STAYING SHARP with Angular.JS

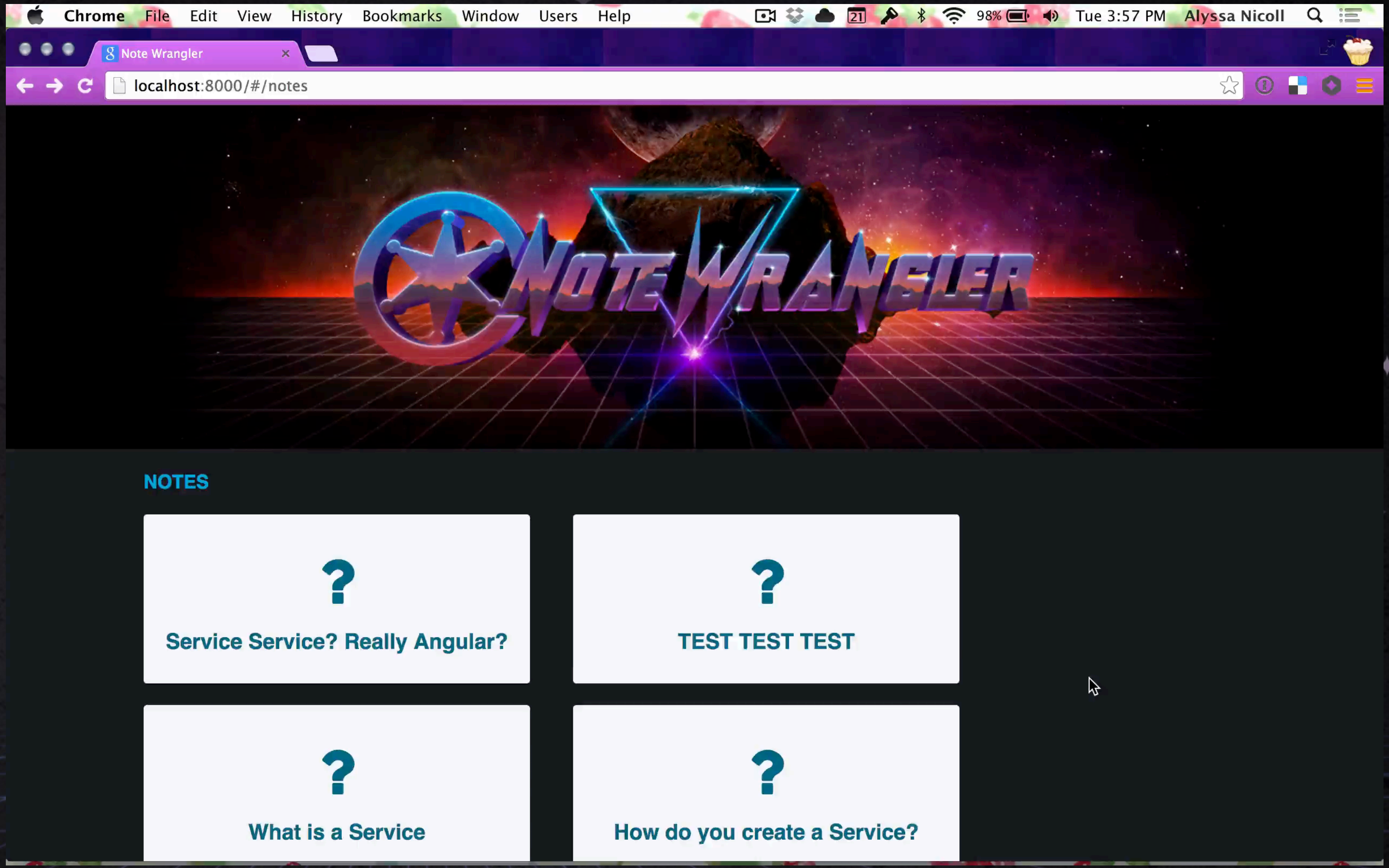
Services Level 3

5 Recipes and a Factory Section 1

STAYING SHARP with Angular.JS

Current State of Our App

- Note Wrangler
  - index.html
  - css
  - js
    - app.js
    - controllers
      - app.js
    - filters
    - directives
  - templates


Chrome File Edit View History Bookmarks Window Users Help
Tue 3:57 PM Alyssa Nicoll
Note Wrangler
localhost:8000/#/notes
Note Wrangler

NOTES

?
Service Service? Really Angular?

?
TEST TEST TEST

?
What is a Service

?
How do you create a Service?

Controllers Currently Have AJAX $http Calls

notes-edit-controller.js

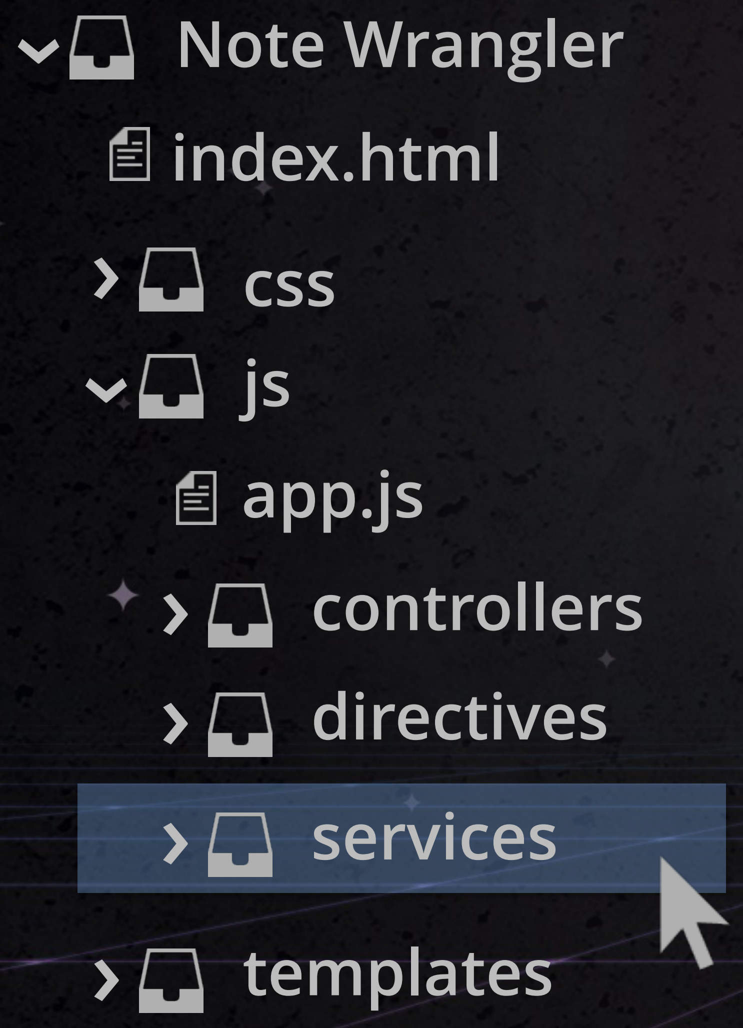
```
angular.module("NoteWrangler")
.controller("NotesEditController", function($scope, $http) {
   $scope.updateNote = function(noteObj) {
      $http({method: "PUT", url: "/notes", data: noteObj})
      ...
```

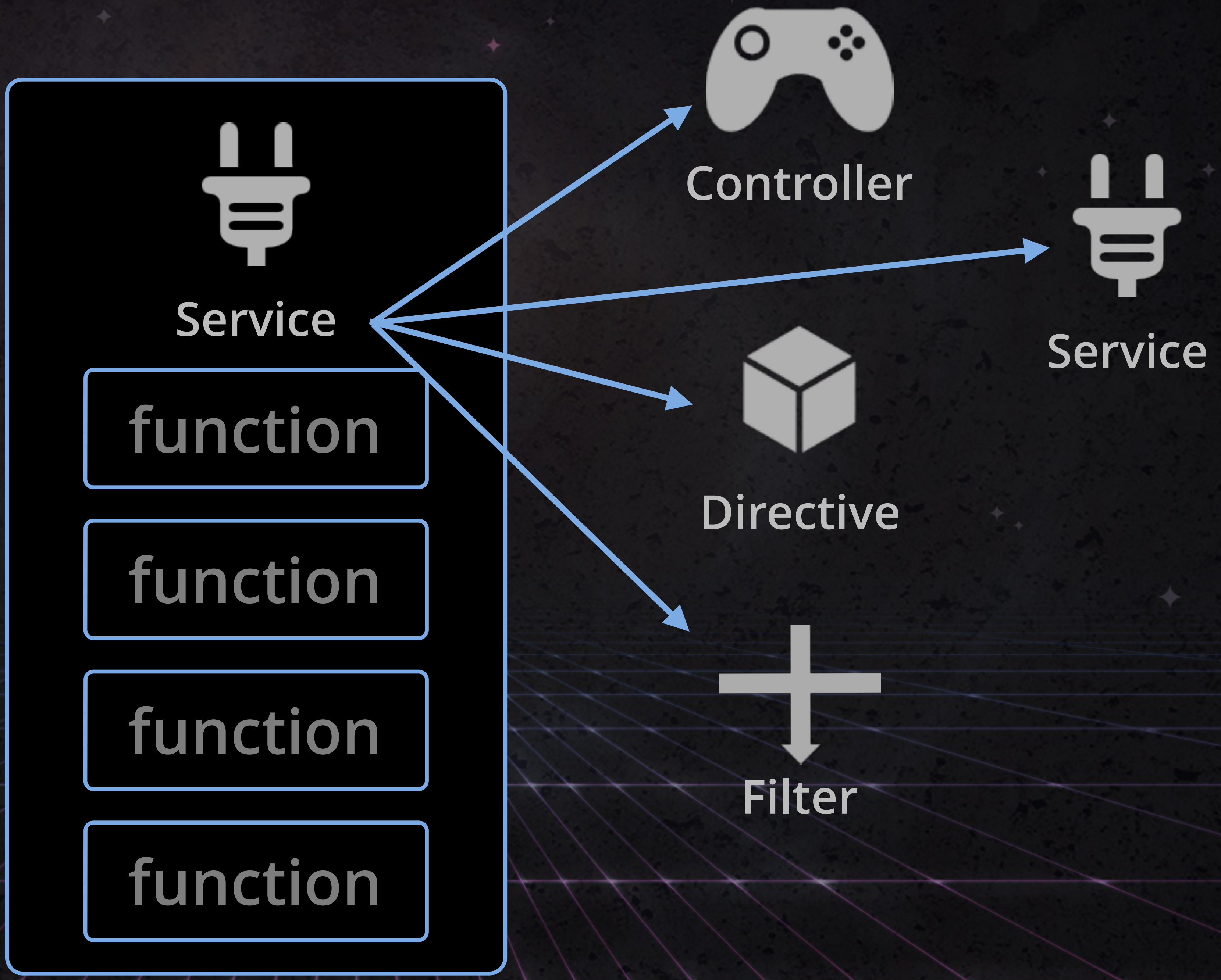
This code is fine, but it's not reusable across other parts of our application. It is also going to be harder to test.

Where should this code be then?

A Service for Your Data

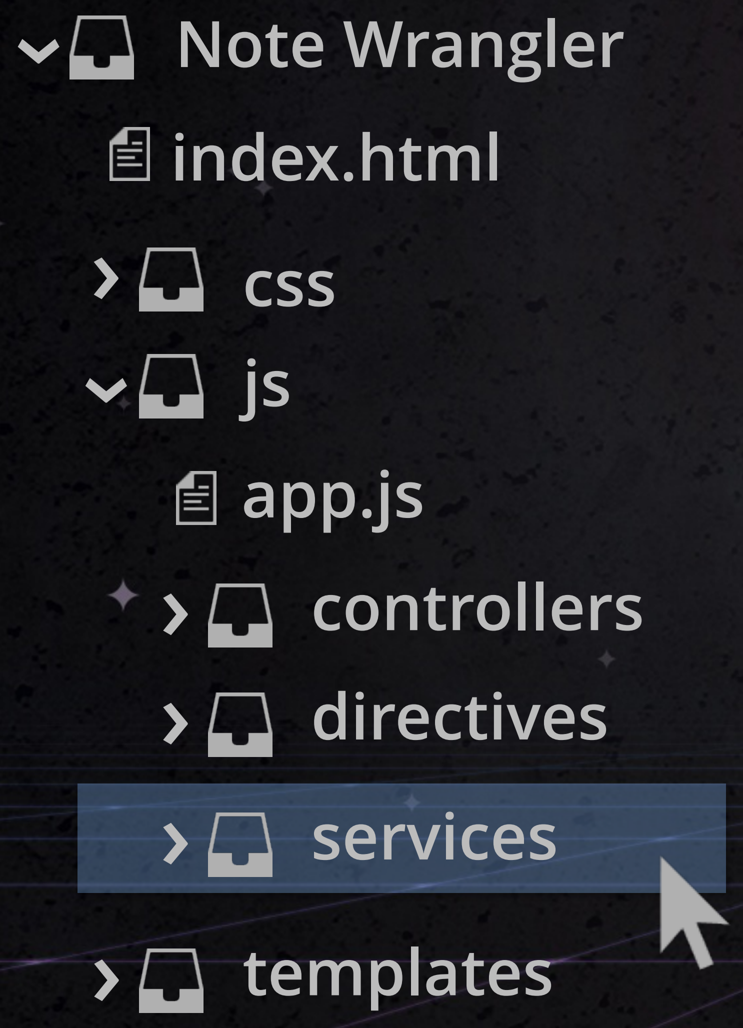
Services should hold functions responsible for connecting and fetching data, and then sharing it across our application.

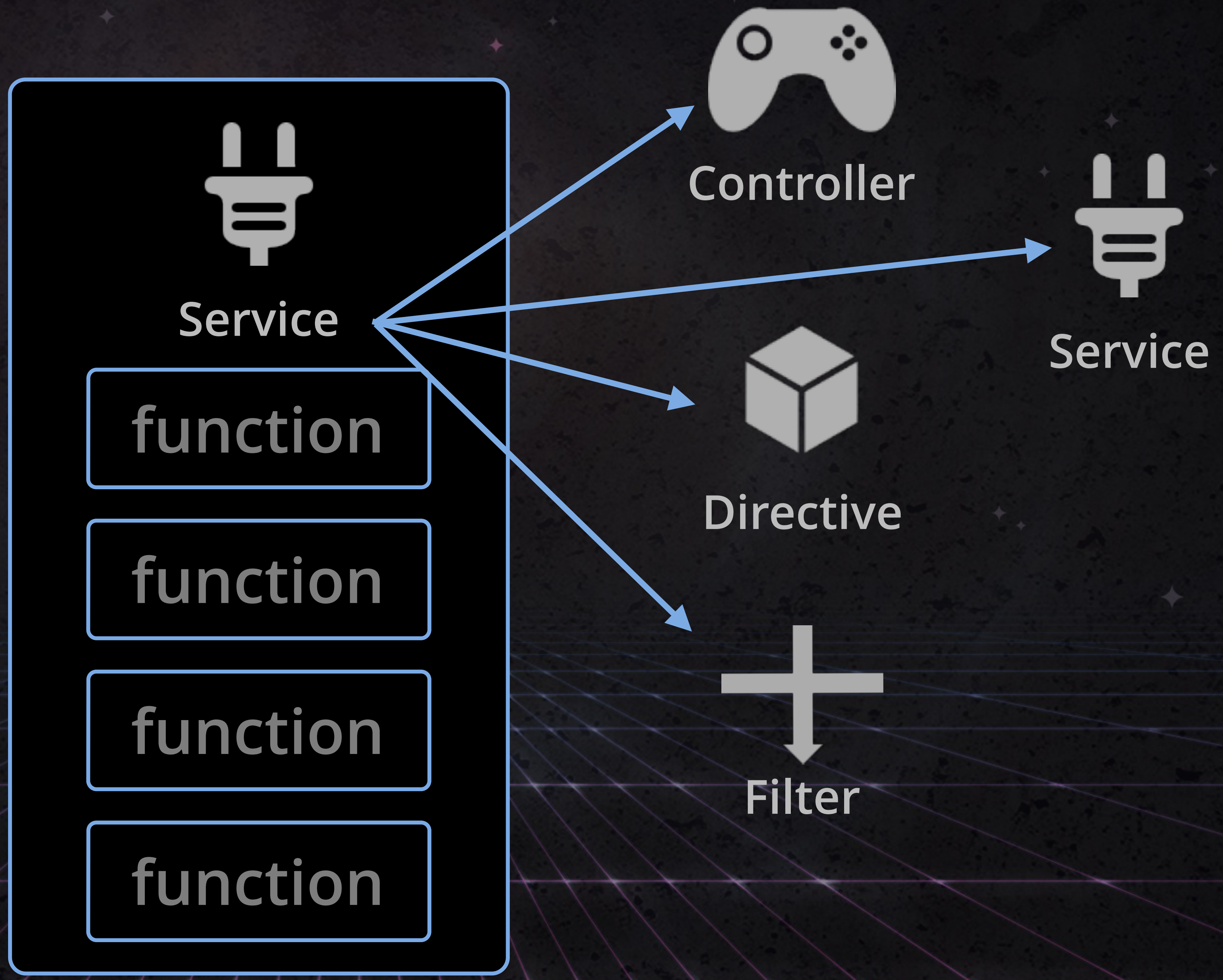




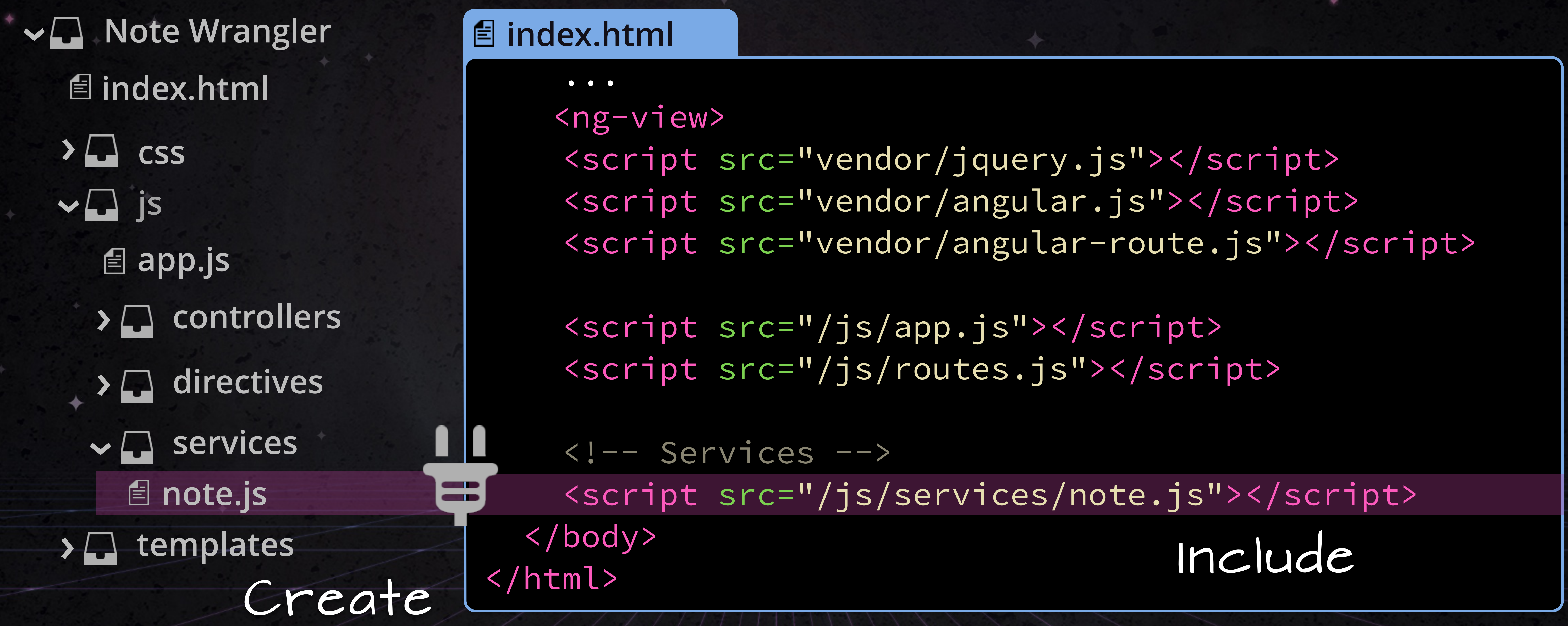
A Service for Your Data

Services should hold functions responsible for connecting and fetching data, and then sharing it across our application.





Creating and Including the Service File



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

     <script src="/js/app.js"></script>
     <script src="/js/routes.js"></script>

     <!-- Services -->
     <script src="/js/services/note.js"></script>
  </body>
</html>
```

5 Service Recipes

We need to choose a recipe. There are 5 total recipes that range in complexity and customization. Factory and Provider are the two most commonly used for creating a Service.

- Value
- Factory
- Service
- Provider
- Constant

2 most common

2 Most Used Service Recipes

- Factory Most commonly used

  Used for sharing functions and objects across an application.

- Provider Commonly used

  Used for sharing methods and objects (like a Factory), but allows for configuration.

Creating a Service With the Factory Recipe

Use the Factory recipe to register the service with our app module.

services/note.js

```
angular.module("NoteWrangler")
.factory("Note", function NoteFactory() {
  ...
});
```

Remember the naming convention <name> + <recipe>

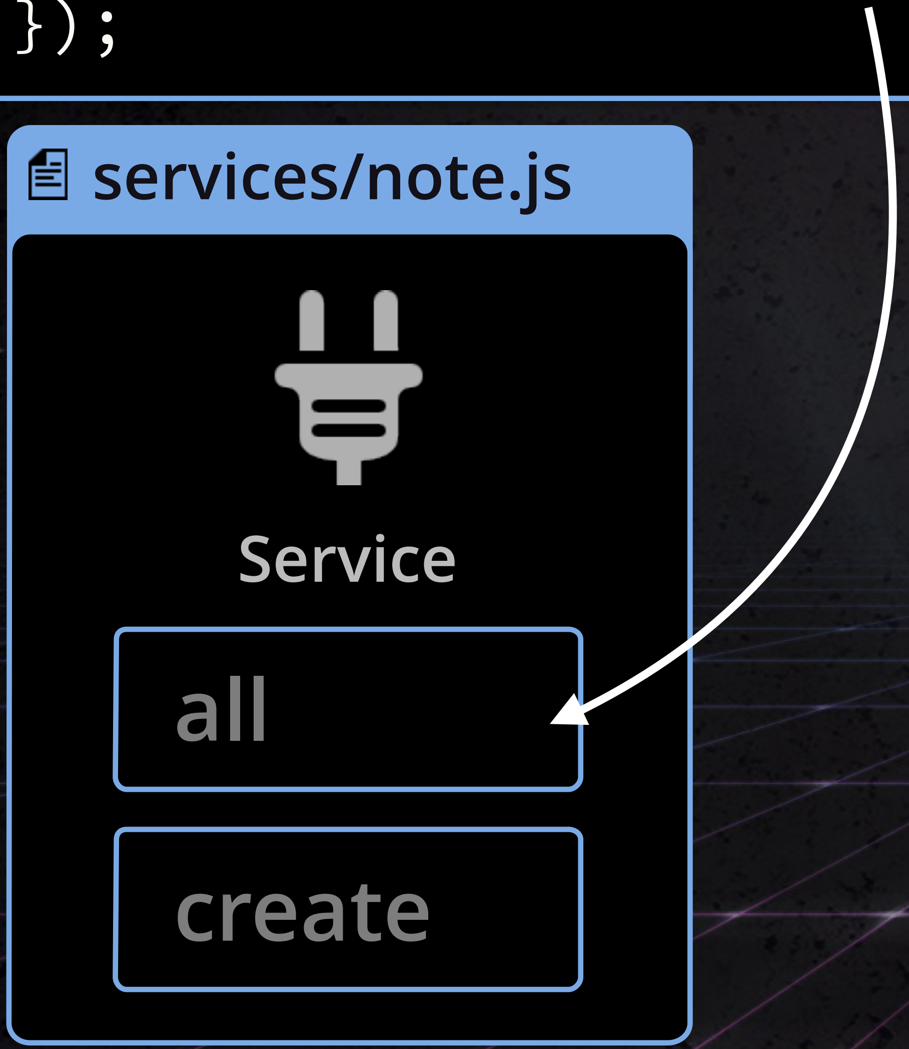
Factory recipe

```
angular.module("<ModuleName>")
.factory("<ServiceName>", function <ServiceName>Factory() {
  return { <object containing shared functions> }
});
```

Populating Our Factory Service

notes-index-controller.js

```
angular.module("NoteWrangler")
.controller("NotesIndexController", function($scope, $http) {
  $scope.notes =  $http({method: "GET", url: "/notes"}); )};
});
```



Inside our factory service we'll define reusable methods that make our $http calls.

Populating Our Factory Service

The all method gets a list of all the notes, and create posts a new note.

services/note.js

```
angular.module("NoteWrangler")
.factory("Note", function NoteFactory() {
  return {
    all: function() {
     return $http({method: "GET", url: "/notes"}); )};
    )},
    create: function(note) {
     return $http({method: "POST", url: "/notes", data: note});
    )}
  }
});
```

Calling Our Factory Service

services/note.js

```
angular.module("NoteWrangler")
.factory("Note", function NoteFactory() {
  return {
    all: function() { return $http({method: "GET", url: "/notes"}); )},
    ...
```

notes-index-controller.js

```
angular.module("NoteWrangler")
.controller("NotesIndexController", function($scope) {
  $http({method: "GET", url: "/notes"})
  .success(function(data) {
      $scope.notes = data;
    });
  });
```

Calling Our Factory Service

Our notes controller now needs to use the notes factory.

notes-index-controller.js

```
angular.module("NoteWrangler")
.controller("NotesIndexController", function($scope) {
  $http({method: "GET", url: "/notes"})
  .success(function(data) {
      $scope.notes = data;
    });
  });
```

STAYING SHARP with Angular.JS

Injecting Factory Service

To use methods from the service, we need to first inject it into our controller.

notes-index-controller.js

```
angular.module("NoteWrangler")
.controller("NotesIndexController", function($scope, Note) {

   .success(function(data) {
     $scope.notes = data;
    });
 });
```

Calling Our Factory Service

Now our controller has access to, and can call, the notes factory.

notes-index-controller.js

```
angular.module("NoteWrangler")
.controller("NotesIndexController", function($scope, Note) {
  Note.all()
  .success(function(data) {
    $scope.notes = data;
  });
});
```

Notice Notes has no $. The $ prefix is reserved for built-in Angular services.

STAYING SHARP with Angular.JS