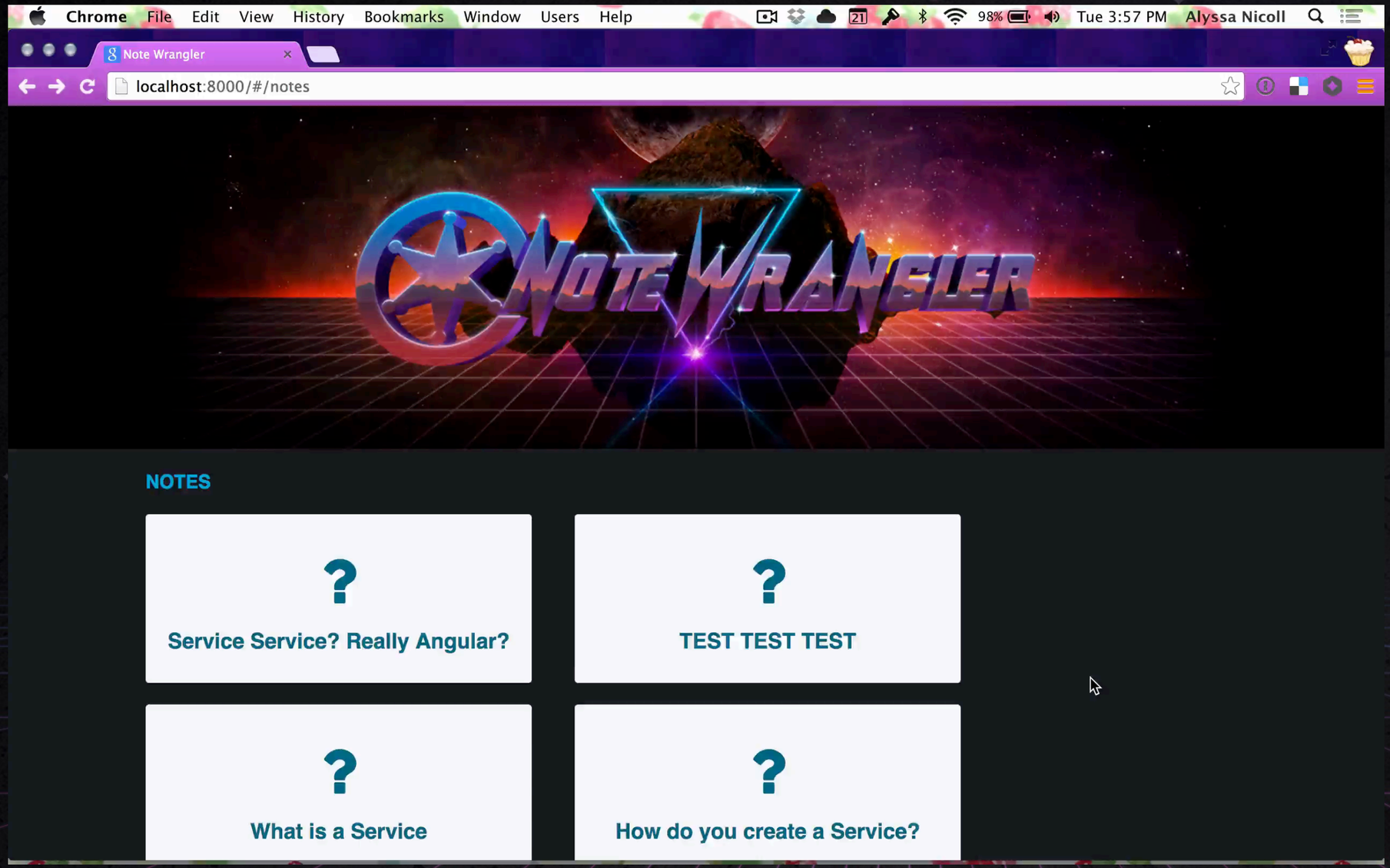

STAYING
SHARP
with
Angular.JS

Directives Level 2

Link Section 3

STAYING SHARP with Angular.JS

We Want to Add Hover Description



Displaying Description on Card Click

notes-index.html

```
...
<nw-card header="note.title"
         description="note.description"></nw-card>
```

nw-card.html

```
<div class="card">
  <h2 class="h3">{{header}}</h2>
  <p class="hidden">{{description}}</p>
</div>
```

style.css

```
.card {
  position: relative;
}
...

.card p.hidden {
  display: none;
}
```

When someone clicks, we want to toggle .hidden class.

jQuery to Toggle the Description

nw-card.html

```
<div class="card">
  <h2 class="h3">{{header}}</h2>
  <p class="hidden">{{description}}</p>
</div>
```

To add this click behavior, we need to run this jQuery:

```
$("div.card").on("click", function() {
  $("div.card p").toggleClass("hidden");
});
```

But where does this go?

Finding a Location for Our jQuery

It may work, but this is a bad practice.

- jQuery selector is searching the entire DOM.
- The controller may run before the element exists.

nw-card.js

```
angular.module("NoteWrangler")
.directive("nwCard", function nwCardDirective(){
 return {
   ...
   controller: function(){
     $("div.card").on("click", function(){
       $("div.card p").toggleClass("hidden")
     }
   }
...
```

Not in the Controller? Where then?

Introducing Link

The link function is run *after* our directive has been compiled and linked up.

nw-card.js

```
angular.module("NoteWrangler")
.directive("nwCard", function nwCardDirective(){
 return {
   link: function(){
     $("div.card").on("click", function(){
       $("div.card p").toggleClass("hidden")
     }
   }
...
```

This is the best spot to do any DOM manipulation or logic functionality for your directive!

Link Solved One Problem...

Still bad though! We are still searching the entire DOM for these elements.

nw-card.js

```
angular.module("NoteWrangler")
.directive("nwCard", function nwCardDirective(){
 return {
   link: function(){
     $("div.card").on("click", function(){
       $("div.card p").toggleClass("hidden")
     }
   }
...
```

We need a way to access just the HTML elements in the template.

Link has an element parameter!

Link's Element Parameter

Link's first two automatically injected parameters are scope and element.

nw-card.js

```
angular.module("NoteWrangler")
.directive("nwCard", function nwCardDirective(){
 return {
   link: function(scope, element){
     $("div.card").on("click", function(){
       $("div.card p").toggleClass("hidden")
     }
   }
...
```

Refers to the outermost element of the included template.

nw-card.html

```
<div class="card">
  <h2 class="h3">{{header}}</h2>
  <p class="hidden">{{description}}</p>
</div>
```

Using Element to Not Search the DOM

nw-card.js

```
angular.module("NoteWrangler")
.directive("nwCard", function nwCardDirective(){
 return {
   link: function(scope, element){
     element.on("click", function(){
        element("div.card p").toggleClass("hidden");
     };
   }
...
```

Now we're no longer searching the DOM!

Link's Attrs Parameter

Link has another parameter: attrs, which stands for "attributes."

nw-card.js

```
angular.module("NoteWrangler")
.directive("nwCard", function nwCardDirective(){
 return {
   link: function(scope, element, attrs){
     element.on("click", function(){
        element("div.card p").toggleClass("hidden");
     };
     console.log(attrs.header);
   }
...
```

Refers to the attributes on the directive element.

notes-index.html

```
...
<nw-card header="note.title"
         description="note.description"></nw-card>
```

Another Example Using Link

We also need to parse the description for the body of our note cards.

Markers on a **DOM element** that tell AngularJS's HTML compiler `$compile` to attach a specified behavior to that DOM element.

Link has *pre* & *post* functions. **Anything that manipulates the DOM goes here!**

See the Markdown?

Include markdown.js Library

We'll need it to parse the body of our nwCard directives.

index.html

```
<!DOCTYPE html>
<html lang="en" ng-app="NoteWrangler">
    ...
    <div class="main-wrapper">
      <div ng-view></div>
    </div>

    <!-- Vendors -->
    <script src="/js/vendor/markdown.js"></script>
    ...
```

Use the Markdown Library

Use the library to change the body from Markdown to HTML.

index.html

```
<nw-card header="note.title" body="note.description" ...></nw-card>
```

nw-card.js

```
angular.module("NoteWrangler")
.directive("nwCard", function() {
  return {
    ...
    scope: {
      header: "=",
      body: "=",
      ...
    },
    link: function(scope, element) {
      scope.body = markdown.toHTML( scope.body );
```

Problem: Tags in Our Body

Now we have a new problem: We seem to be getting the HTML tags in our output.

```
<p>Markers on a
<strong>DOM
element</strong> that tell
AngularJS&amp;#39;s HTML
compiler
<code>$compile</code>
to attach a specified
```

Angular doesn't know if it can trust this injected HTML.
How do we tell Angular it's safe?

```
<p><code>link:
function(scope, element) {
scope.body =
$sce.trustAsHtml(
markdown.toHTML(scope.bo
); }</code> </p>
```

We'll need to use two new things:

- ngBindHtml
- $sce

Step 1: Use ngBindHtml Attribute in Template

The {{}} bindings in templates won't insert HTML directly for safety reasons. We need to use the ngBindHtml attribute to bind the body.

nw-card.html

```
<div class="card">
  ...
  <div class="card-hidden">
   <a ng-href="#/notes/{{id}}">{{body}}</a>
 </div>
</div>
```

nw-card.html

```
<div class="card">
  ...
  <div class="card-hidden">
    <a ng-href="#/notes/{{id}}" ng-bind-html="body"></a>
  </div>
</div>
```

Step 2: Use $sce Service in the Directive

We can tell Angular the HTML we want to insert is safe by using the $sce service.

nw-card.js

```
angular.module("NoteWrangler")
.directive("nwCard", function($sce) {
  return {
    ...
    link: function(scope, element) {
    scope.body = $sce.trustAsHtml(markdown.toHTML(scope.body));
    }
```

Strict Contextual Escaping service tells Angular, “I trust this as HTML; don’t worry about escaping HTML that could be potentially unsafe.”

Cards Now Have Beautiful Bodies