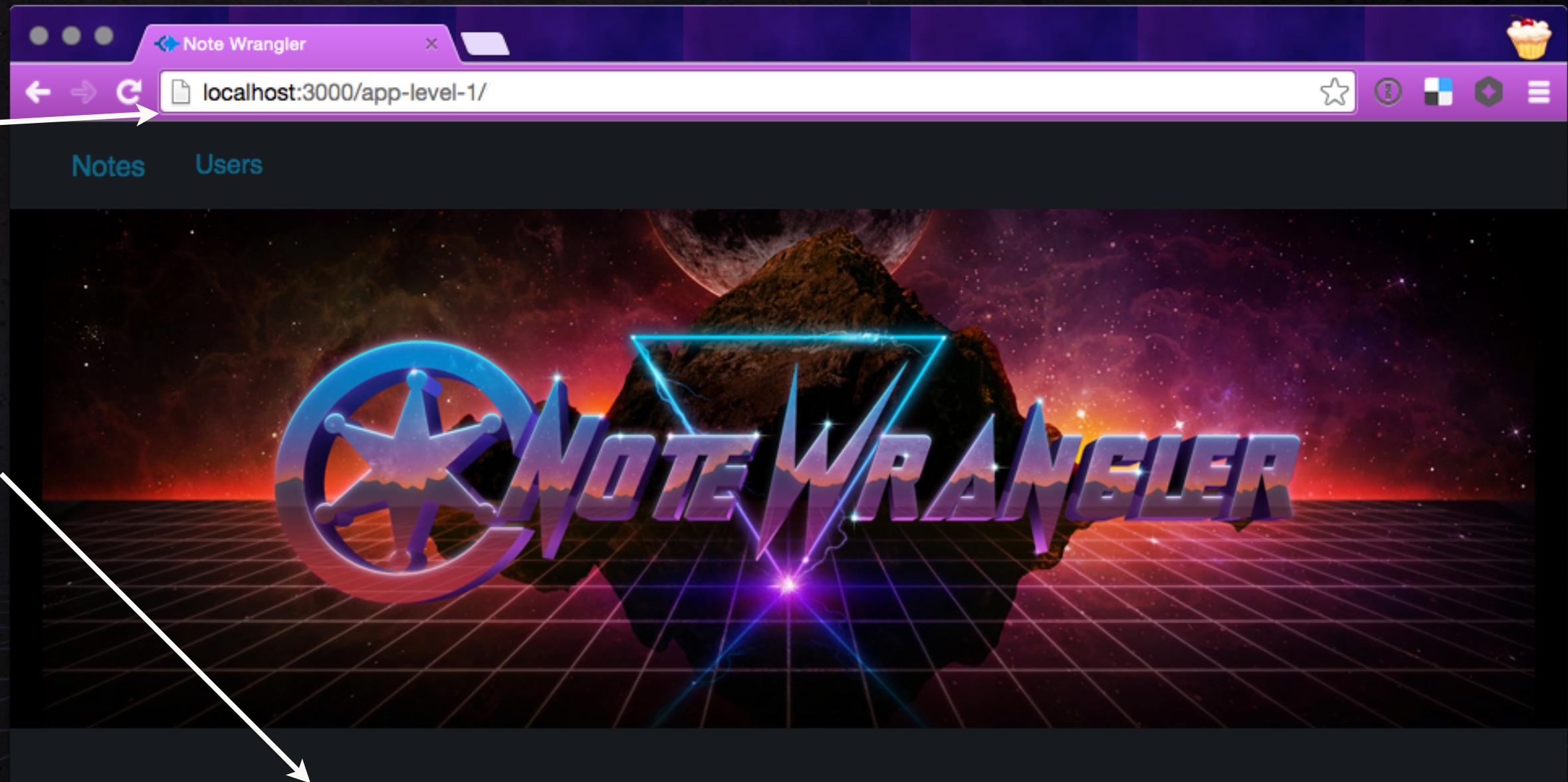
users/index.html

```
<div class="users-wrapper"> ...
```

Defining the Root Route

If the route is `http://example.com/` though, we want index to load in the `notes/index.html` by default.

- ✓  Note Wrangler
- index.html
- >  css
- ✓  templates
- ✓  pages
- ✓  notes
- ✓  index.html
- ✓  users
- ✓  index.html



Missing Notes List!

🔗 Options for Loading In Notes for Root Route

routes.js

```
angular.module('NoteWrangler')
.config(function($routeProvider) {
  $routeProvider.when('/notes', {
    templateUrl: 'templates/pages/notes/index.html',
  })
  ...
  .when('/', {
    templateUrl: 'templates/pages/notes/index.html',
  })
});
```



Setting a Catch-all Route

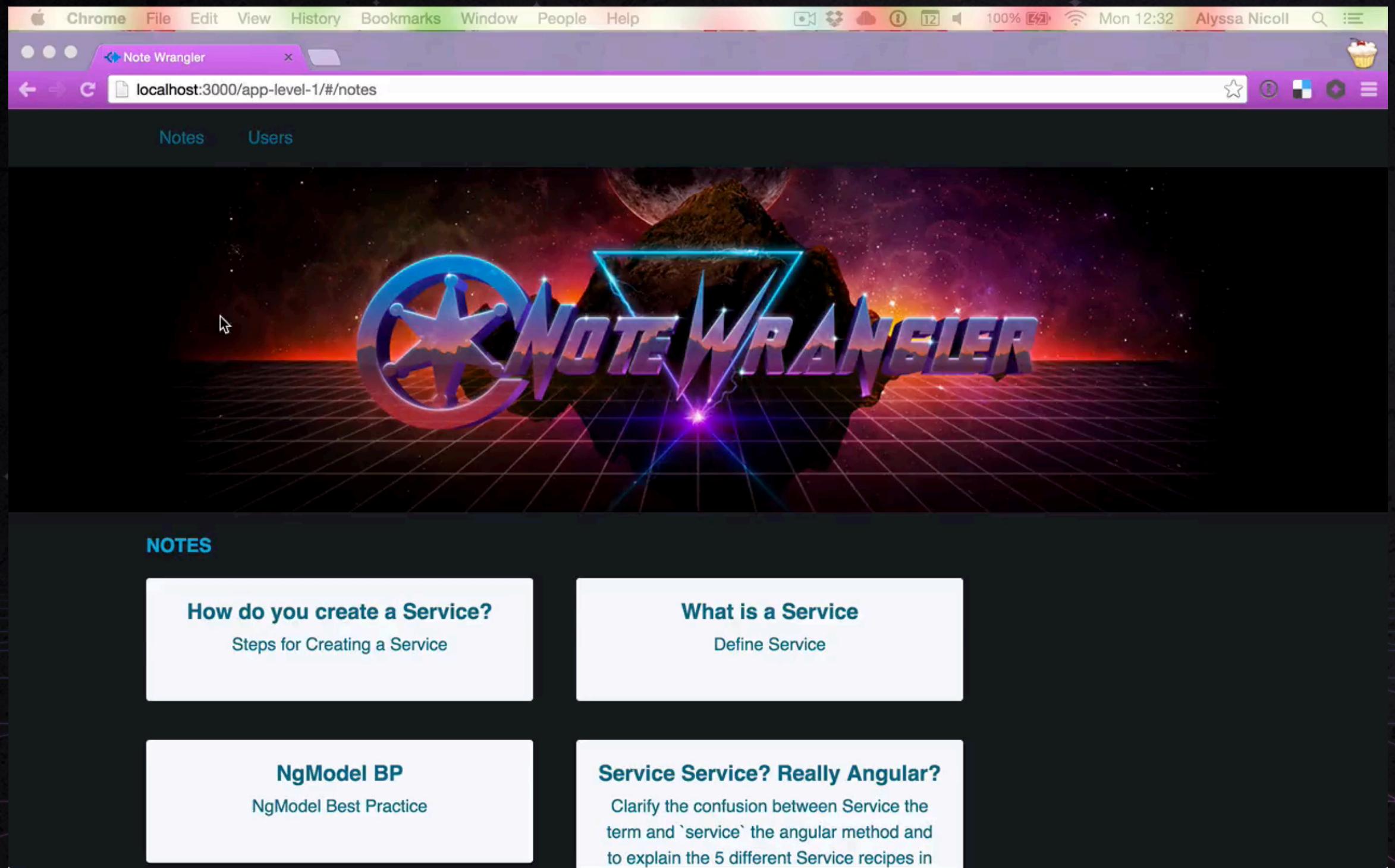
routes.js

```
angular.module('NoteWrangler')
.config(function($routeProvider) {
  $routeProvider.when('/notes', {
    templateUrl: 'templates/pages/notes/index.html',
  })
  ...
  .when('/', {
    templateUrl: 'templates/pages/notes/index.html',
  })
  .otherwise({ redirectTo: '/' });
});
```

.otherwise(params);

Sets route definition that will be used on route change when no other route definition is matched.

Templates Are Now Wired Up



Serving Up Our Application

