

# Warming Up With ember.js

## Level 1 - Warming Up

Setting up Ember.js



# Ember.js

“A framework for creating ambitious web applications”



# Need To Know

HTML & CSS

JavaScript

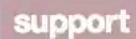
jQuery

# Don't need to know

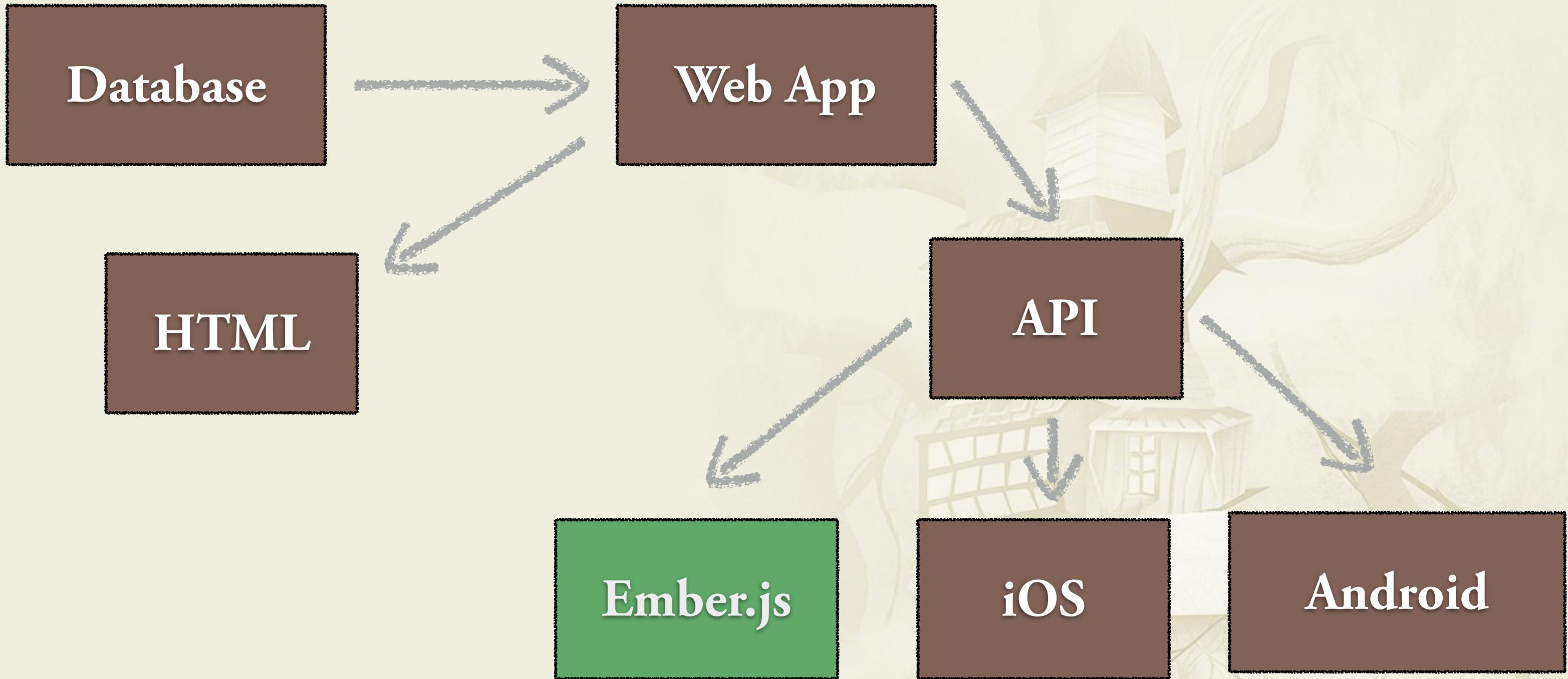
Ruby & Rails

Python & Django



| Topics – Discourse Meta   |   |   |       |  |            |          |       |  |   |   |
|---|---|---|-------|--|------------|----------|-------|--|---|---|
| meta.discourse.org  |   |   |       |  |            |          |       |  |   |   |
|    |   |   |       |  | Log In     |          |       |  |   |   |
| All Categories  |   |   |       | Latest   | Categories |          |       |  |  |  |
| Topic ▾   | Category  | Users   | Posts | Likes  | Views      | Activity |       |  |   |   |
| <a href="#">  Welcome to meta.discourse.org</a>                                       |   |    | 5     | 105   | 13.0K      | 31 Jan   | 7 Feb |  |   |   |
| Welcome to meta, the official site for discussing the next-gen open source Discourse forum software. You'll find topics on features, bugs, hosting, development, and general support here. Discourse is early beta softwar... <a href="#">read more</a> |   |   |       |  |            |          |       |  |   |   |
| <a href="#">Something wrong with admin_js translations?</a>   |    |   | 1     |  | 2          | 2m       | 2m    |  |   |   |
| <a href="#">Back button does not work on old Android browsers</a>   |  |  | 11    |  | 40         | 1d       | 9m    |  |   |   |
| <a href="#">Invalid multibyte char (US-ASCII) on 'rake spec'</a>  |  |  | 4     |  | 22         | 11h      | 27m   |  |   |   |
| <a href="#">Trigger pad 'stops short' on reply as new topic - Firefox</a>   |  |  | 1     |  | 7          | 40m      | 40m   |  |   |   |
| <a href="#">Allow view for topics in all children categories</a>  |  |  | 3     |  | 43         | 2d       | 1h    |  |   |   |
| <a href="#">Translation Tools: Transifex? Localeapp?</a>  |  |  | 25    | 26  | 680        | 25 Jun   | 1h    |  |   |   |
| <a href="#">How to add advertisements to your Discourse forum?</a>  |  |  | 1     | 1   | 18         | 1h       | 1h    |  |   |   |
| <a href="#">Adding Advertising</a>  |  |  | 43    | 50  | 1.6K       | 26 Mar   | 1h    |  |   |   |
| <a href="#">Monetizing Discourse using Adsense</a>  |  |  | 13    | 7   | 678        | 5 Apr    | 1h    |  |   |   |

# Why Ember.js?



# Why Ember.js

You want your application to feel responsive

## Reviews

- No reviews yet. Be the first to write one!



Review



# When Not to Use

When your web app doesn't have a lot of interactivity.

- Blog
- Newspaper
- Static Content



# Getting Started With Ember.js

## 1. Grab the starter kit & Ember Data

<http://emberjs.com/>

<https://github.com/emberjs/data>

## 2. We're using Twitter Bootstrap

<http://getbootstrap.com/>



# Setting Up index.html

index.html

```
<html>
  <head>
    <script src="jquery.js"></script>
    <script src="handlebars.js"></script>
    <script src="ember.js"></script>
    <script src="ember-data.js"></script>
    <script src="app.js"></script>

    <link href="bootstrap.css"
          media="screen"
          rel="stylesheet" />
  </head>
</html>
```





# Welcome to The Flint & Flame!

Everything you need to make it through the winter.

[Browse All 6 Items »](#)

## Flint

On Sale   Seasonal

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

[Buy for \\$99](#)

## Matches

On Sale   Seasonal

One end is coated with a material that can be ignited by frictional heat generated by striking the match against a suitable surface.

[Buy for \\$499](#)

## Tinder

On Sale

Tinder is easily combustible material used to ignite fires by rudimentary methods.

[Buy for \\$499](#)

# The Ember Application

We need a single object to contain all functionality of our app.

app.js

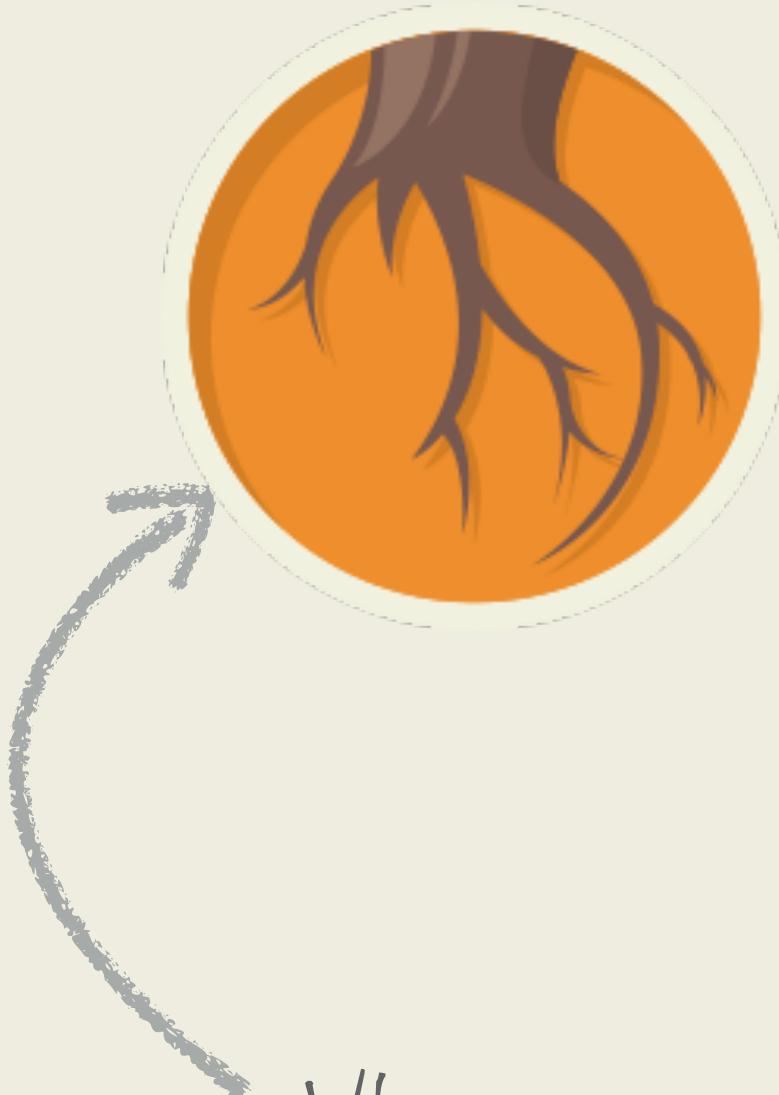
```
var App = Ember.Application.create({ });
```



We're namespacing our application App, but  
you could name this variable anything



# Ember Application



The roots of our application  
Need to create an application only once  
Everything we do comes after

When you see this icon, it's our application



# Ember Application Options

Inside our application we can send in a JavaScript Object with options.

For example, if we wanted to log out a message to the browser every time a new page is accessed.

```
var App = Ember.Application.create({  
  LOG_TRANSITIONS: true  
});
```

Not required, but helpful for debugging!



# The HTML Content

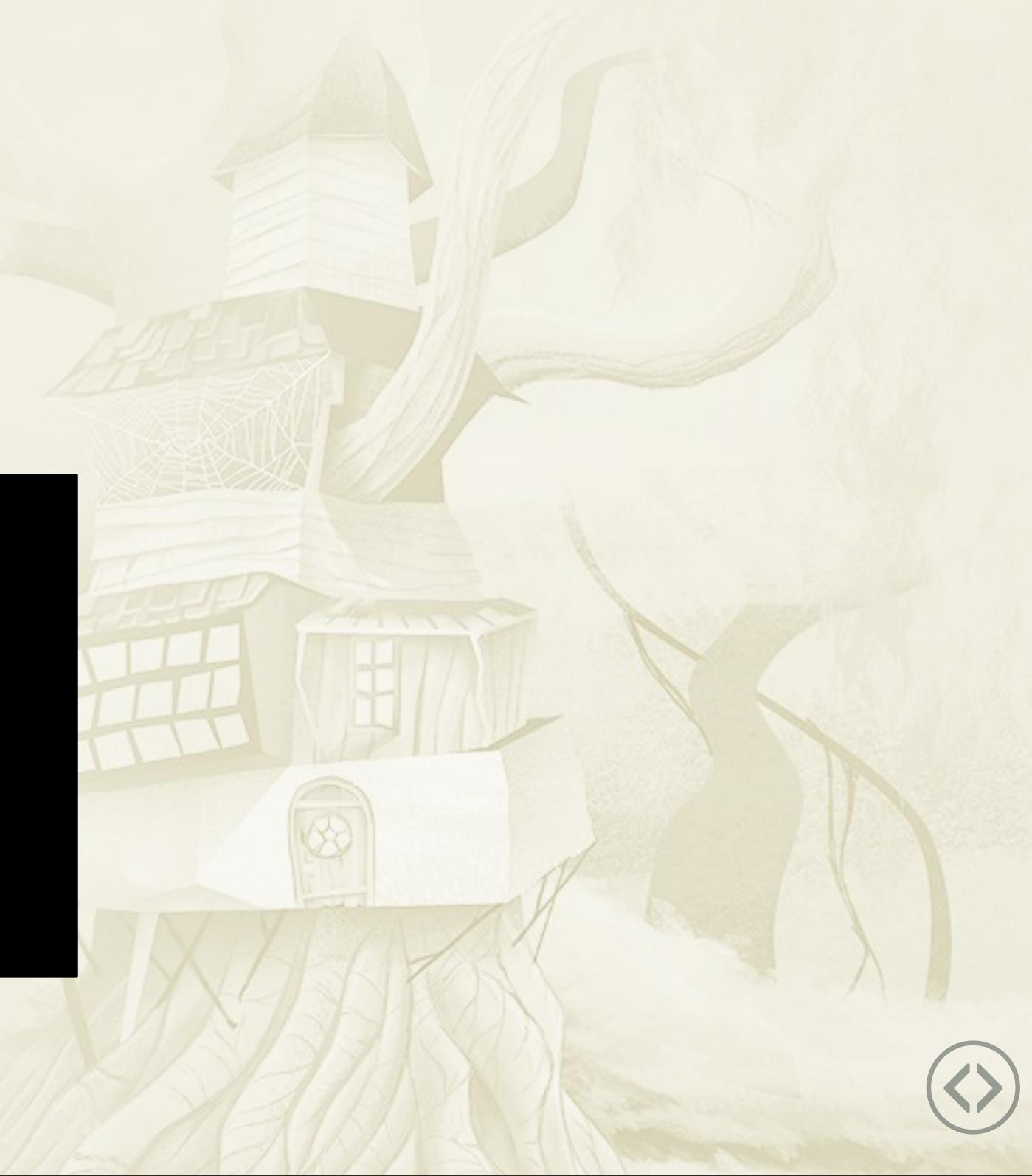
app.js

```
var App = Ember.Application.create({  
  LOG_TRANSITIONS: true  
});
```



index.html

```
<html>  
  ...  
  <body>  
    <div class='navbar'>...</div>  
    <div class='container'>...</div>  
    <footer class='container'>...</footer>  
  </body>  
</html>
```



# Running Our First App

If we open the app and view the source you'll notice Ember added a body class and data-ember-extension to our code.

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body class='ember-application' data-ember-extension='1'>
    <div class='navbar'>...</div>
    <div class='container'>...</div>
    <footer class='container'>...</footer>
  </body>
</html>
```

So Ember will know the part of the page it will control.



# We Need to Dynamically Update

We're going to want to dynamically update the content inside this body.  
Thus, we need a templating language.

***body***

***div.navbar***

***div.container***

***footer.container***

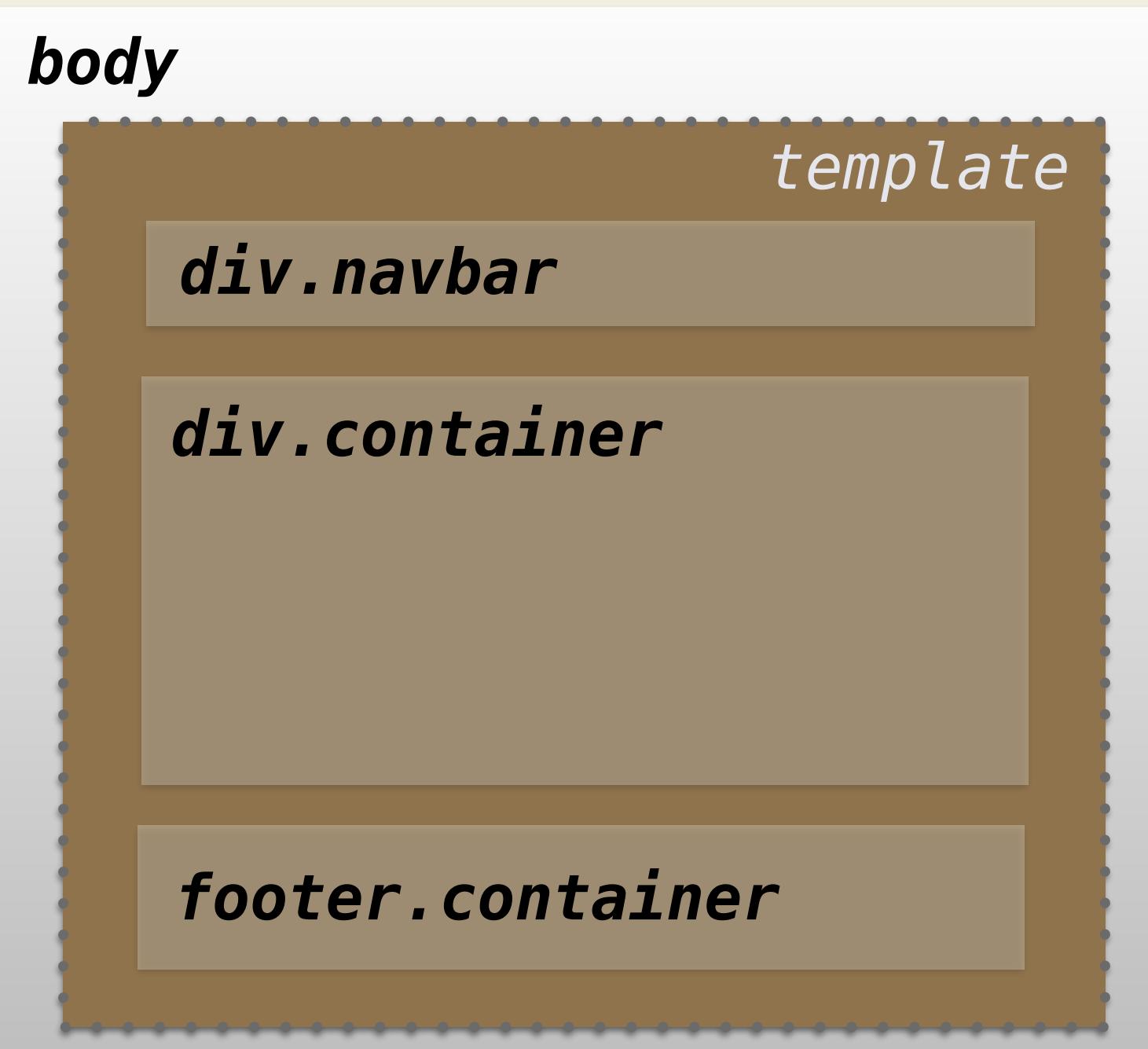
```
<body>
  <div class='navbar'>...</div>
  <div class='container'>...</div>
  <footer class='container'>...</footer>
</body>
```

We want to move everything inside  
`<body>` into a template



# Handlebars Template

Handlebars templates look like regular HTML with handlebars expressions



This is a Handlebars template

```
<body>
  <script type='text/x-handlebars'>
    <div class='navbar'>...</div>
    <div class='container'>...</div>
    <div class='container'>...</div>
  </script>
</body>
```



# Running With Handlebars

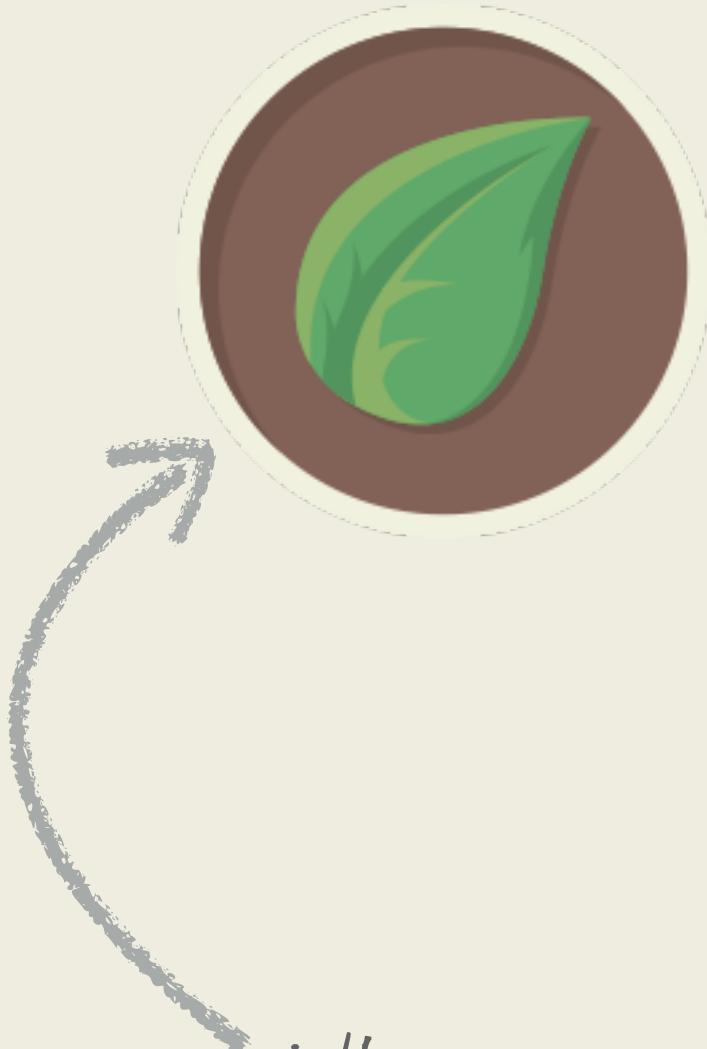
Ember by default will render the Handlebars template into a div.

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body class='ember-application' data-ember-extension='1'>
    <div id='ember248' class='ember-view'>
      <div class='navbar'>...</div>
      <div class='container'>...</div>
      <footer class='container'>...</footer>
    </div>
  </body>
</html>
```

So Ember can uniquely identify this div



# Ember Template



Works just like HTML

Ember uses the Handlebars.js for templates

The part of our application people see

When you see this icon, it's an Ember template



# Warming Up With ember.js

Level 1 - Warming Up

Named Templates



# Review

We can put HTML inside a Handlebars template

index.html

```
<body>
  <script type='text/x-handlebars'>
    <div class='navbar'>...</div>
    <div class='container'>...</div>
    <footer class='container'>...</footer>
  </script>
</body>
```

What if we wanted to make the content  
in this template dynamic?



# Adding Data to Templates

index.html

```
<script type='text/x-handlebars'>
  <div class='navbar'>...</div>
  <div class='container'>
    <h1>Welcome to {{siteName}}!</h1>
  </div>
  <footer class='container'>...</footer>
</script>
```

Called a "handlebars expression"

{{siteName}}'s value will need to be provided by our Ember app, so it can print The Flint & Flame!

Later on in this course we'll be using templates to add dynamic content.



# Our Current Webpage Layout

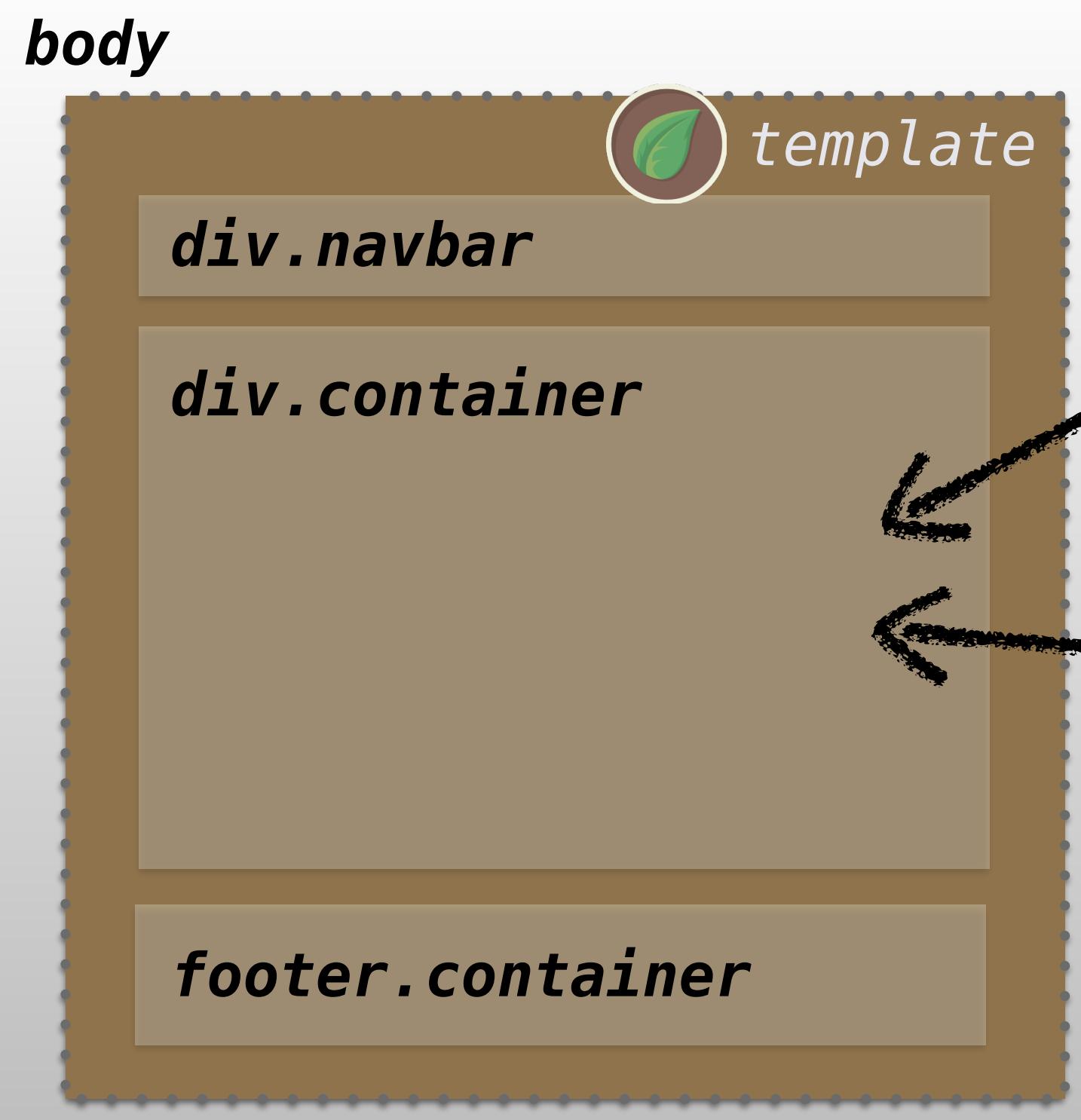


We are going to have multiple pages on our website.

- Homepage
- About Page
- Product List



# The Different Pages



## About Page

```
<h1>About The Fire Spirits</h1>
```

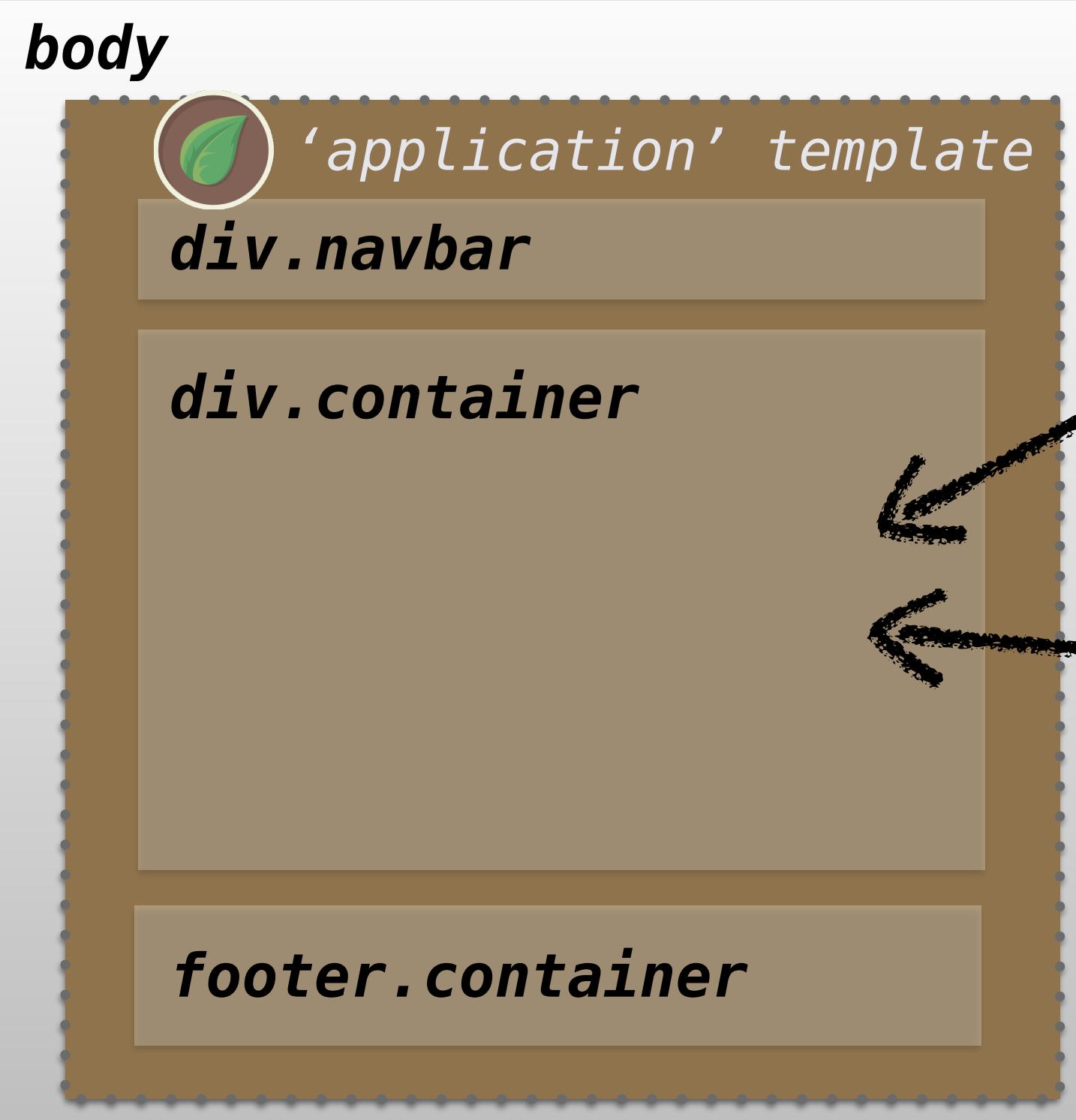
## Home Page

```
<h1>Welcome to The Flint & Flame!</h1>
```

We want this to be the default



# We Need to Name our Templates



**'about' template**

<h1>About The Fire Spirits</h1>

**'index' template**

<h1>Welcome to The Flint & Flame!</h1>

We want this to be the default



# Adding the Template Name

Each template needs a unique name.

index.html

```
<script type='text/x-handlebars' data-template-name='application'>
  <div class='navbar'>...</div>
  <div class='container'>...</div>
  <footer class='container'>...</footer>
</script>
```



The application template is shown  
for every page by default.



# Adding the Homepage Template

index.html

```
<script type='text/x-handlebars' data-template-name='application'>
  <div class='navbar'>...</div>
  <div class='container'>...</div>
  <footer class='container'>...</footer>
</script>

<script type='text/x-handlebars' data-template-name='index'>
  <h1>Welcome to The Flint & Flame!</h1>
</script>
```



'index' is what we'll call the homepage template name



# Adding our About Template

index.html

```
<script type='text/x-handlebars' data-template-name='application'>
  <div class='navbar'>...</div>
  <div class='container'>...</div>
  <footer class='container'>...</footer>
</script>

<script type='text/x-handlebars' data-template-name='index'>
  <h1>Welcome to The Flint & Flame!</h1>
</script>

<script type='text/x-handlebars' data-template-name='about'>
  <h1>About The Fire Spirits</h1>
</script>
```





---

© 2013 The Flint & Flame

Credits

Only our application template is showing!

No welcome message!



# But where do they go?

We need a way of telling our templates where on the page to render.

```
<script type='text/x-handlebars' data-template-name='application'>
  <div class='navbar'>...</div>
  <div class='container'>...</div>
  <footer class='container'>...</footer>
</script>

<script type='text/x-handlebars' data-template-name='index'>
  <h1>Welcome to The Flint & Flame!</h1>
</script>

<script type='text/x-handlebars' data-template-name='about'>
  <h1>About The Fire Spirits</h1>
</script>
```



# Outlet

Using the handlebars expression “outlet” we’re giving our code a hint as to where templates should be inserted.

```
<script type='text/x-handlebars' data-template-name='application'>
  <div class='navbar'>...</div>
  <div class='container'>{{outlet}}</div>
  <footer class='container'>...</footer>
</script>

<script type='text/x-handlebars' data-template-name='index'>
  <h1>Welcome to The Flint & Flame!</h1>
</script>

<script type='text/x-handlebars' data-template-name='about'>
  <h1>About The Fire Spirits</h1>
</script>
```



If our Ember code reaches an {{outlet}}, by default it will look to find a template named ‘index’ and render that in place of the outlet.





# Welcome to The Flint & Flame!

---

© 2013 The Flint & Flame

Credits



Our index template is rendered within our application template



# Warming Up With ember:js

## Level 1 - Warming Up

The Router



# Routing Our Way In

index.html

```
<script type='text/x-handlebars' data-template-name='application'>
  <div class='navbar'>...</div>
  <div class='container'>
    {{outlet}} ←
  </div>
  <footer class='container'>...</footer>
</script>

<script type='text/x-handlebars' data-template-name='about'>
  <h1>About The Fire Spirits</h1>
</script>
```

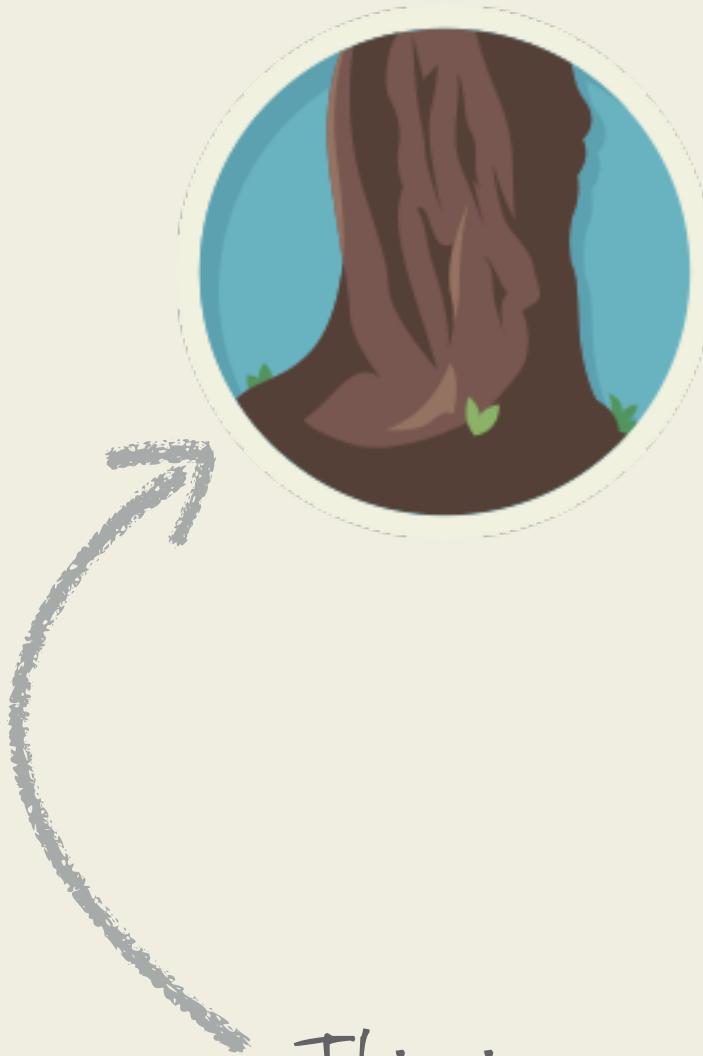


How can we ever render the 'about' template?

and map it to a URL, like <http://example.com/about> ?



# The Ember Router



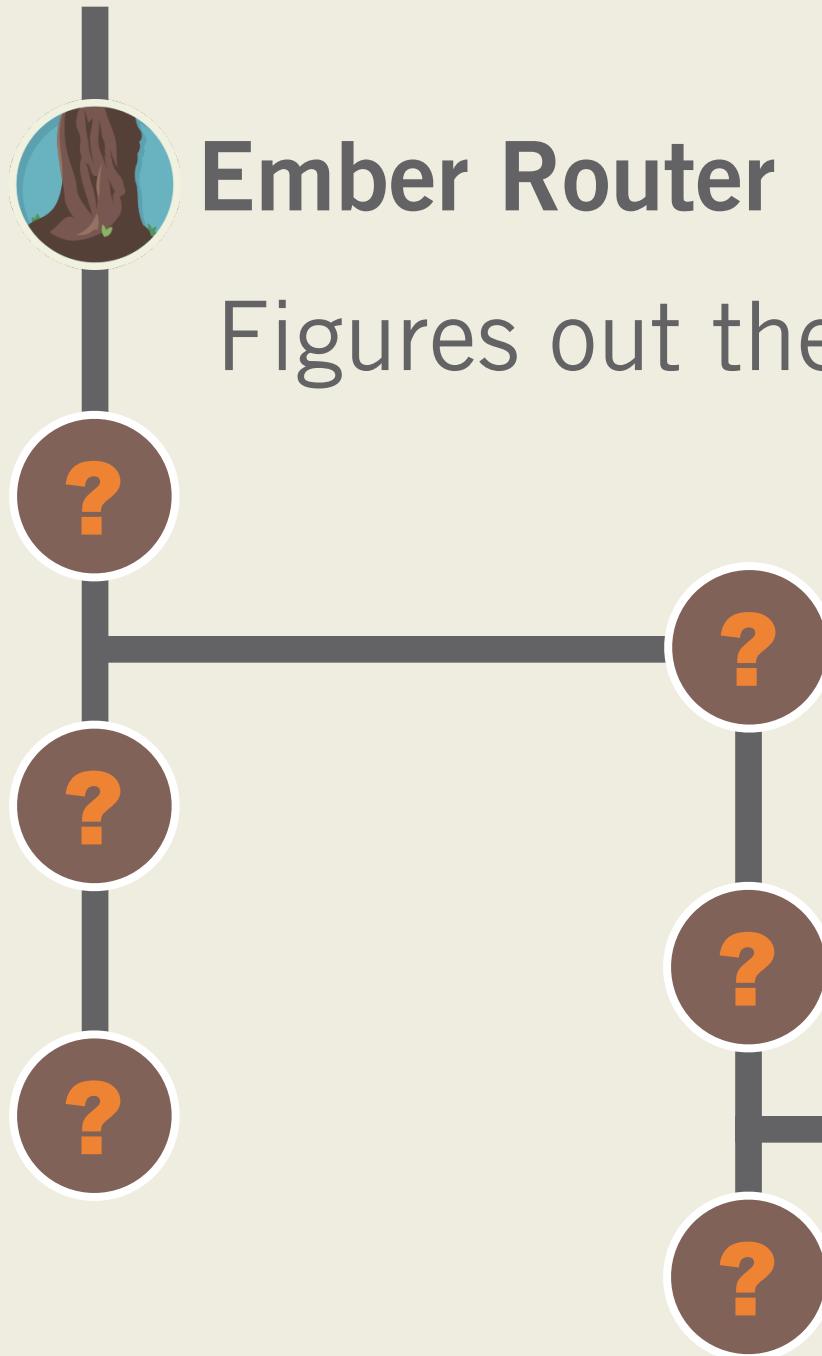
This is our Router icon

Translates a path into a route.  
Like a tree trunk — all requests start here.



# Routing Our Way In

Browser Request



**Ember Router**

Figures out the route name for each request

**Handlebars Template**



# Introducing the Router

app.js

```
var App = Ember.Application.create({  
  LOG_TRANSITIONS: true  
});  
  
App.Router.map(function() {  
  ...  
});
```

Every page for our website will be defined by the router.



# How can we render the ‘about’ template?

app.js

```
App.Router.map(function() {  
  ...  
});
```

index.html

```
<script type='text/x-handlebars' data-template-name='application'>  
...  
  {{outlet}}  
...  
</script>  
  
<script type='text/x-handlebars' data-template-name='about'>  
  <h1>About The Fire Spirits</h1>  
</script>
```



# The about Route

app.js

```
App.Router.map(function() {  
  this.route('about');  
});
```



URL

http://example.com#/about

Path

/about

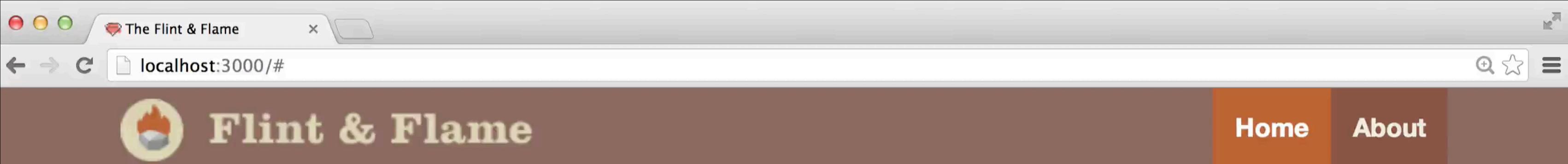
Route

about

When an outlet is found, ‘about’ template is loaded into it.

```
<script type='text/x-handlebars' data-template-name='application'>  
...  
  {{outlet}} ←  
...  
</script>  
  
<script type='text/x-handlebars' data-template-name='about'>  
  <h1>About The Fire Spirits</h1>  
</script>
```





# Flint & Flame

[Home](#)[About](#)

# Welcome to The Flint & Flame!

© 2013 The Flint & Flame

[Credits](#)

Elements Resources Network Sources Timeline Profiles Audits **Console** Ember

DEBUG: -----

[ember-1.0.0.js?body=1:394](#)

DEBUG: Ember.VERSION : 1.0.0

[ember-1.0.0.js?body=1:394](#)

DEBUG: Handlebars.VERSION : 1.0.0

[ember-1.0.0.js?body=1:394](#)

DEBUG: jQuery.VERSION : 1.10.2

[ember-1.0.0.js?body=1:394](#)

DEBUG: -----

[ember-1.0.0.js?body=1:394](#)

Transitions into 'index'

[ember-1.0.0.js?body=1:394](#)

>



<top frame>



<page context>



All

Errors

Warnings 10

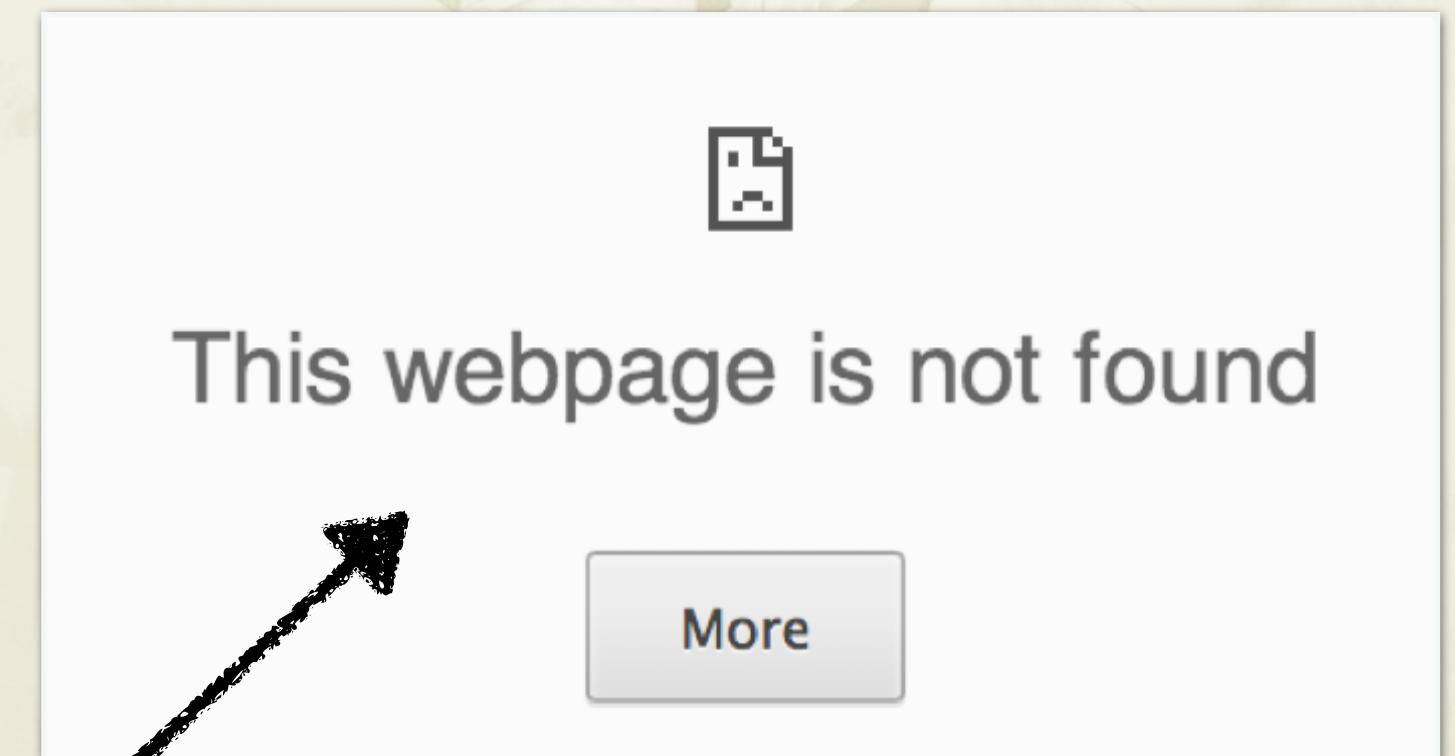


# What's with the Hash Sign?

<http://example.com#/about>



<http://example.com/about>



There is no 'about/index.html' or 'about' file  
path our application knows about



# Where our Application Exists

In Ember.js, our entire application is loaded through **one file**.

In our case *index.html* which is loaded from <http://example.com>

No matter what page we load...

- Homepage
- About Page
- Product List

...we will have first called up *index.html*

This loads up all our templates!



# All Templates Sent!



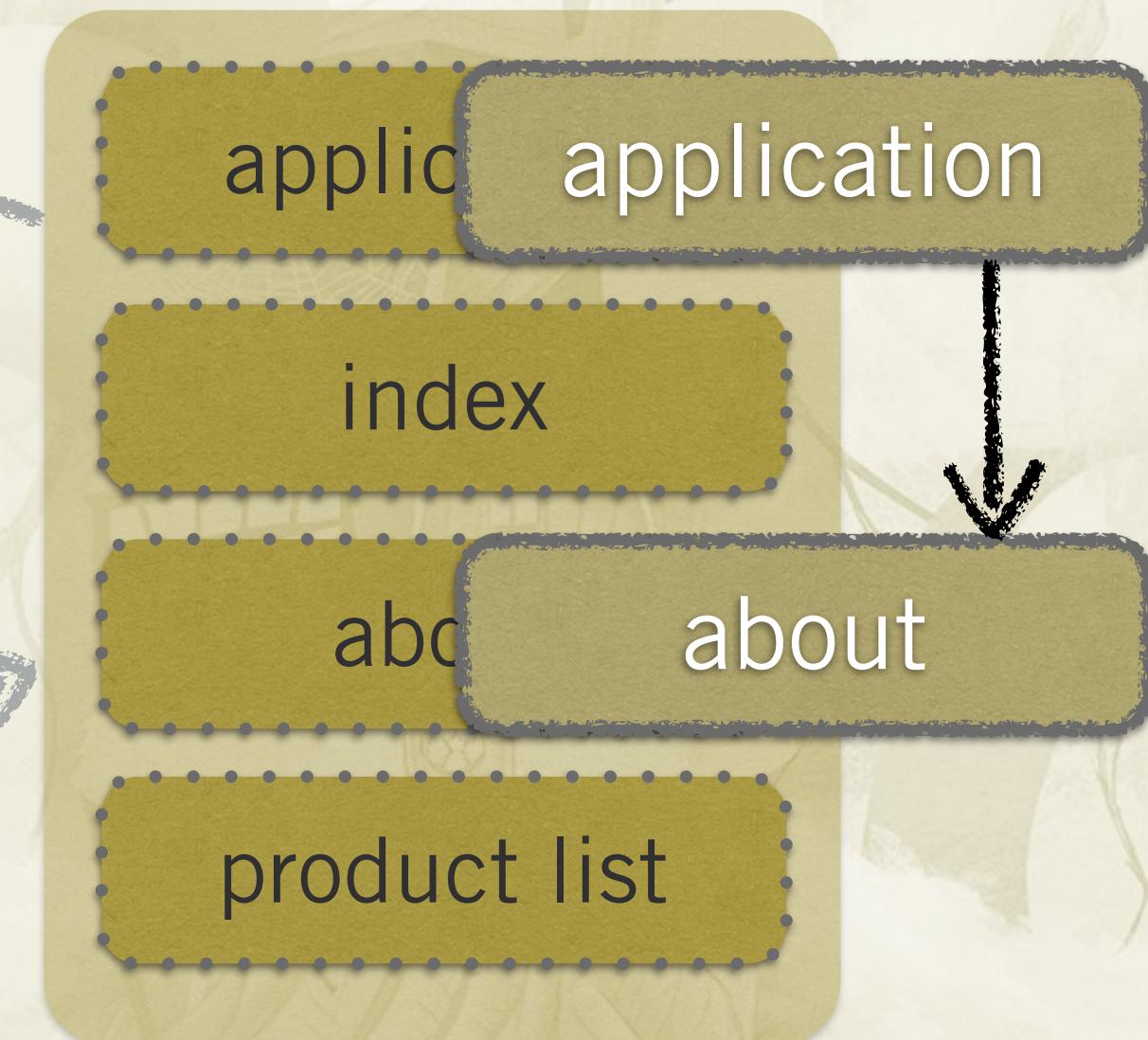
In Ember.js, all our templates are sent over when the application loads.

Server



Initial Request

Client Browser



# What does the hash say?

http://example.com#/about



Loads '' or root file path

When a browser see's this it knows the file path information is over

http://example.com/about

Loads 'about' file path

http://example.com/store#/shopping\_cart

Loads 'store' file path



# Back to Our About Route

app.js

```
App.Router.map(function() {  
  this.route('about');  
});
```



**URL** http://example.com#/about

**File Path** <none>

**Ember Path** /about

**Route** about

What if we want the '/aboutus' path to use the about route?



# Using '/aboutus' custom path

app.js

```
App.Router.map(function() {  
  this.route('about', { path: '/aboutus' } );  
});
```

|                   |                             |
|-------------------|-----------------------------|
| <b>URL</b>        | http://example.com#/aboutus |
| <b>Ember Path</b> | /aboutus                    |
| <b>Route</b>      | about                       |



## About The Fire Spirits



# What about the Index Route?

app.js

```
App.Router.map(function() {  
  this.route('about', { path: '/aboutus' } );  
 ?  
});
```

URL <http://example.com>

Ember Path <none>

Route index

```
this.route('index', { path: '' });
```

?



# No! This is a Default

app.js

```
App.Router.map(function() {  
  this.route('about', { path: '/aboutus' } );  
});
```



URL <http://example.com>

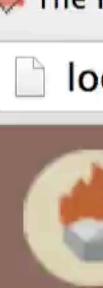
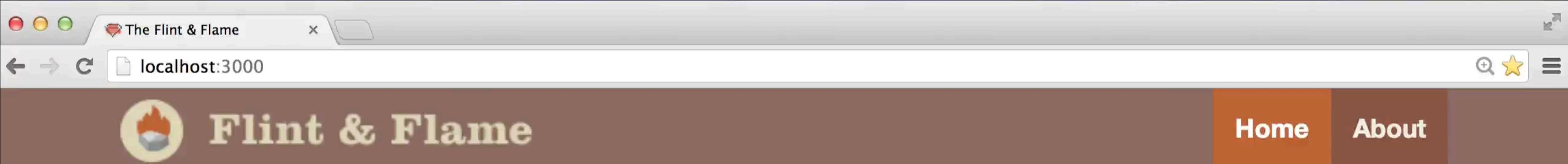
Ember Path <none>

Route index

```
this.route('index', { path: '' } );
```

Default Behavior!





# Flint & Flame

[Home](#)[About](#)

# Welcome to The Flint & Flame!

© 2013 The Flint & Flame

Credits

Elements Resources Network Sources Timeline Profiles Audits **Console** Ember

DEBUG: -----

[ember-1.0.0.js?body=1:394](#)

DEBUG: Ember.VERSION : 1.0.0

[ember-1.0.0.js?body=1:394](#)

DEBUG: Handlebars.VERSION : 1.0.0

[ember-1.0.0.js?body=1:394](#)

DEBUG: jQuery.VERSION : 1.10.2

[ember-1.0.0.js?body=1:394](#)

DEBUG: -----

[ember-1.0.0.js?body=1:394](#)

Transitions into 'index'

[ember-1.0.0.js?body=1:394](#)

>

# Warming Up With ember.js

## Level 2 - Rendering the Flame

Handlebars



# Review

Router is solely responsible for all URLs.

app.js

```
App.Router.map(function() {  
  this.route('about');  
});
```



How do we link to the *about* and *index* route?



# Linking to Routes

Router is solely responsible for all URLs.

app.js

```
App.Router.map(function() {  
  this.route('about');  
});
```

index.html

```
<a href="#/">Home</a>  
<a href="#/about">About</a>
```



It's considered a bad practice to do this. We don't hard code URL paths. Instead we ask the routes for their path.



# The Handlebars link-to helper

Use the route name to look up the path.

```
{{#link-to 'index'}}Homepage{{/link-to}}  
{{#link-to 'about'}}About{{/link-to}}
```



link-to will determine the path based on the route name.



```
<a class="ember-view" href="#" id='ember2'>Homepage</a>  
<a class="ember-view" href="#/about" id='ember3'>About</a>
```



# Extra Attributes

How to add a custom class to our link.

```
{#{link-to 'index' class='navbar-brand'}}Homepage{{/link-to}}
```



Rendered from template into

```
<a class="ember-view navbar-brand" href="#" id='ember2'>Homepage</a>
```





# Welcome to The Flint & Flame!

© 2013 The Flint & Flame

## Credits

```
x Elements Resources Network Sources Timeline Profiles Audits | Console | Ember  
DEBUG: ----- ember-1.0.0.js?body=1:394  
DEBUG: Ember.VERSION : 1.0.0 ember-1.0.0.js?body=1:394  
DEBUG: Handlebars.VERSION : 1.0.0 ember-1.0.0.js?body=1:394  
DEBUG: jQuery.VERSION : 1.10.2 ember-1.0.0.js?body=1:394  
DEBUG: ----- ember-1.0.0.js?body=1:394  
Transitions into 'index' ember-1.0.0.js?body=1:394
```



# Welcome to The Flint & Flame!



© 2013 The Flint & Flame

Credits

Elements Resources Network Sources Timeline Profiles Audits Console Ember

Styles Computed »

element.style { }

media="screen" 2-10  
bootstrap.css?body=1:3770  
@media (min-width: 768px)  
bootstrap.css?body=1:3771  
.navbar > .container .navbar-  
brand {  
margin-left: -15px;  
}

media="screen" 2-10

a#ember269.ember-view.navbar-brand.active

html body #ember249 div div a#ember269.ember-view.navbar-brand.active

10

☰ ≡ 🔎 ⚙️

# tagName option

How to change the HTML element.

index.html

```
{#{link-to 'index' tagName='li'}}Home{{/link-to}}  
{#{link-to 'about' tagName='li'}}About{{/link-to}}
```



```
<li class='ember-view' id='ember1'>Home</li>  
<li class='ember-view' id='ember2'>About</li>
```



Notice the href disappeared!

It will still work, because Ember adds click handlers.

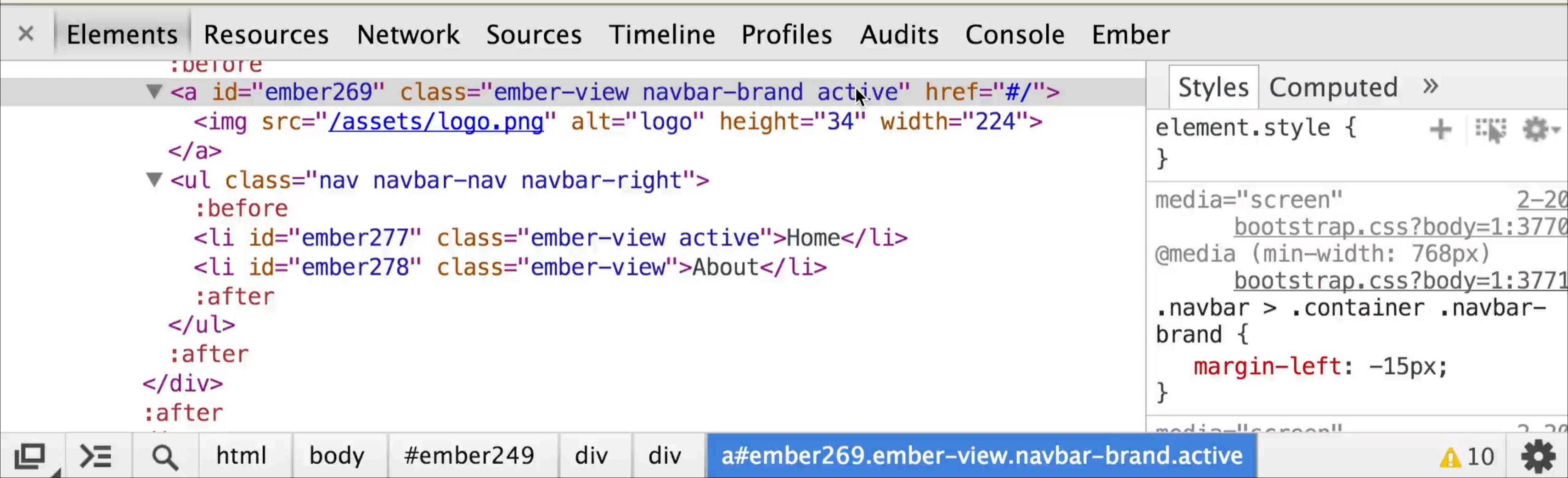


a#ember269.ember-view.navbar-brand.active 254px x 44px

# Welcome to The Flint & Flame!

© 2013 The Flint & Flame

## Credits



# Warming Up With ember.js

## Level 2 - Rendering the Flame

Controller Basics



# Review

Mapped paths to route names.

app.js

```
App.Router.map(function() {  
  this.route('about');  
});
```



Link to other pages using route names, not paths.

index.html

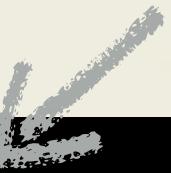
```
{{#link-to 'about' tagName='li'}}About{{/link-to}}
```



# Template Variables

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  <p>There are {{productsCount}} products</p>
</script>
```



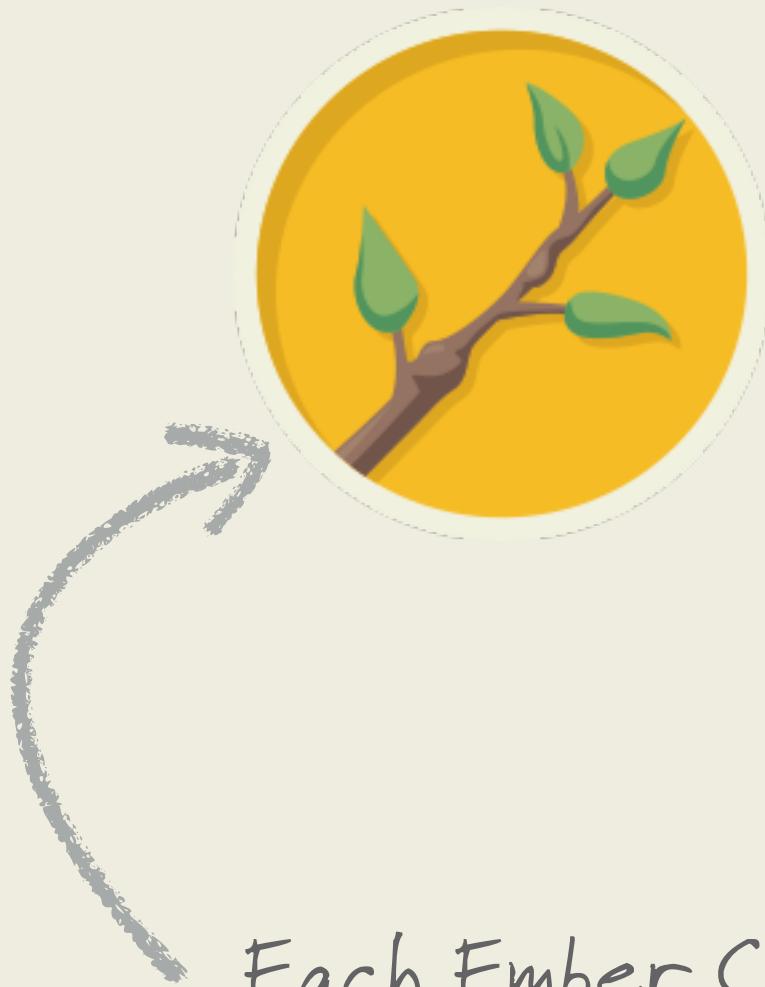
You can use properties in your template



How do you set a property for use in a template?



# Ember Controller



Where a template looks to find a property value

Decorate your applications data for the template

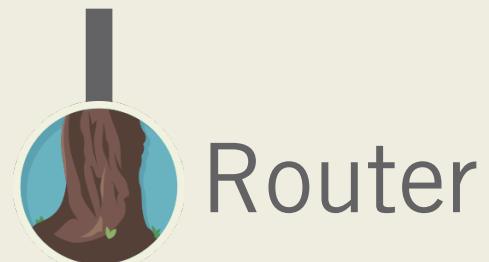
Contains information that is not saved to a server

Each Ember Controller will use this icon



# Routing Our Way In

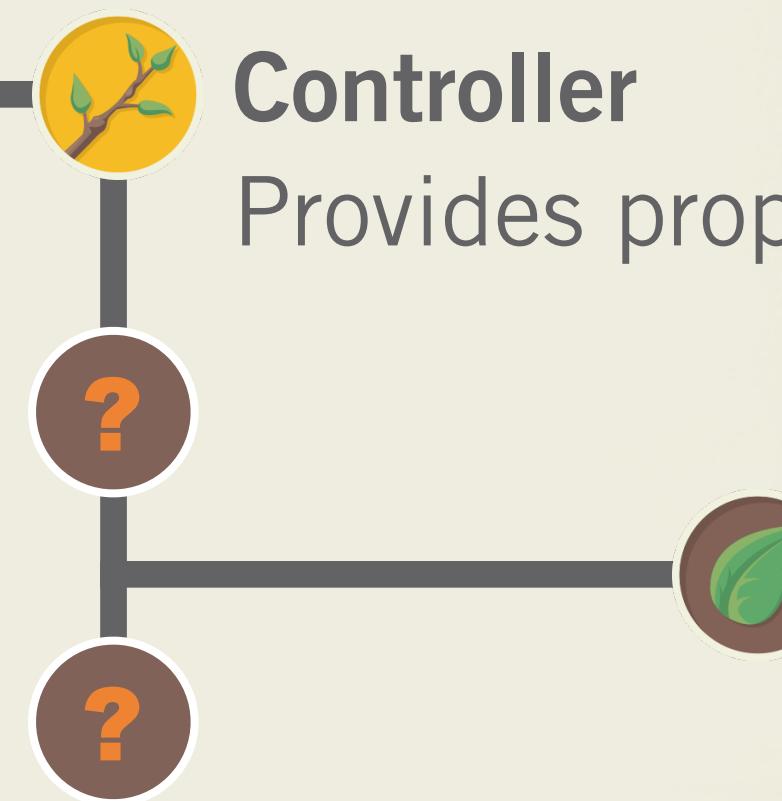
Browser Request



Router

Controller

Provides properties for the template



Handlebars Template



# Asking the Controller

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  <p>There are {{productsCount}} products</p>
</script>
```



app.js

```
App.IndexController = Ember.Controller.extend({
  productsCount: 6
});
```



There are 6 products



# Every Route has a Default Controller

```
App.Router.map(function() {  
  this.route('about');  
});
```

Behind the scenes, Ember creates this About controller.

```
App.AboutController = Ember.Controller.extend({ });
```

If we want to declare properties that get rendered inside a template, then we can declare this Controller.



# Binding with Metamorph!

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  <p>There are {{productsCount}} products</p>
</script>
```



```
<p>There are
<script id="metamorph-2-start" type="text/x-placeholder"></script>
6
<script id="metamorph-2-end" type="text/x-placeholder"></script>
products
</p>
```

Ember needs to wrap properties so it can potentially update it later. (AKA. Binding)



# Binding Attributes?

app.js

```
App.IndexController = Ember.Controller.extend({  
  productsCount: 6,  
  logo: '/images/logo.png'  
});
```

index.html

```
<script type='text/x-handlebars' data-template-name='index'>  
  <p>There are {{productsCount}} products</p>  
  <img src='{{logo}}' alt='Logo' />  
</script>
```





# Welcome to The Flint & Flame!

There are 6 products



X Elements Resources Network Sources Timeline Profiles Audits Console Ember

```
<h1>Welcome to The Flint & Flame!</h1>


/p>
</script>
/images/logo.png
<script id='metamorph-2-end' type="text/x-placeholder"></script>
">
```



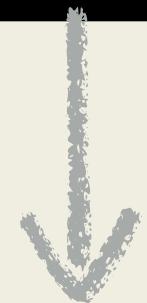
We need to bind attributes differently.



# How to Bind Attributes

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  <p>There are {{productsCount}} products</p>
  <img {{bind-attr src='logo'}} alt='Logo' />
</script>
```



```

```



Ember meta data is still added to enable binding





# Welcome to The Flint & Flame!

There are 6 products



X Elements Resources Network Sources Timeline Profiles Audits Console Ember

```

<script id="metamorph-0-end" type="text/x-placeholder"></script>
<script id="metamorph-1-end" type="text/x-placeholder"></script>
:after
</div>
▶ <footer class="container">...</footer>
</div>
▶ <iframe id="rdbIndicator" width="100%" height="270" border="0" src="chrome-extension://oknpjjbmpnndlpmnhmekjpocelpnlfdi/indicator.html" style="display: none; border: 0; position: fixed; left: 0; top: 0; z-index: 2147483647">
...</iframe>
</body>
</html>
```

Styles Computed »

element.style { }

media="screen" 2-60  
img { bootstrap.css?body=1:294  
vertical-align: middle;  
}

media="screen" 2-60  
img { bootstrap.css?body=1:100  
border: 0;  
}

□ ╳ 🔎 ⚡ html body.ember-application div#ember251.ember-view div.container img ! 10 ⚙

# Dynamic Content

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  <p>Rendered on {{time}}</p>
</script>
```



How do we fetch the current time?

We need to create a property that calls this function:

```
(new Date()).toDateString()
```

This is more then just a value, we need to execute some code to get the value.



# Controller Functions

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  <p>Rendered on {{time}}</p>
</script>
```



app.js

```
App.IndexController = Ember.Controller.extend({
  productsCount: 6,
  logo: '/images/logo.png',
  time: function() {
    return (new Date()).toDateString()
  }
});
```



We need to tell Ember this is a property and to call it



# Controller Functions as Properties

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  <p>Rendered on {{time}}</p>
</script>
```



app.js

```
App.IndexController = Ember.Controller.extend({
  productsCount: 6,
  logo: '/images/logo.png',
  time: function() {
    return (new Date()).toDateString()
  }.property()
});
```



time will be called, and that value used as a property





# Welcome to The Flint & Flame!

There are 6 products



## Flint & Flame

Rendered on Tue Nov 26 2013

Elements Resources Network Sources Timeline Profiles Audits Console Ember

```
▶ <p>...</p>

▼ <p>
  "Rendered on "
  <script id="metamorph-3-start" type="text/x-placeholder"></script>
  "Tue Nov 26 2013"
  <script id="metamorph-3-end" type="text/x-placeholder"></script>
</p>
<script id="metamorph-0-end" type="text/x-placeholder"></script>
<script id="metamorph-1-end" type="text/x-placeholder"></script>
:after
...
```

Styles Computed »

element.style { }

media="screen" 2-70

p { bootstrap.css?body=1:335

margin: 0 0 10px;

}

media="screen" 2-70

\*, bootstrap.css?body=1:254

\*:before, \*:after { }



html

body.ember-application

div#ember251.ember-view

div.container

p

(text)

⚠ 10



# Warming Up With ember.js

## Level 3 - A Route Through the Woods

Resource Routes



# Review

app.js

```
App.IndexController = Ember.Controller.extend({  
  productsCount: 6,  
  logo: '/images/test.png',  
  time: function() {  
    return (new Date()).toDateString();  
  }.property()  
});
```

index.html

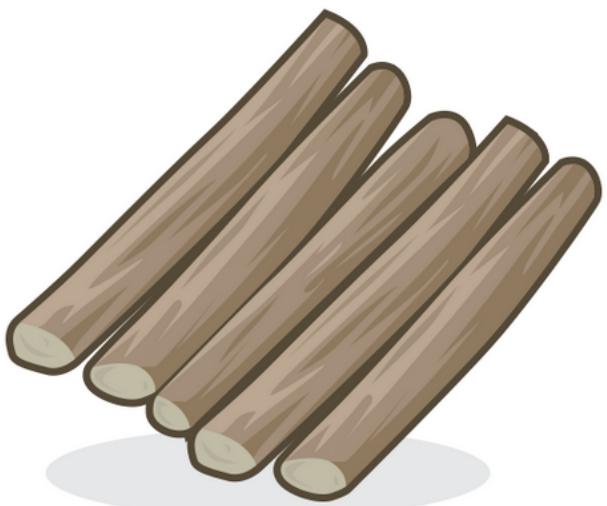
```
<script type='text/handlebars' data-template-name='index'>  
  <p>There are {{productsCount}} products</p>  
  <img {{bind-attr src='logo'}} alt='Logo' />  
  <p>Current Time: {{time}}</p>  
</script>
```

The name of the controller matches  
the route and template names





# Products



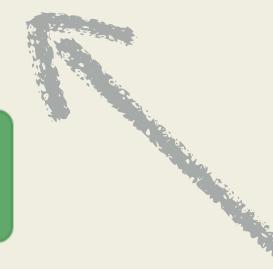
## Flint

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

[Buy for \\$99](#)

## Kindling

Easily combustible small sticks or twigs used for starting a fire.

[Buy for \\$249](#)

We'll show all of our products on  
a new products page

# Use route for adjectives, verbs, adverbs

app.js

```
App.Router.map(function() {  
  this.route('about');  
});
```

Adjective Routes

```
this.route('onsale');  
this.route('expiring');  
this.route('new');  
this.route('internal');
```

Verb Routes

```
this.route('signup');  
this.route('ignite');  
this.route('help');  
this.route('rate');
```



# Use ‘Resource’ Routes for Nouns

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products');  
});
```



Noun Resources

```
this.resource('tree');  
this.resource('review');  
this.resource('books');  
this.resource('contacts');
```

Can be singular or plural

The resource keyword has some additional functionality.

That difference is what this level is all about!



# Resource Route Options

Like with route, we can pass in an optional path

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products', { path: '/items' });  
});
```



# Resource Views

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products');  
});
```

index.html

```
<script type='text/x-handlebars' data-template-name='products'>  
  <h1>Products</h1>  
</script>
```

Matches the route name



# Linking to Resources

We can link to this route with its name like any other.

index.html

```
<script type='text/x-handlebars' data-template-name='application'>
...
{{#link-to 'products' tagName='li'}}Products{{/link-to}}
...
</script>
```



<http://example.com#/products>



**Flint & Flame**

Home

About

Products

# Products

© 2013 The Flint & Flame

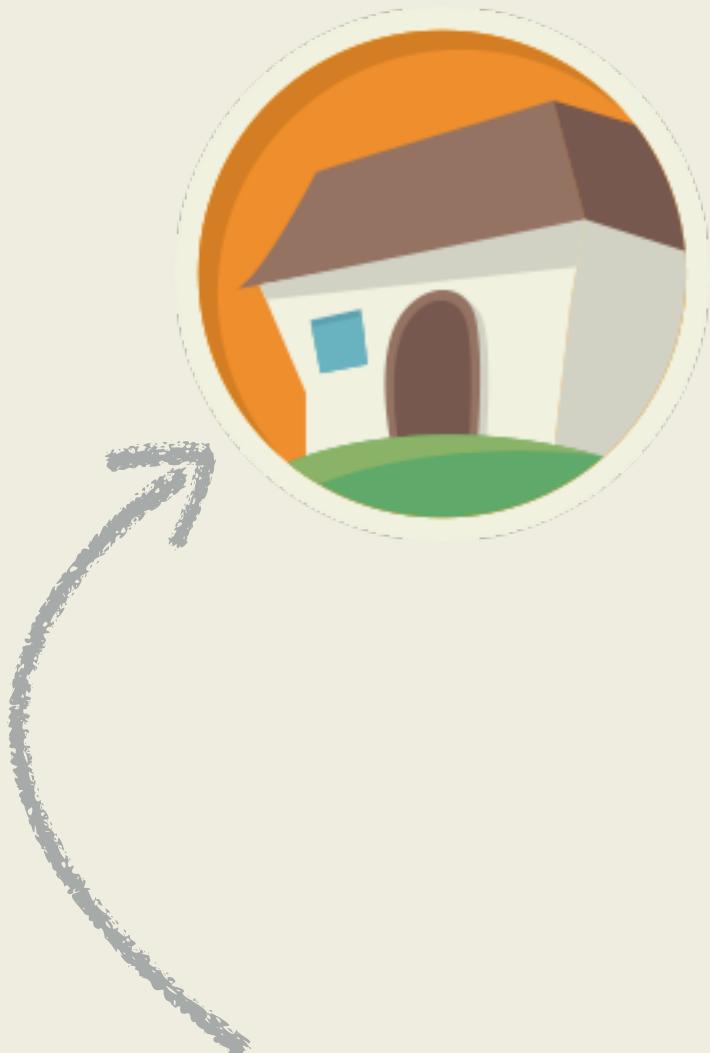
Rendering the products template

Credits

The link has a class of active if  
we're within the products route

How can we add some data to this page?

# Ember Route



Each Ember Route will use this icon

Fetches data and passes it to the Ember Controller

Decides on what model to use for the current route

Decides which template to render to the screen

A model could be a JavaScript  
Object or an Array of objects





# Ember Router

Translates a path into a route

*Don't get these confused*



# Ember Route

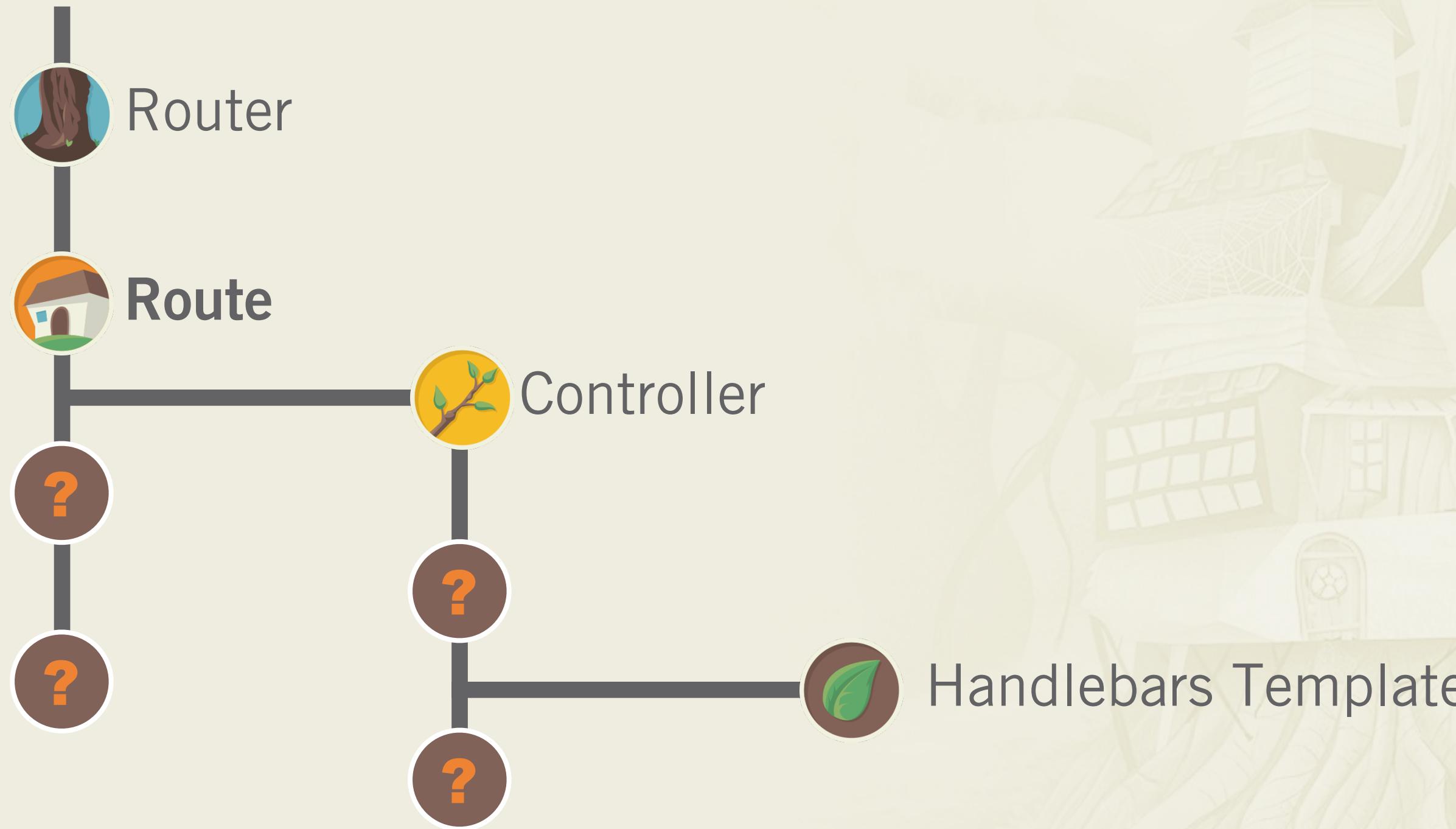
Provides data for the controller



# Ember Route

Provides data for the controller

Browser Request



# Router? Route? Controller?



## Router

Used to define routes our application accepts.

```
this.resource('products');
```



## Route

Responsible for getting data from external resources

```
App.ProductsRoute = Ember.Route.extend({});
```

*created by Ember if not defined*



## Controller

Decorates the model, provides property values

```
App.ProductsController = Ember.Controller.extend({});
```

*created by Ember if not defined*



# A Very Basic Model

app.js

```
App.PRODUCTS = [
  {
    title: 'Flint',
    price: 99,
    description: 'Flint is...',
    isOnSale: true,
    image: 'flint.png'
  },
  {
    title: 'Kindling',
    price: 249,
    description: 'Easily...',
    isOnSale: false,
    image: 'kindling.png'
  }
];
```

app.js

```
App.ProductsRoute = Ember.Route.extend({
  model: function() {
    return App.PRODUCTS;
  }
});
```

The model property should return an object or an array.

This could be pulled from an API



# Putting it All Together



Router



Route

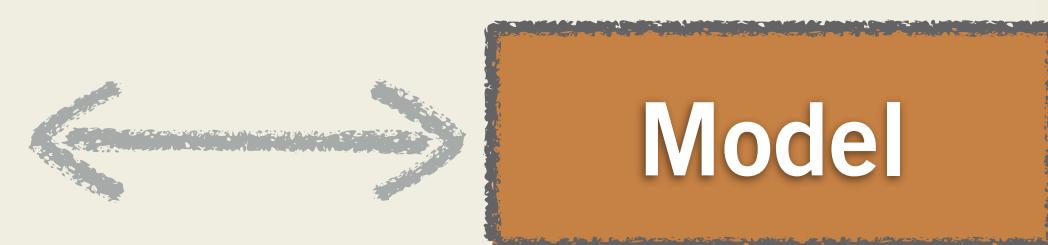


Controller



Template

The Route is responsible for fetching the model. Ember will set this model on the Controller, and that's what'll be used in your Handlebars Template.



index.html

```
<script type='text/x-handlebars' data-template-name='products'>
  <h2>Products</h2>
  {{model}}
</script>
```



# Rendering the Model

index.html

```
<script type='text/x-handlebars' data-template-name='products'>
  <h2>Products</h2>
  {{model}}
</script>
```

  
**<h2>Products</h2>**  
[object Object], [object Object]

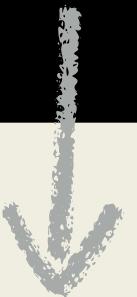
Model is an array, so we'll need to loop over it!



# Looping Over the Array of Objects

index.html

```
<script type='text/x-handlebars' data-template-name='products'>
  <h2>Products</h2>
  {{#each product in model}}
    <h2>{{product.title}}</h2>
  {{/each}}
</script>
```



```
<h1>Products</h1>
<h2>Flint</h2>
<h2>Kindling</h2>
```



# A Simpler #each Loop

#each without extra options will look for a model property.

index.html

```
<script type='text/x-handlebars' data-template-name='products'>
...
{{#each}}
  <h2>{{title}}</h2>
{{/each}}
</script>
```



Within the #each loop, you can access properties on that product



# A Dash of Style

index.html

```
<script type='text/x-handlebars' data-template-name='products'>
<h1>Products</h1>
<ul class='list-unstyled col-md-8'>
{{#each}}
<li class='row'>
  <img {{bind-attr src='image'}} class='img-thumbnail col-md-5'>
  <div class='col-md-7'>
    <h2>{{title}}</h2>
    <p class='product-description'>{{description}}</p>
    <p><button class='btn btn-success'>Buy for ${{price}}</button></p>
  </div>
</li>
{{/each}}
</ul>
</script>
```



Output a few more properties and add some styling



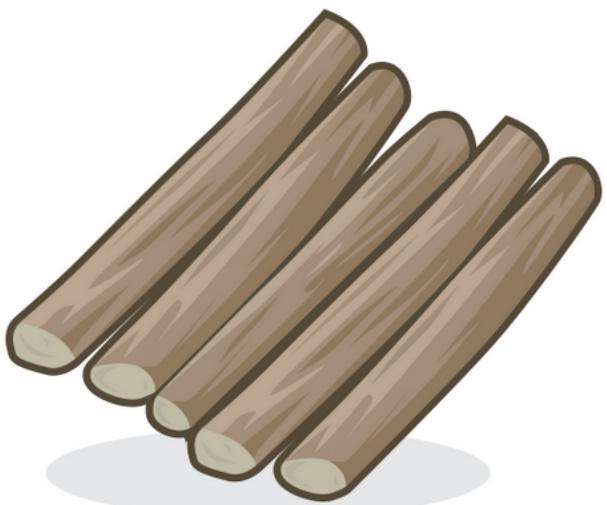


# Products



## Flint

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

[Buy for \\$99](#)

## Kindling

Easily combustible small sticks or twigs used for starting a fire.

[Buy for \\$249](#)

# Warming Up With ember.js

## Level 3 - A Route Through the Woods

Related and Dynamic Routes



# Review

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products');  
});  
  
App.ProductsRoute = Ember.Route.extend({  
  model: function() {  
    return App.PRODUCTS;  
  }  
});
```

Added a resource route

```
{  
  title: 'Flint',  
  price: 99,  
  description: 'Flint is...',  
  isOnSale: true,  
  image: 'flint.png'  
}
```

Create a Route to provide the model

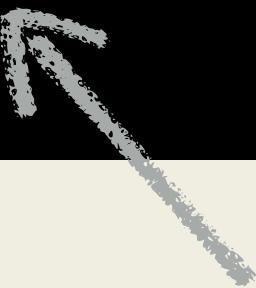
How would we create a page for a specific product? Based on title?



# Adding a Singular Route

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products');  
  this.resource('product');  
});
```



Nothing unique to a specific product in path

We need a path with a parameter.

Maybe #/products/<product\_title>



# Dynamic Segment Routes

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products');  
  this.resource('product', { path: '/products/:title' });  
});
```



<http://example.org#/products/Flint>

```
{ title: 'Flint' }
```

<http://example.org#/products/4>

```
{ title: '4' }
```

Our route object can use the title



# Logging out Params from the Route

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products');  
  this.resource('product', { path: '/products/:title' });  
});
```



```
App.ProductRoute = Ember.Route.extend({  
  model: function(params) {  
    console.log(params);  
  }  
});
```



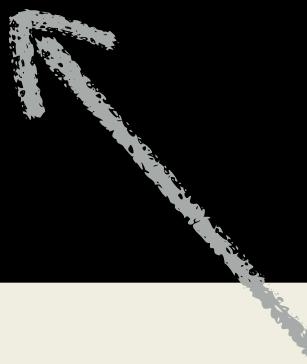
<http://example.org#/products/Flint> { title: 'Flint' }



# Returning a Single Object

app.js

```
App.ProductRoute = Ember.Route.extend({  
  model: function(params) {  
    }  
});
```



We need to lookup a PRODUCT  
with title of params.title



```
App.PRODUCTS = [  
  {  
    title: 'Flint',  
    price: 99,  
    description: 'Flint is...',  
    isOnSale: true,  
    image: 'flint.png'  
  },  
  {  
    title: 'Kindling',  
    price: 249,  
    description: 'Easily...',  
    isOnSale: false,  
    image: 'kindling.png'  
  }  
];
```



# findBy Array Helper

Ember adds helpers to JavaScript base objects, in this case to Array.

```
App.ProductRoute = Ember.Route.extend({
  model: function(params) {
    return App.PRODUCTS.findBy('title', params.title);
  }
});
```

<http://example.org#/products/Flint>

```
{
  title: 'Flint',
  price: 99,
  description: 'Flint is...',
  isOnSale: true,
  image: 'flint.png'
}
```



# Our Product Template

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <h2>{{title}}</h2>
  <p>{{description}}</p>
  <p>Buy for ${{price}}</p>
</script>
```

```
{
  title: 'Flint',
  price: 99,
  description: 'Flint is...',
  isOnSale: true,
  image: 'flint.png'
}
```

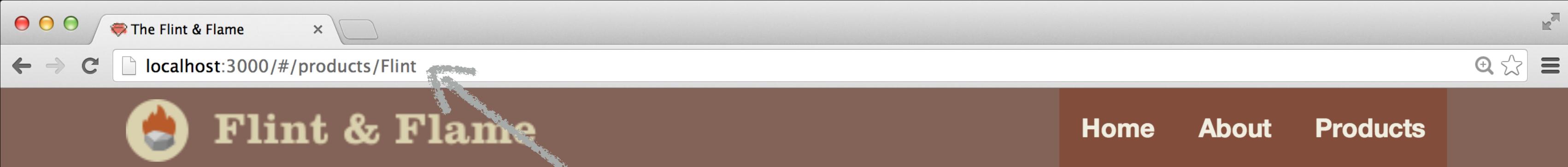
First looks in the controller for properties



Controller

Second looks in the model for properties





# Flint

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

Buy for \$99

---

© 2013 The Flint & Flame

Credits

Our product template is rendered

# Styling Our Product

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='row'>
    <div class='col-md-7'>
      <h2>{{title}}</h2>
      <h3 class='text-success'>${{price}}</h3>
      <p class='text-muted'>{{description}}</p>
    </div>
    <div class='col-md-5'>
      <img {{bind-attr src='image'}} class='img-thumbnail img-rounded'>
    </div>
  </div>
</script>
```

Our more detailed view of a product



# Linking To a Product

index.html

```
<script type='text/x-handlebars' data-template-name='products'>
{{each}}
  {{#link-to 'product' this classNames='list-group-item'}}
    {{title}}
  {{/link-to}}
{{/each}}
</script>
```



Pass in the product we want to link to



The link-to helper knows to use the **this.title** value in the URL since we defined our route with **:title** in the Router.

```
this.resource('product', { path: '/products/:title' });
```



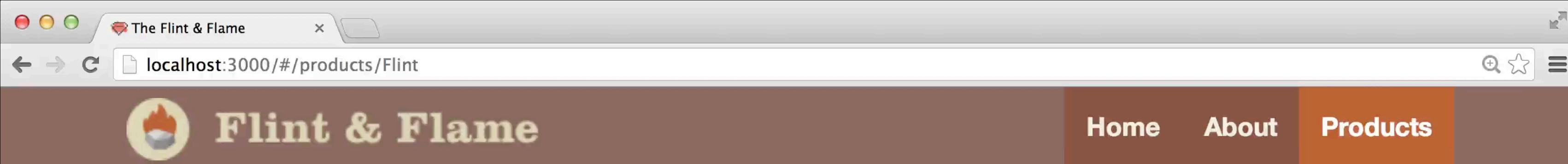
Thus....

```
{{#link-to 'product' this}}
```



/products/Flint





# Flint & Flame

[Home](#)[About](#)[Products](#)

Flint

Kindling

## Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



# Warming Up With ember.js

## Level 3 - A Route Through the Woods

Nested Routes



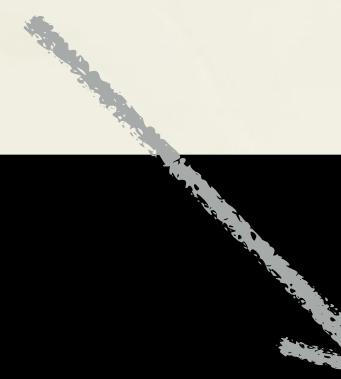
# Review

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products');  
  this.resource('product', { path: '/products/:title' });  
});
```

```
App.ProductRoute = Ember.Route.extend({  
  model: function(params) {  
    return App.PRODUCTS.findBy('title', params.title);  
  }  
});
```

Resource route with a  
dynamic segment



Provide the singular product for the product route





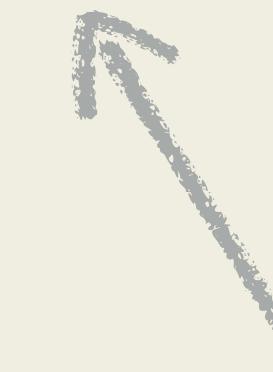
## Products



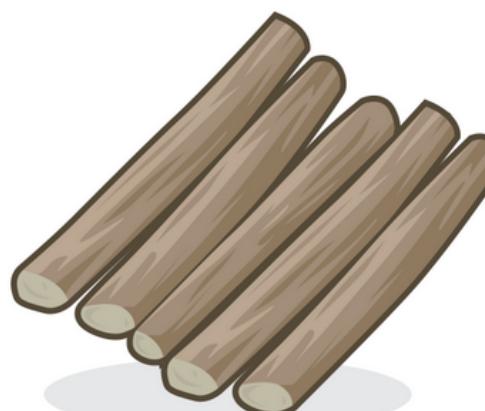
### Flint

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

[Buy for \\$99](#)



products template  
rendering within {{outlet}}  
in the application template



### Kindling

Easily combustible small sticks or twigs used for starting a fire.

[Buy for \\$249](#)



## Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



---

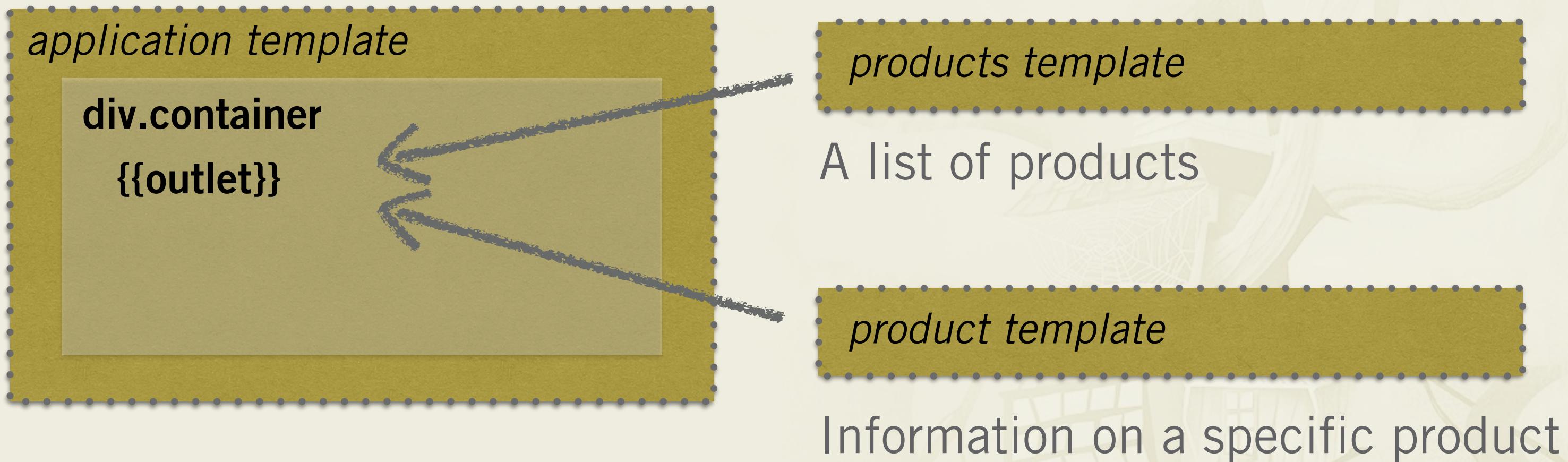
© 2013 The Flint & Flame

product template also  
rendering within {{outlet}}  
in the application template

Credits



# Our Existing Templates



They both are rendered in the same `{outlet}`, but what if we wanted to do something different?



The Flint & Flame

localhost:3000/#/products/Flint

# Flint & Flame

Home About Products

Flint

Kindling

# Flint

\$99

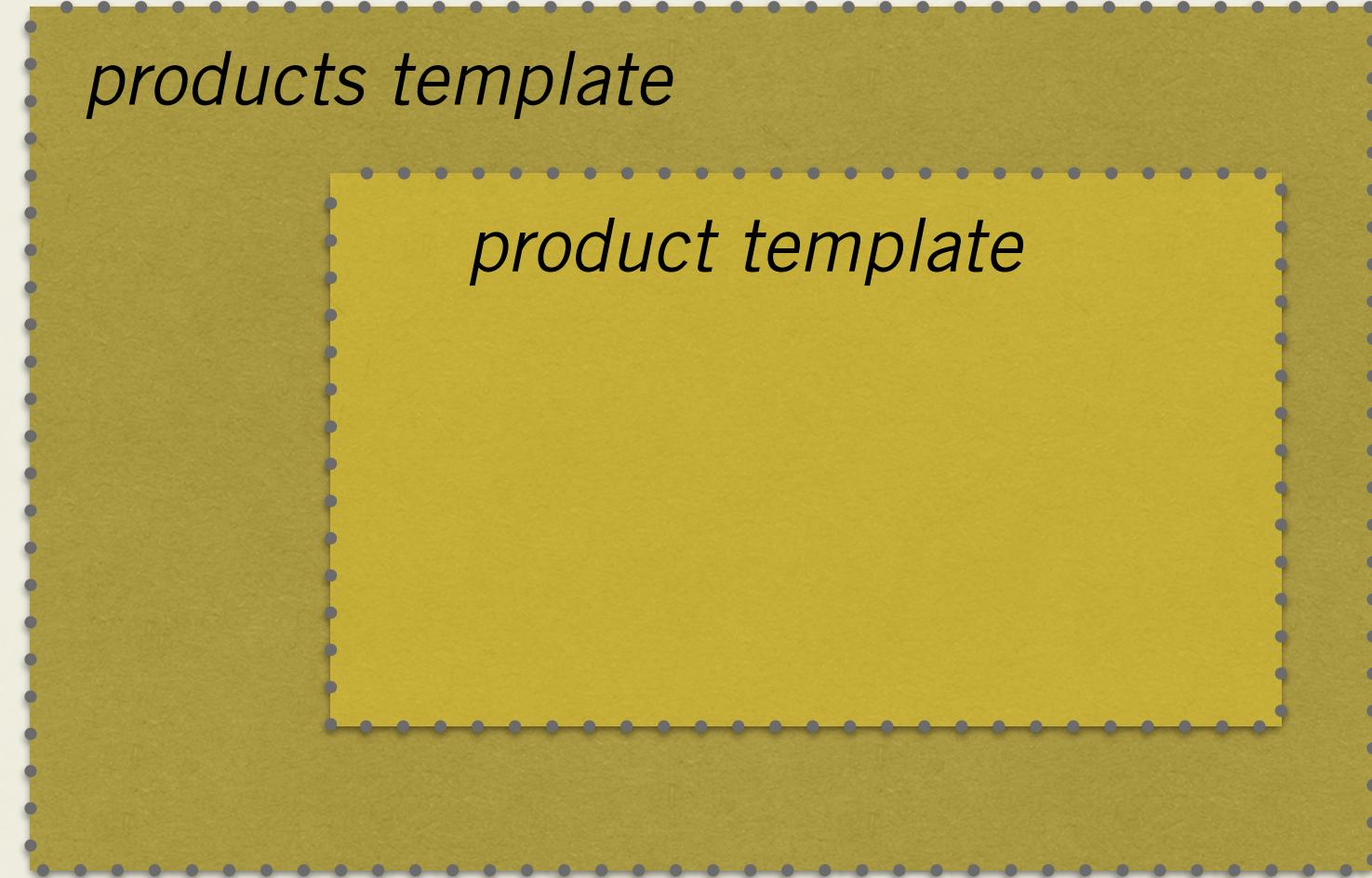
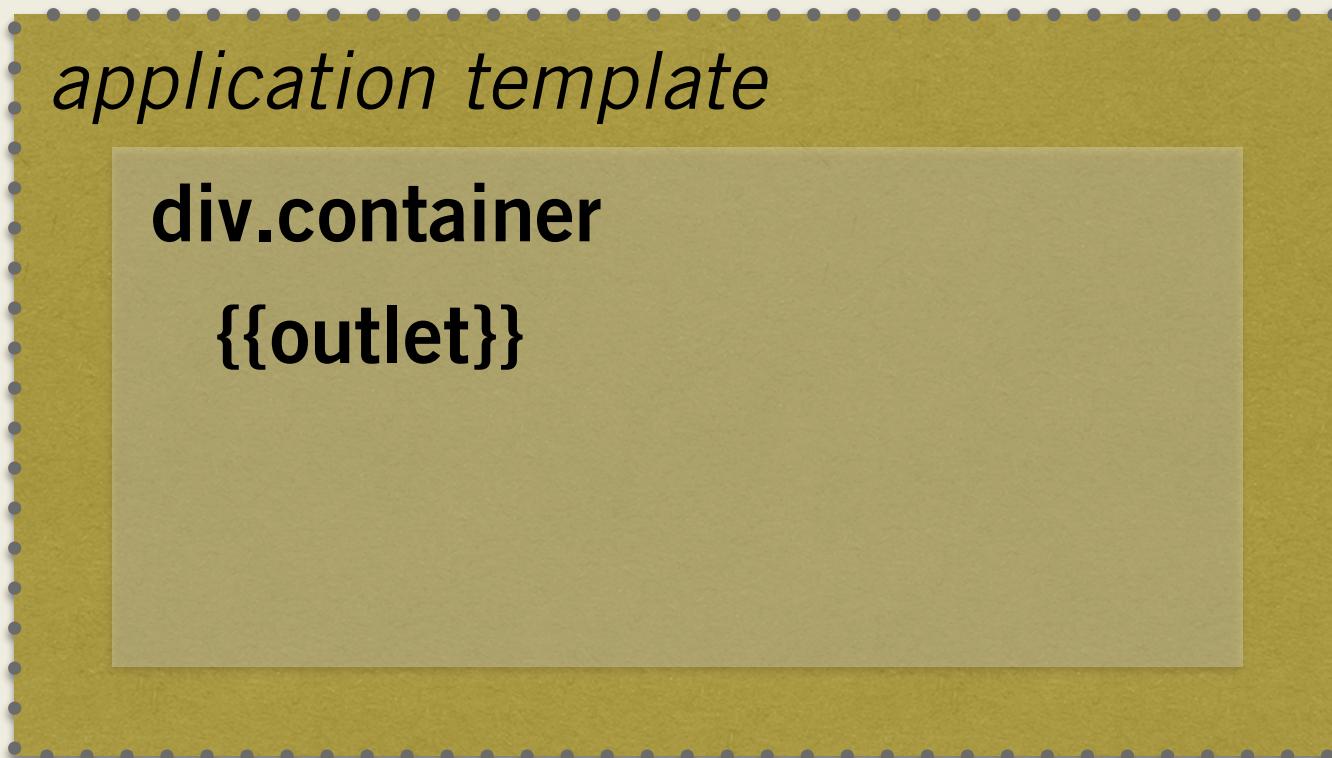
Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



© 2013 The Flint & Flame

Credits

# Template in a Template



How do we nest the **product** template  
inside the **products** template?



# If your Views are nested your Routes should be nested

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products');  
  this.resource('product', { path: '/products/:title' });  
});
```

Old Router



Nest the resources

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products', function() {  
    this.resource('product', { path: '/:title' });  
  });  
});
```



Notice we don't need /products anymore.



# Following the Path

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products', function() {  
    this.resource('product', { path: '/:title' });  
  });  
});
```



**Ember Path**    products/:title

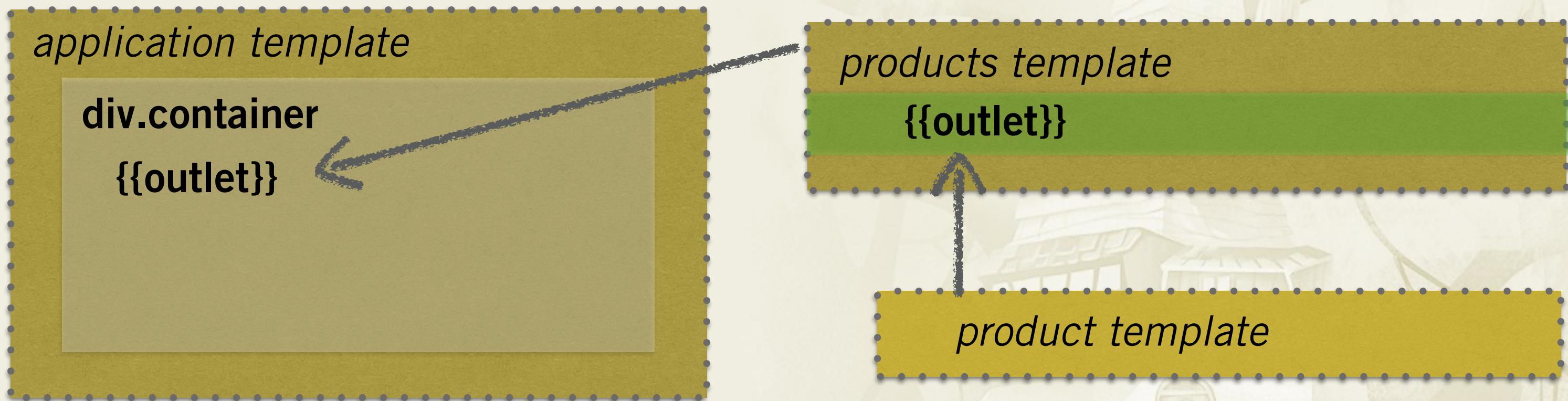
**Route**    product

**URL**    <http://example.com#/products/Flint>



# Adding an Outlet

We need an outlet to show where the product template will be rendered inside the products template.

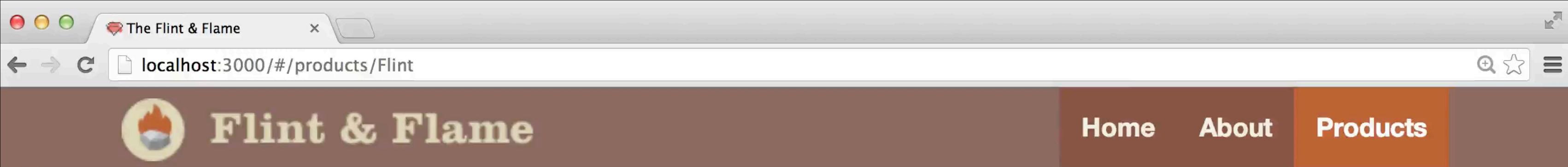


# Adding the Outlet to our Template

index.html

```
<script type='text/x-handlebars' data-template-name='products'>
<div class='row'>
  <div class='col-sm-3'>
    <div class='list-group'>
      {{#each}}
        {{#link-to 'product' this classNames='list-group-item'}}
          {{title}}
        {{/link-to}}
      {{/each}}
    </div>
  </div>
  <div class='col-sm-9'>
    {{outlet}}
  </div>
</div>
</script>
```





# Flint & Flame

[Home](#)[About](#)[Products](#)

Flint

Kindling

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



# Ember Inspector For Chrome

<https://chrome.google.com/webstore/detail/ember-inspector/bmdblncegkenkacieihfhpjfppoconhi>



The Flint & Flame

localhost:3000/#/products/Flint

Home About Products



# Flint & Flame

Flint

Kindling

## Flint

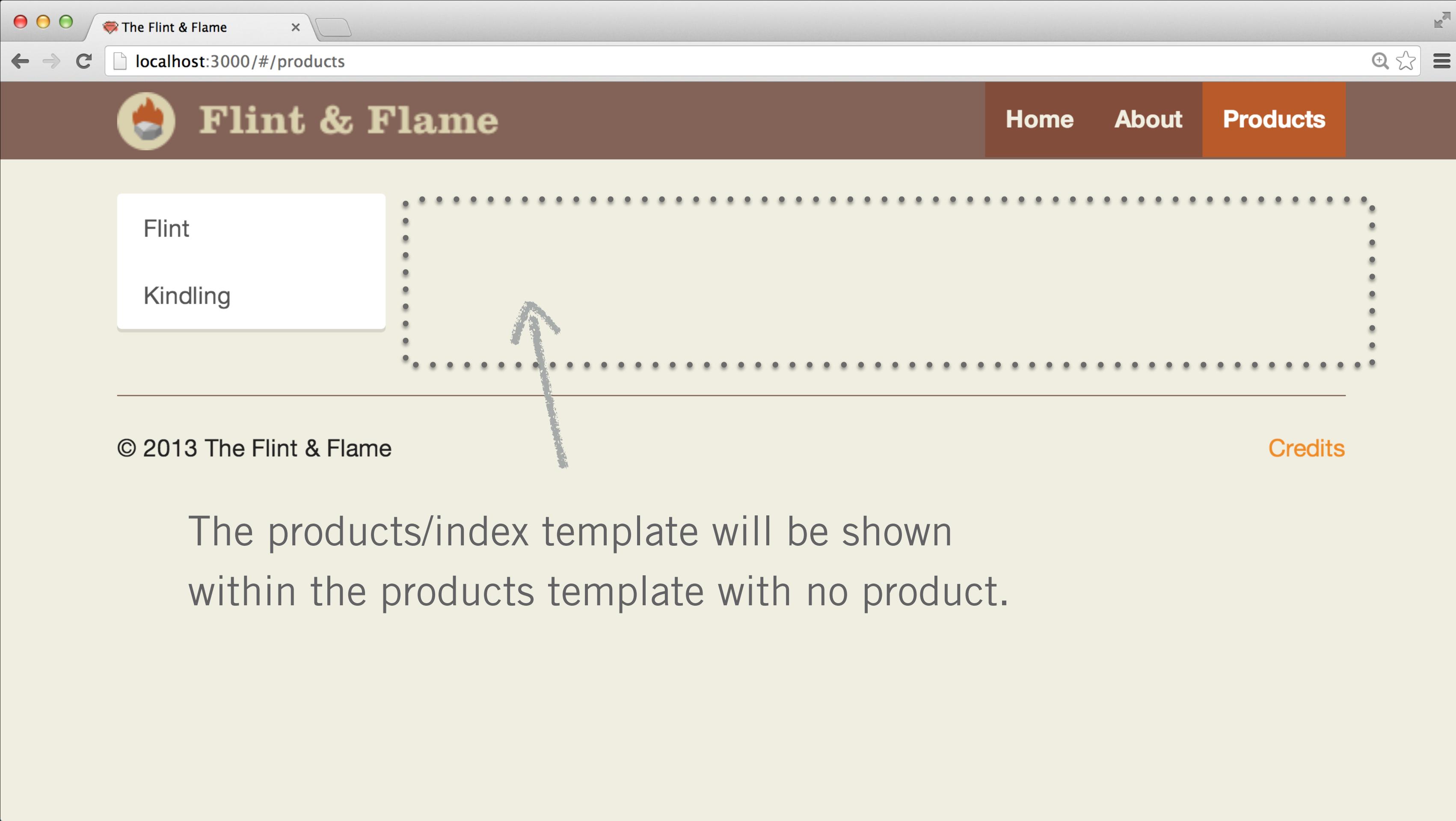
\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



© 2013 The Flint & Flame

Credits



The products/index template will be shown  
within the products template with no product.

# Template for products/index Route

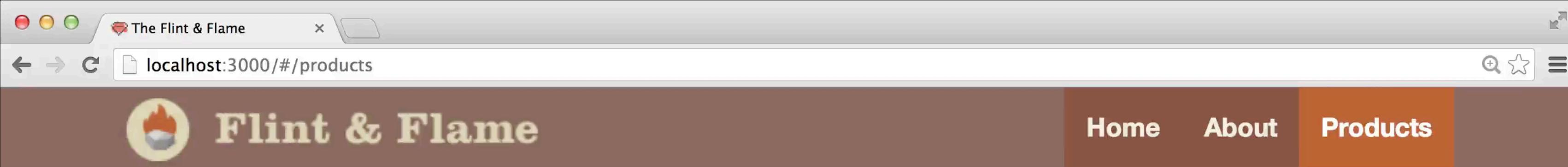
index.html

```
<script type='text/x-handlebars' data-template-name='products/index'>
  <p class='text-muted'>Choose a product from those on the left!</p>
</script>
```



Will be filled into the {{outlet}} of the products template





# Flint & Flame

[Home](#)[About](#)[Products](#)[Flint](#)[Kindling](#)

Choose a product from those on the left!

© 2013 The Flint & Flame

[Credits](#)

# Warming Up With ember:js

## Level 4 - Acorn Models and Pinecone Data

Models and Ember Data



# Review

What if we wanted to fetch products from a server?

app.js

```
App.PRODUCTS = [...];

App.ProductsRoute = Ember.Route.extend({
  model: function() {
    return App.PRODUCTS;
  }
});

App.ProductRouter = Ember.Route.extend({
  model: function(params) {
    return App.PRODUCTS.findBy('title', params.title);
  }
});
```



# Ember Model



A class that defines the properties and behavior of the data that you present to the user.

Every Route can have a Model.

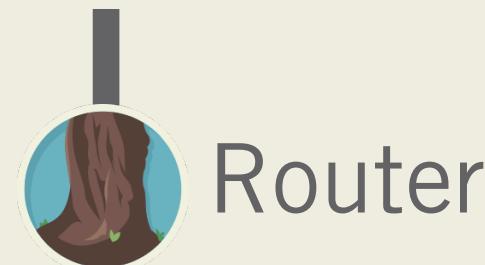
Previously our route set the model to an Array.

This is our Model icon



# Ember Model

Browser Request



A class that defines the properties and behavior of the data that you present to the user.

Controller

?

?

Handlebars Template



# Creating an Ember Model

We are working with products, so lets define a product class.

*Models are typically nouns.*

app.js

```
App.Product = DS.Model.extend({ });
```



Next, our Ember Model needs to know what attributes it contains.



# Ember Model Attributes

app.js

```
App.Product = DS.Model.extend({ });
```



We need to know what data types our Model should expect.

string

number

boolean

date

Remember our data?

```
{  
  title: 'Flint',  
  price: 99,  
  description: 'Flint is...',  
  isOnSale: true,  
  image: 'flint.png'  
}
```



# Building the Product Model

We'll define all the different properties in our model class.

app.js

```
App.Product = DS.Model.extend({
  title: DS.attr('string'),
  price: DS.attr('number'),
  description: DS.attr('string'),
  isOnSale: DS.attr('boolean'),
  image: DS.attr('string')
});
```



```
{
  title: 'Flint',
  price: 99,
  description: 'Flint is...',
  isOnSale: true,
  image: 'flint.png'
}
```



# Implied Data Types

If property types aren't supplied, they will be implied.

app.js

```
App.Product = DS.Model.extend({  
  title: DS.attr(),  
  price: DS.attr(),  
  description: DS.attr(),  
  isOnSale: DS.attr(),  
  image: DS.attr()  
});
```



```
{  
  title: 'Acorn',  
  price: 99,  
  description: 'These...',  
  isOnSale: true,  
  isSeasonal: true  
}
```



# Ember Data



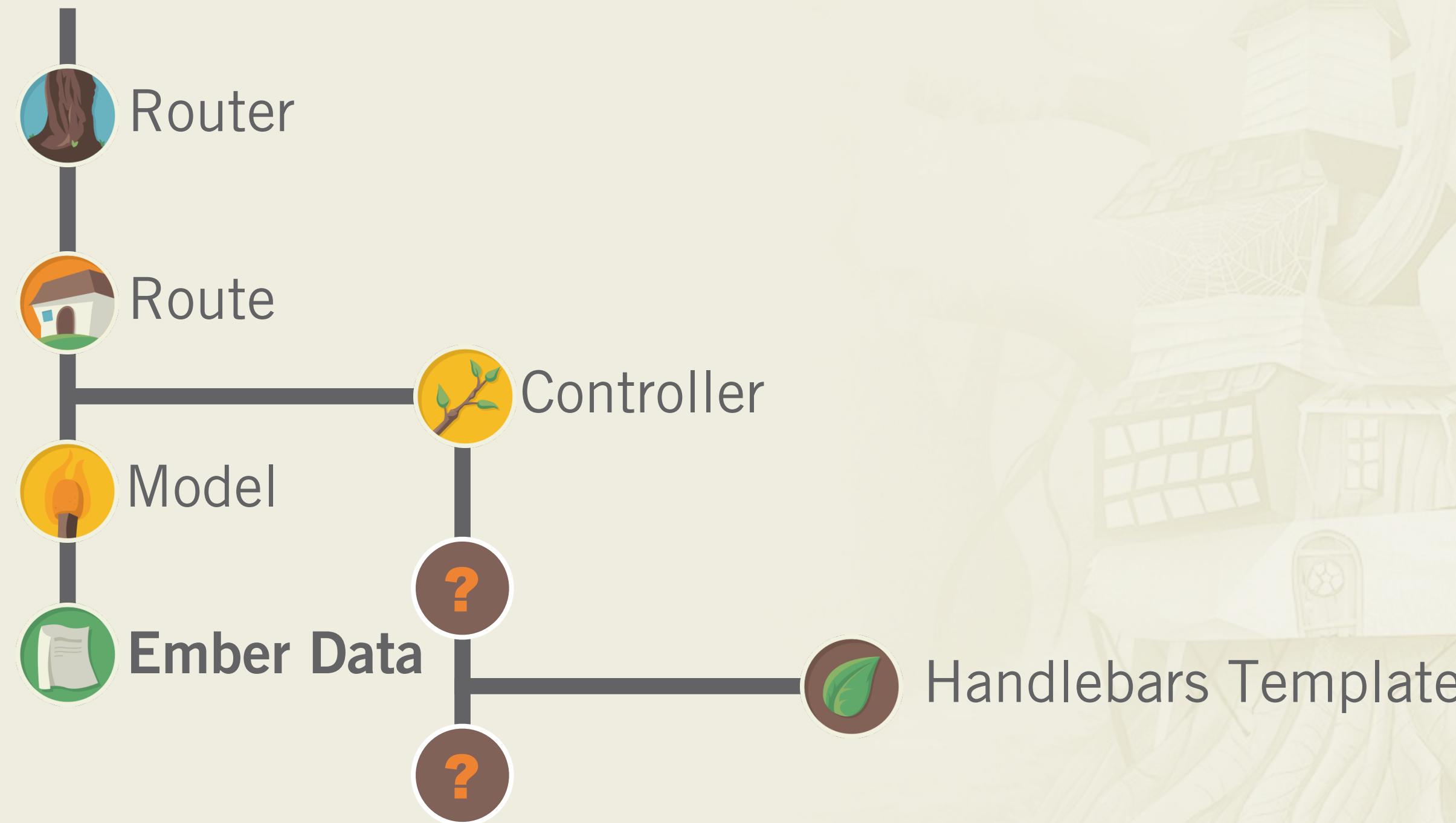
Ember Data makes it easy to use a Model to retrieve records from a server, cache them for performance, save updates back to the server, and create new records on the client.

This is our Ember Data icon



# Ember Data

Browser Request



# Ember Data Adapters

To communicate with an HTTP server using JSON

```
App.ApplicationAdapter = DS.RESTAdapter.extend();
```



This is the default adapter.

To load records from memory

```
App.ApplicationAdapter = DS.FixtureAdapter.extend();
```



Allows us to hardcode our data in fixtures for getting started.



# Migrating to Fixtures

app.js

```
App.PRODUCTS = [
  {
    title: 'Flint',
    price: 99,
    description: 'Flint is...',
    isOnSale: true,
    image: 'flint.png'
  },
  {
    title: 'Kindling',
    price: 249,
    description: 'Easily...',
    isOnSale: false,
    image: 'kindling.png'
  }
];
```



We'll need to convert this to use Ember Data FixtureAdapter.



# Ember Data Fixtures for Product

app.js

```
App.Product.FIXTURES = [
  { id: 1,
    title: 'Flint',
    price: 99,
    description: 'Flint is...',
    isOnSale: true,
    image: 'flint.png'
  },
  { id: 2, ←
    title: 'Kindling',
    price: 249,
    description: 'Easily...',
    isOnSale: false,
    image: 'kindling.png'
  }
];
```



Needs to use the FIXTURES constant within the model

Need to give each product a unique ID



# Ember Data Has A Store

Central repository for records in your application, available in routes and controllers.

Think of it as a cache storage of all your records.



We'll use the store to retrieve records and create new ones.



# Changing :title to :product\_id

app.js

```
this.resource('products', function() {  
  this.resource('product', { path: '/:title' });  
});
```



Ember Data (by default) must use  
a unique identifier. We'll use :product\_id.

```
this.resource('products', function() {  
  this.resource('product', { path: '/:product_id' });  
});
```



Switch to using product\_id as the dynamic segment



# Need to Update Our Routes

app.js

```
App.ProductsRoute = Ember.Route.extend({  
  model: function() {  
    return App.PRODUCTS;  
  }  
});
```



In order to get our fixture data out of the store

```
App.ProductsRoute = Ember.Route.extend({  
  model: function() {  
    return this.store.findAll('product');  
  }  
});
```



Finds all products from the fixture adapter



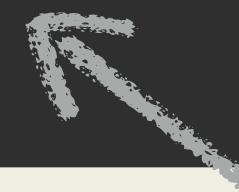
# Updating the Product Route

app.js

```
App.ProductRoute = Ember.Route.extend({  
  model: function(params) {  
    return App.PRODUCTS.findBy('title', params.title);  
  }  
});
```



```
App.ProductRoute = Ember.Route.extend({  
  model: function(params) {  
    return this.store.find('product', params.product_id);  
  }  
});
```



Finds only the product with the matching ID



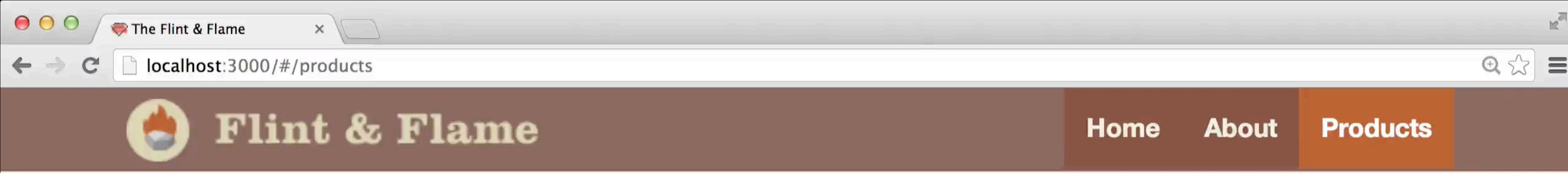
# Using the Default Product Route

```
App.ProductRoute = Ember.Route.extend({  
  model: function(params) {  
    return this.store.find('product', params.product_id);  
  }  
});
```



We can delete the ProductRoute and use the default!





Flint

Kindling

Choose a product from those on the left!

© 2013 The Flint & Flame

Credits

# Warming Up With ember:js

## Level 4 - Acorn Models and Pinecone Data

Related models



# Review

app.js

```
App.Product = DS.Model.extend({  
  title: DS.attr('string'),  
  price: DS.attr('number'),  
  description: DS.attr('string'),  
  isOnSale: DS.attr('boolean'),  
  image: DS.attr('string')  
});
```

```
App.ProductsRoute = Ember.Route.extend({  
  model: function() {  
    return this.store.findAll('product');  
  }  
});
```

What if our products had reviews?



# The Review Model

First lets create a new model to represent a Review.

app.js

```
App.Product = DS.Model.extend({
  title: DS.attr('string'),
  price: DS.attr('number'),
  description: DS.attr('string'),
  isOnSale: DS.attr('boolean'),
  image: DS.attr('string')
});
```

```
App.Review = DS.Model.extend({
  text: DS.attr('string'),
  reviewedAt: DS.attr('date')
});
```



# Create Associations

app.js

```
App.Product = DS.Model.extend({
  title: DS.attr('string'),
  price: DS.attr('number'),
  description: DS.attr('string'),
  isOnSale: DS.attr('boolean'),
  image: DS.attr('string')
  reviews: DS.hasMany('review', {async: true})
});
```

```
App.Review = DS.Model.extend({
  text: DS.attr('string'),
  reviewedAt: DS.attr('date')
  product: DS.belongsTo('product')
});
```



Allows reviews to be lazy loaded



# Creating our Review Fixtures

app.js

```
App.Review.FIXTURES = [  
  { id: 100,  
    text: "Started a fire in no time!"  
  },  
  { id: 101,  
    text: "Not the brightest flame, but warm!"  
  }  
];
```

We'll use higher IDs so we can differentiate them



# Mapping Reviews to a Product

app.js

```
App.Review.FIXTURES = [
  { id: 100,
    product: 1,
    text: "Started a fire in no time!"
  },
  { id: 101,
    product: 1,
    text: "Not the brightest flame, but warm!"
  }
];
```



# Mapping a Product to Reviews

app.js

```
App.Product.FIXTURES = [
  { id: 1,
    title: 'Flint',
    price: 99,
    description: 'Flint is...',
    isOnSale: true,
    image: 'flint.png',
    reviews: [100, 101]
  }
];
```



Need to use an array to list out all reviews

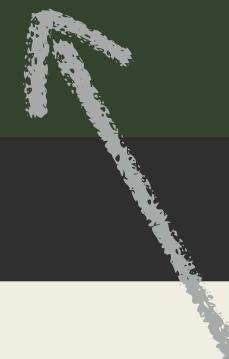


# Loop Over All Reviews

index.html

```
<script type="text/x-handlebars" data-template-name='product'>
  <h1>{{title}}</h1>
  <p>{{description}}</p>
  <p>Buy for ${{price}}</p>

  <h3>Reviews</h3>
  <ul>
    {{#each review in reviews}}
      <li>{{review.text}}</li>
    {{/each}}
  </ul>
</script>
```



Loop over all reviews and display them

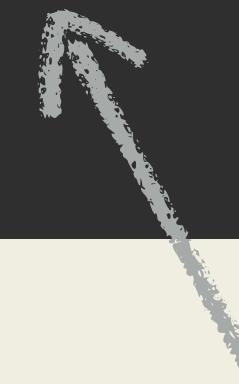


# Easier Loop Variables

index.html

```
<script type="text/x-handlebars" data-template-name='product'>
  <h1>{{title}}</h1>
  <p>{{description}}</p>
  <p>Buy for ${{price}}</p>

  <h3>Reviews</h3>
  <ul>
    {{#each reviews}}
      <li>{{text}}</li>
    {{/each}}
  </ul>
</script>
```



Within the loop, you can  
reference the current review



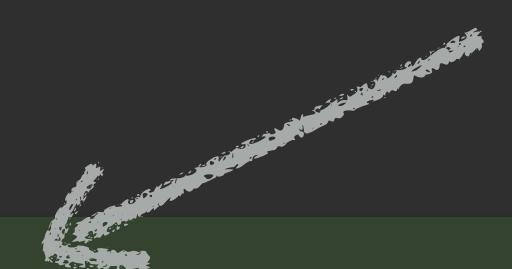
# Loops With {{else}}

index.html

```
<script type="text/x-handlebars" data-template-name='product'>
  <h1>{{title}}</h1>
  <p>{{description}}</p>
  <p>Buy for ${{price}}</p>

  <h3>Reviews</h3>
  <ul>
    {{#each reviews}}
      <li>{{text}}</li>
    {{else}}
      <li><p class='text-muted'>
        <em>No reviews yet. Be the first to write one!</em>
      </p></li>
    {{/each}}
  </ul>
</script>
```

Add a default for when there  
are no reviews





Flint

Kindling

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



## Reviews

- Started a fire in no time!
- Not the brightest flame, but warm!



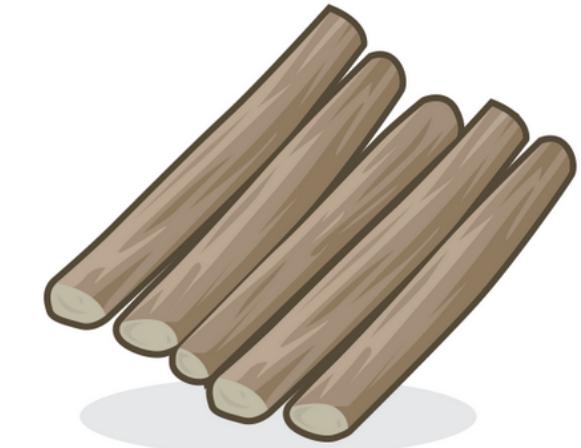
Flint

Kindling

## Kindling

\$249

Easily combustible small sticks or twigs used for starting a fire.



## Reviews

- *No reviews yet. Be the first to write one!*

# Switching to REST

```
App.ApplicationAdapter = DS.RESTAdapter.extend();
```



Let's switch our application adapter back to REST



The Flint & Flame

localhost:3000/#/products/

Ember Data hit the /products URL to get JSON data for our products route

Elements Resources Network Sources Timeline Profiles »

ember-1.0.0.js?body=1:394

ember-1.0.0.js?body=1:394

ember-1.0.0.js?body=1:394

DEBUG: jQuery.VERSION : 1.10.2

DEBUG: -----

GET http://localhost:3000/products 404 (Not Found)

<top frame>

<pa 3 ! 11

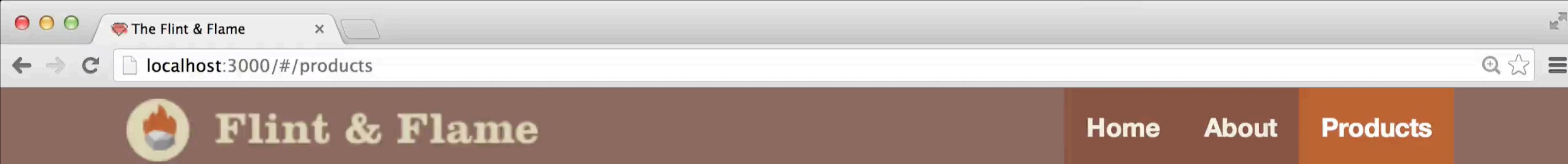
# Asynchronous JSON

http://example.com/products

```
{ "products": [  
    {  
        "id": 1,  
        "title": "Flint",  
        "price": 99,  
        "description": "Flint is...",  
        "isOnSale": true,  
        "image": "/assets/products/flint.png",  
        "reviews": [100, 101]  
    }, { ... }  
],  
"reviews": [  
    { "id": 100, "product": 1, "text": "Started a fire in no time!" }, ...  
]
```

This JSON file could be  
generated or hardcoded





# Flint & Flame

[Home](#)[About](#)[Products](#)[Flint](#)[Kindling](#)

Choose a product from those on the left!



© 2013 The Flint & Flame

Credits

[Elements](#) [Resources](#) [Network](#) [Sources](#) [Timeline](#) [Profiles](#)

► XHR finished loading: "<http://localhost:3000/products>".  
jquery.js?body=1:8707



<top frame>



<page

⚠ 10



# Warming Up With ember.js

## Level 5 - Controlling our Growth

Array Controllers



# Review

app.js

```
App.ProductsRoute = Ember.Route.extend({  
  model: function() {  
    return this.store.findAll('product');  
  }  
});
```



What about Sorting or Filtering?





Flint

Kindling

Matches

Bow Drill

Tinder

Birch Bark Shaving

Choose a product from those on the left!



How could we sort products?

# Sorting Models By Title

app.js

```
App.ProductsRoute = Ember.Route.extend({  
  model: function() {  
    return this.store.find('product', { order: 'title' });  
  }  
});
```



Tells our store how we want the product sorted



<http://example.com/products?order=title>



# Sorting at the Server vs Client

To have the server do the sorting

```
this.store.find('product', { order: 'title'});
```

This will send our browser products already sorted.

To have the client (or browser) do the sorting

we will have to sort in the **controller**.

**Controller**



Decorate your applications data for the template.



# We Need a Special Controller

app.js

```
App.ProductsController = Ember.Controller.extend({});
```



We need a controller that knows how to deal with Arrays of objects.



# EmberController Variants

Ember.Controller

Ember.ArrayController

Decorates an Array

Ember.ObjectController

Decorates a single Object



# Array Controller

If our model is an array, we should use an ArrayController.

app.js

```
App.ProductsController = Ember.ArrayController.extend({});
```

This gives our Controller some special abilities.



# Sorting by a Property

app.js

```
App.ProductsController = Ember.ArrayController.extend({  
  sortProperties: ['title']  
});
```



Sorts by the title of each product (in the browser)



# Sort Direction

By default this will sort ascending

A-Z

```
App.ProductsController = Ember.ArrayController.extend({  
  sortProperties: ['title']  
});
```



To sort descending

```
App.ProductsController = Ember.ArrayController.extend({  
  sortProperties: ['title'],  
  sortAscending: false  
});
```





Birch Bark Shaving

Bow Drill

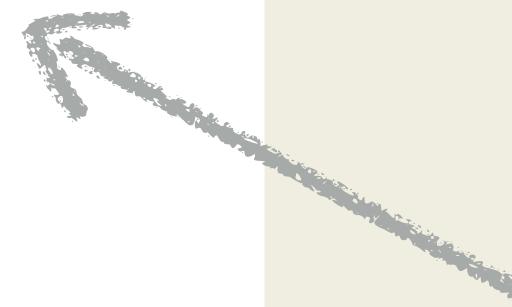
Flint

Kindling

Matches

Tinder

Choose a product from those on the left!



Products are now sorted by name

# Refactoring IndexController

We need to convert this to actually count the length of our models.

app.js

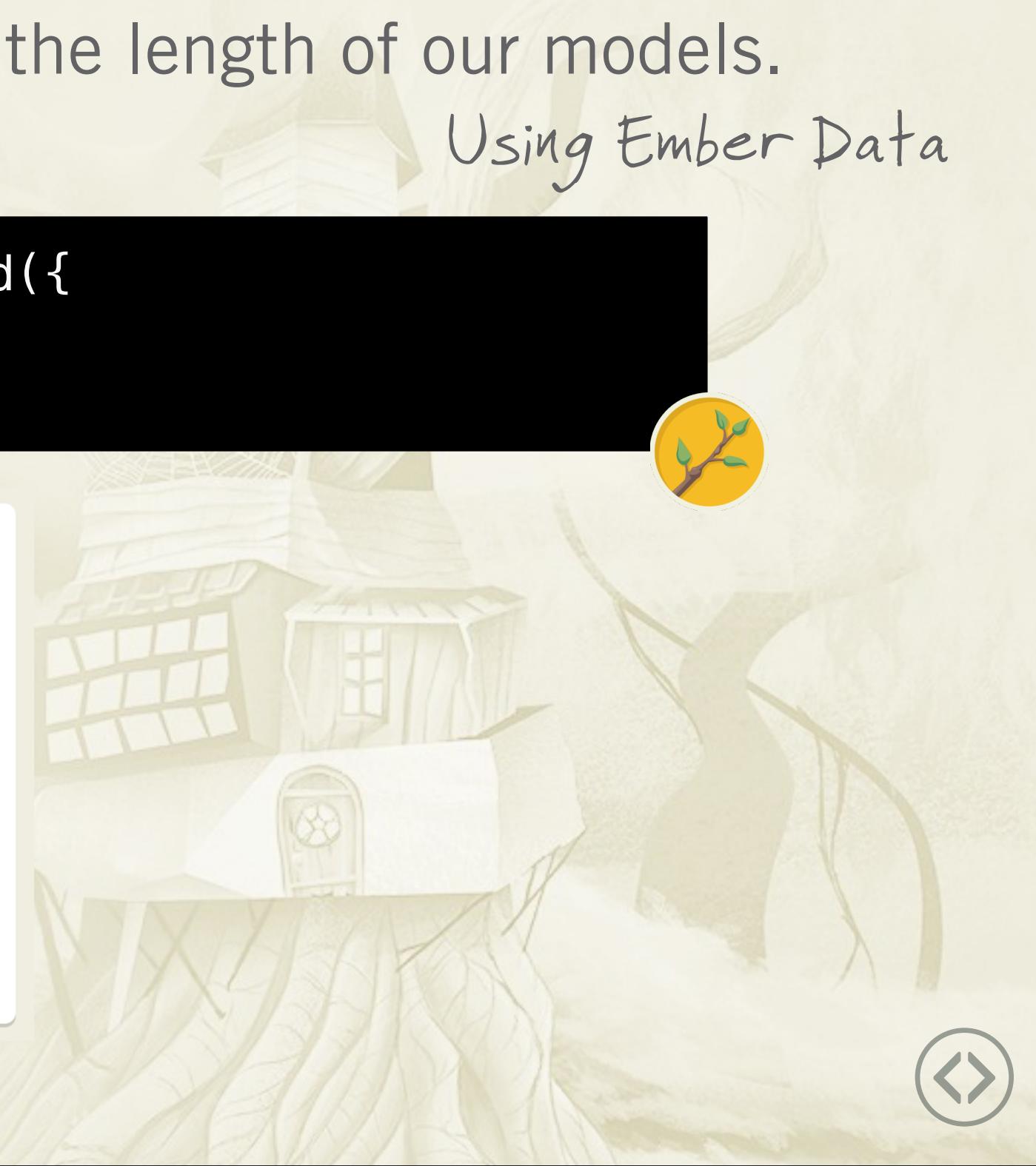
```
App.IndexController = Ember.Controller.extend({  
  productsCount: 6  
});
```

Using Ember Data

Welcome to The Flint & Flame!

 **Flint & Flame** Everything you need to make it through the winter.

[Browse All 6 Items »](#)



# Step 1 - Associate a Model

Set the appropriate model that the IndexController will use.

app.js

```
App.IndexController = Ember.Controller.extend({  
  productsCount: 6  
});
```



```
App.IndexRoute = Ember.Route.extend({  
  model: function() {  
    return this.store.findAll('product');  
  }  
});
```



# Switching Controller Type

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: 6  
});
```

The model is an array



# Computed Properties

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: function() {  
    return 6;  
  }.property()  
});
```



Convert `productsCount` to a function



# Getting the Model Length

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: function() {  
    return this.get('length');  
  }.property()  
});
```

This will first look for a property called length in the ArrayController.

Then it will delegate to the model, and call length on the model.

model.length



# Handlebars Property Binding

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: function() {  
    return this.get('length');  
  }.property('length')  
});
```



This will keep a watch on the 'length' property of the controller, and if it changes update productsCount

index.html

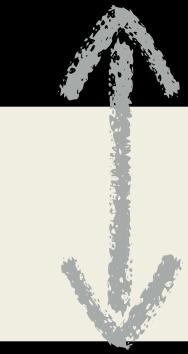
```
<script type='text/x-handlebars' data-template-name='index'>  
  <p>There are {{productsCount}} products</p>  
</script>
```



# Computed Alias

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: function() {  
    return this.get('length');  
  }.property('length')  
});
```



Functionally the same

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: Ember.computed.alias('length')  
});
```



The Flint & Flame

localhost:3000/#/

# Flint & Flame!

 **Flint & Flame** Everything you need to make it through the winter.

[Browse All 6 Items »](#)

Elements Resources Network Sources Timeline Profiles Audits »

> App.Product.store.find('product', 1).then(function(product){});

□ ▷≡ 🔎 ⚔ <top frame> ▼ <page context> 🛡

# Warming Up With ember.js

## Level 5 - Controlling our Growth

Computed Properties



# Review

app.js

```
App.ProductsController = Ember.ArrayController.extend({  
  sortProperties: ['title']  
});
```



Sort the array of products

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: Ember.computed.alias('length')  
});
```



Create a property tied to a method on the model





# Welcome to The Flint & Flame!



**Flint & Flame** Everything you need to make it through  
the winter.

[Browse All 6 Items »](#)



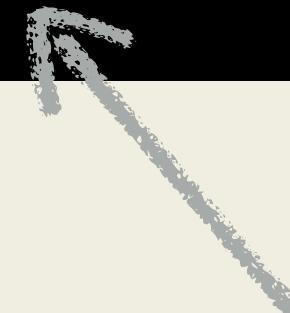
What about showing 3 products  
that are onSale here?

Rendered on Thu Jan 02 2014

# onSale Property

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: Ember.computed.alias('length'),  
  
  onSale: function() {  
    // Return only 3 products that are on sale  
  }.property()  
});
```



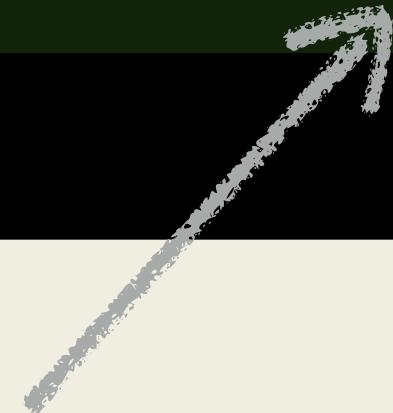
We can filter down the model and use that property in our template



# filter in A Controller

app.js

```
App.IndexController = Ember.ArrayController.extend({
  productsCount: Ember.computed.alias('length'),
  onSale: function() {
    return this.filter(function(product) {
      // Return true if the product is on sale
    });
  }.property()
});
```



Filter loops over all products and returns only those that return true in the function.



# Returning in a filter

app.js

```
App.IndexController = Ember.ArrayController.extend({
  productsCount: Ember.computed.alias('length'),
  onSale: function() {
    return this.filter(function(product) {
      return product.get('isOnSale');
    });
  }.property()
});
```



We have access to the product in the loop,  
so we can use the `isOnSale` property



# filterBy in A Controller

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: Ember.computed.alias('length'),  
  
  onSale: function() {  
    return this.filterBy('isOnSale', true);  
  }.property()  
});
```

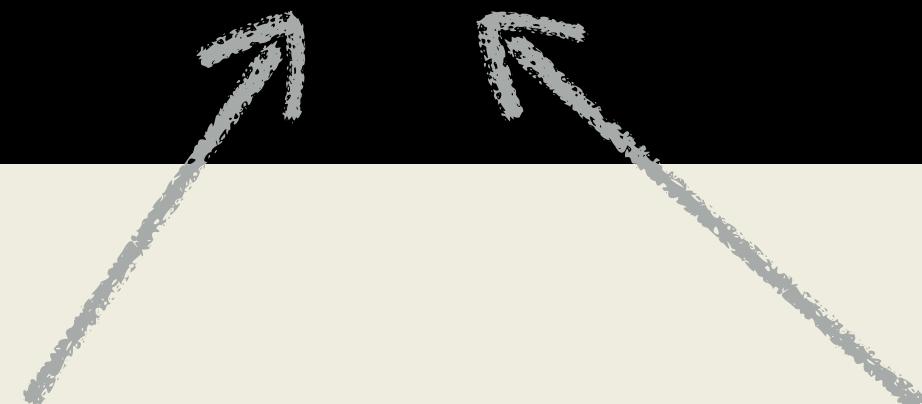
We can filter by a property the same way we filtered by title earlier



# filterBy Defaults

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: Ember.computed.alias('length'),  
  
  onSale: function() {  
    return this.filterBy('isOnSale');  
  }.property()  
});
```



filterBy defaults to true, so  
we can remove that argument

Still need to limit the  
result to 3 products



# Limiting a Result

app.js

```
App.IndexController = Ember.ArrayController.extend({
  productsCount: Ember.computed.alias('length'),
  onSale: function() {
    return this.filterBy('isOnSale').slice(0,3);
  }.property()
});
```

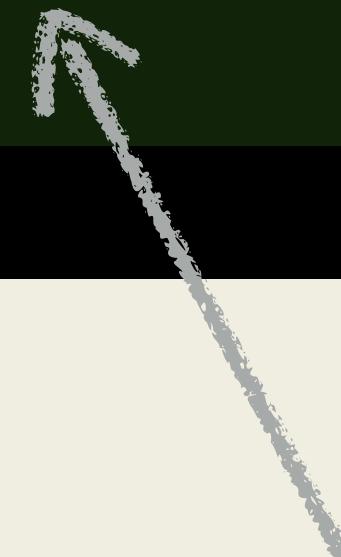
Use JavaScript to limit  
the result to the first 3



# Updating Our index Template

index.html

```
<script type="text/x-handlebars" data-template-name="index">
  ...
  {{#each onSale}}
    {{/each}}
</script>
```



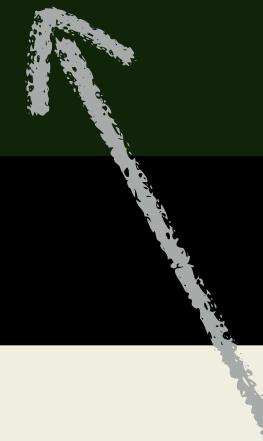
Loop over the 3 products from our the  
onSale property in our IndexController



# Product Details

index.html

```
<script type="text/x-handlebars" data-template-name="index">
  ...
  {{#each onSale}}
    <div class='col-sm-4'>
      <img {{bind-attr src='image'}} class='img-thumbnail col-sm-5' />
      <div class='col-sm-7'>
        <h2>{{title}}</h2>
        <p class="product-description">{{description}}</p>
        <p>{{#link-to 'product' this class='btn btn-success'}}<br/>
          Buy for ${{price}}{{/link-to}}</p>
      </div>
    </div>
  {{/each}}
</script>
```



Give some details about the product



The Flint & Flame

localhost:3000/#/

Browse All Items »



## Flint

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

Buy for \$99



## Matches

One end is coated with a material that can be ignited by frictional heat generated by striking the match against a suitable surface.

Buy for \$499



## Tinder

Tinder is easily combustible material used to ignite fires by rudimentary methods.

Buy for \$499

# Watching for Changes

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: Ember.computed.alias('length'),  
  
  onSale: function() {  
    return this.filterProperty('isOnSale', true).slice(0,3);  
  }.property('@each.isOnSale')  
});
```



Update this property if the `isOnSale` attribute changes on any product





# Flint

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of

short



# Matches

One end is coated with a material that can be ignited by frictional heat generated by striking the match against



# Tinder

Tinder is easily combustible material used to ignite fires by rudimentary methods.

Buy for \$499

Elements Resources Network Sources Timeline Profiles Audits Console Ember

View Tree Routes Data

Model Type

App.Product

»

App.Review

title: Flint

price: 99

description: Flint is a hard, sedimentary c...

isOnSale: true

image: /assets/products/flint.png

\$E

11

# Warming Up With ember.js

## Level 5 - Controlling our Growth

Nested Routes with Controllers



# Review

app.js

```
App.IndexController = Ember.ArrayController.extend({  
  productsCount: Ember.computed.alias('length'),  
  
  onSale: function() {  
    return this.filterBy('isOnSale').slice(0,3);  
  }.property('@each.isOnSale')  
});
```



We're showing 3 products that are on sale on  
the homepage. What about the rest?



# Creating Our Filtered Route

app.js

```
App.Router.map(function() {  
  this.route('about');  
  this.resource('products', function() {  
    this.resource('product', { path: '/:product_id' });  
    this.route('onsale');  
  });  
});
```



onsale is an adjective, so we'll use a route

<http://example.org#/products/onsale>



/products/onsale



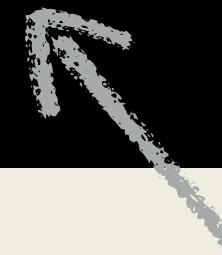
products.onsale



# Nested Route Object

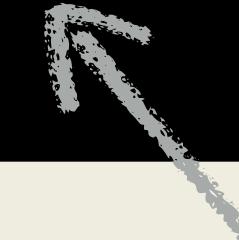
app.js

```
App.OnsaleRoute = Ember.Route.extend({  
});
```



There is no top level route named onsale (it's nested)

```
App.ProductsOnsaleRoute = Ember.Route.extend({  
});
```



Products is the Parent route, so we need to prefix the onsale route



# Model for a Nested Route

app.js

```
App.ProductsRoute = Ember.Route.extend({
  model: function() {
    return this.store.findAll('product');
  }
});
```

The parent route of ProductsOnsale is Products



```
App.ProductsOnsaleRoute = Ember.Route.extend({
  model: function () {
    // Get all products from the parent route and filter them
  }
});
```



# Using the Parent Routes Model

app.js

```
App.ProductsRoute = Ember.Route.extend({  
  model: function() {  
    return this.store.findAll('product');  
  }  
});
```



```
App.ProductsOnsaleRoute = Ember.Route.extend({  
  model: function () {  
    return this.modelFor('products');  
  }  
});
```



modelFor returns the model from the given route



# Using modelFor With Filters

app.js

```
App.ProductsRoute = Ember.Route.extend({
  model: function() {
    return this.store.findAll('product');
  }
});
```



```
App.ProductsOnsaleRoute = Ember.Route.extend({
  model: function () {
    return this.modelFor('products').filterBy('isOnSale');
  }
});
```



The Ember Inspector helps give some details about our new route

| Route Name             | Route               | Controller               | Template        | URL              |
|------------------------|---------------------|--------------------------|-----------------|------------------|
| <b>products.onsale</b> | ProductsOnsaleRoute | ProductsOnsaleController | products/onsale | /products/onsale |

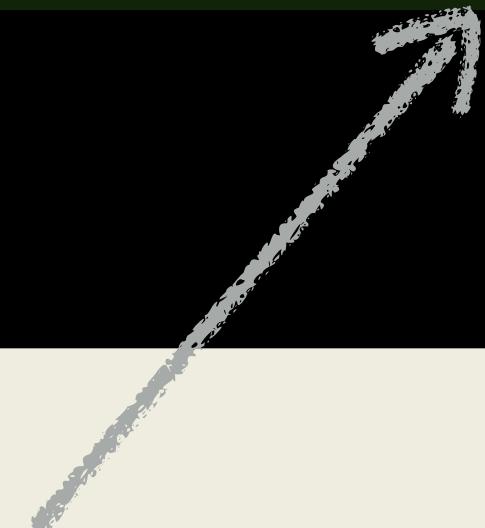
We can link to this route by it's name: products.onsale



# Linking to Our New Route

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  <div class="body-content">
    <div class="row">
      {{#each onSale}}
        ...
        <span class="label label-warning">
          {{#link-to 'products.onSale'}}On Sale{{/link-to}}
        </span>
      {{/each}}
    </div>
  </div>
</script>
```



Use the route name to link to our new route





# Flint

On Sale

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

Buy for \$99



# Matches

On Sale

One end is coated with a material that can be ignited by frictional heat generated by striking the match against a suitable surface.

Buy for \$499



# Tinder

On Sale

Tinder is easily combustible material used to ignite fires by rudimentary methods.

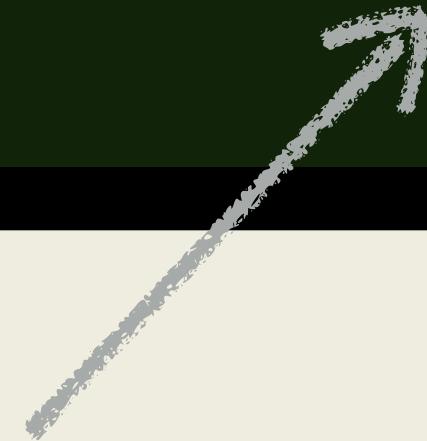
Buy for \$499

Links to our new On Sale Route

# On Sale Products List

index.html

```
<script type="text/x-handlebars" data-template-name="products/onsale">  
/</script>
```



Use the template that matches the route path

**Path** #/products/onsale

**Route Name** products.onsale

**Template Name** products/onsale



# products/onsale Template

index.html

```
<script type="text/x-handlebars" data-template-name="products/onsale">



```

Show all products in the model

Same product details as on the homepage





Birch Bark Shaving

Bow Drill

Flint

Kindling

Matches

Tinder

# On Sale Products



## Flint

On Sale

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

Buy for \$99



## Matches

On Sale

One end is coated with a material that can be ignited by frictional heat generated by striking the match against a suitable surface.

# Warming Up With ember.js

Level 6 - The Template Forest

Components



# Review

app.js

```
App.ProductsRoute = Ember.Route.extend({
  model: function() {
    return this.store.findAll('product');
  }
});
```

```
App.ProductsOnSaleRoute = Ember.Route.extend({
  model: function () {
    return this.modelFor('products').filterBy('isOnSale');
  }
});
```



We're getting products in multiple places



# Review

index.html

```
<script type='text/x-handlebars' data-template-name='products/onsale'>
  {{#each}} ... {{/each}}
</script>

<script type='text/x-handlebars' data-template-name='index'>
  {{#each onSale}} ... {{/each}}
</script>
```

Showing the same details about  
each product in multiple places



# Ember Component



Similar implementation to Web Components

Split out parts of your page to reusable components

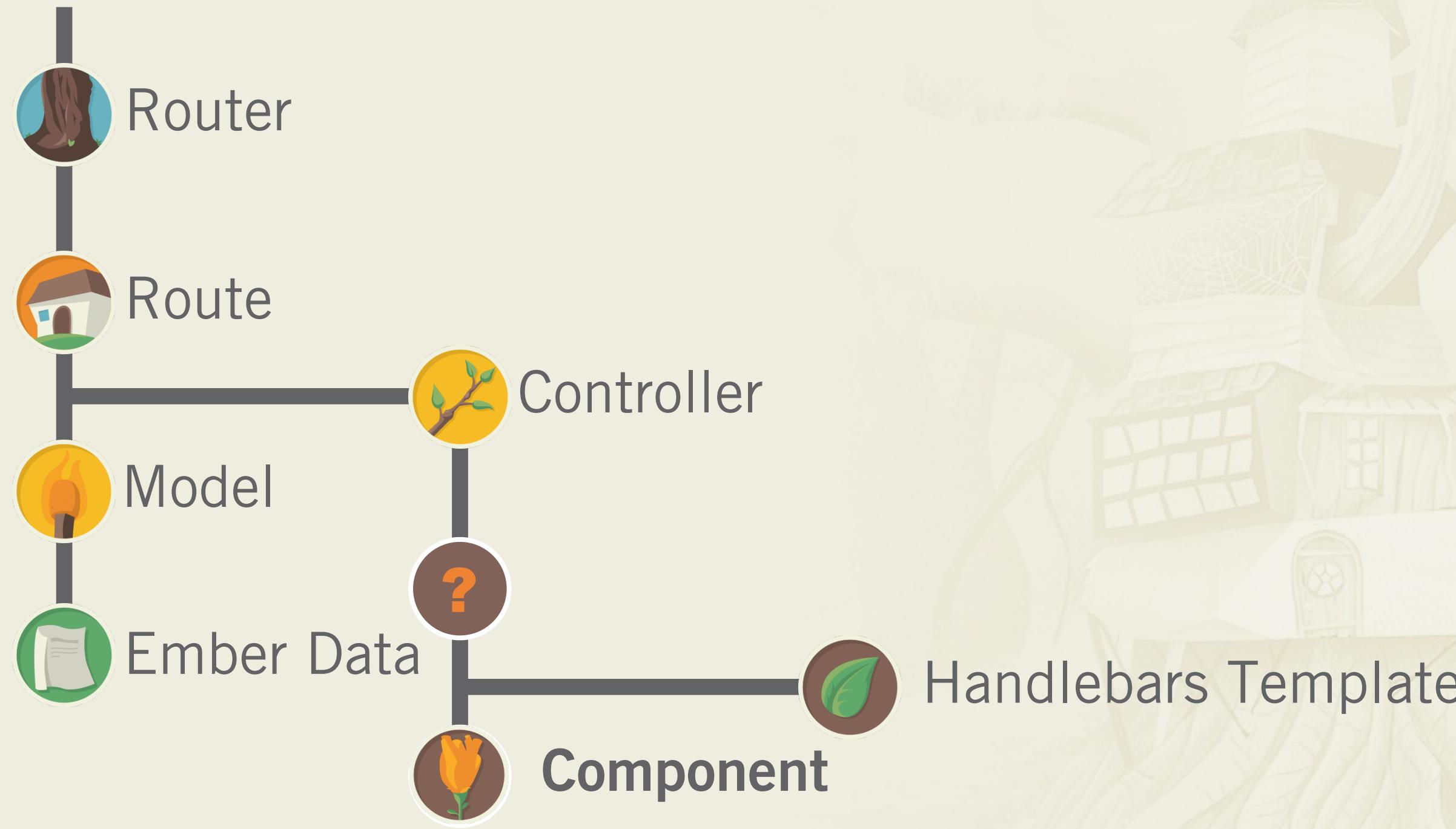
Can wrap user interactivity

Each Ember Component will use this icon



# Ember Component

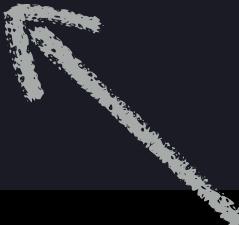
Browser Request



# Splitting out to a Component

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  {{#each onSale}}
    <div class='col-sm-4'>
      <img {{bind-attr src='image'}} class='img-thumbnail col-md-5' />
      <div class='col-sm-7'>
        <h2>{{title}}</h2>
        <p>{{description}}</p>
        <p>
          {{#link-to 'product' this classNames='btn btn-primary'}}
            Buy for ${{price}}
          {{/link-to}}
        </p>
      </div>
    </div>
  {{/each}}
</script>
```



This could be moved to a component



# Splitting out to a Component

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  {{#each onSale}}
    <div class='col-sm-4'>
      ...
      <h2>{{title}}</h2>
      ...
    </div>
  {{/each}}
</script>

<script type='text/x-handlebars' data-template-name='>
</script>
```



# Creating a Component Template

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  {{#each onSale}}
    <!-- Show the component -->
  {{/each}}
</script>
```

What should we name this new template?



```
<script type='text/x-handlebars' data-template-name=' '|>
<div class='col-sm-4'>
  ...
  <h2>{{title}}</h2>
  ...
</div>
</script>
```



# Component Template Naming

index.html

```
<script type='text/x-handlebars' data-template-name='index'>  
{{#each onSale}}  
  <!-- Show the component -->  
  {{/each}}  
</script>
```

Must start with components/



```
<script type='text/x-handlebars' data-template-name='components/product-details'>  
<div class='col-sm-4'>  
  ...  
  <h2>{{title}}</h2>  
  ...  
</div>  
</script>
```

Must have a dash in the component name



# Using a Component

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  {{#each onSale}}
    {{product-details}} ←
  {{/each}}
</script>

<script type='text/x-handlebars' data-template-name='components/product-details'>
<div class='col-sm-4'>
  ...
  <h2>{{title}}</h2>
  ...
</div>
</script>
```

Same as the component name

How do we let it know about the product?



# Passing Arguments to a Component

index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  {{#each onSale}}
    {{product-details product=this}}
  {{/each}}
</script>
```

```
<script type='text/x-handlebars' data-template-name='components/product-details'>
<div class='col-sm-4'>
  ...
    <h2>{{ product.title }}</h2>
  ...
</div>
</script>
```

Setting product in the component to  
the local product in the each loop

Reference product



# Component Wrapper Element

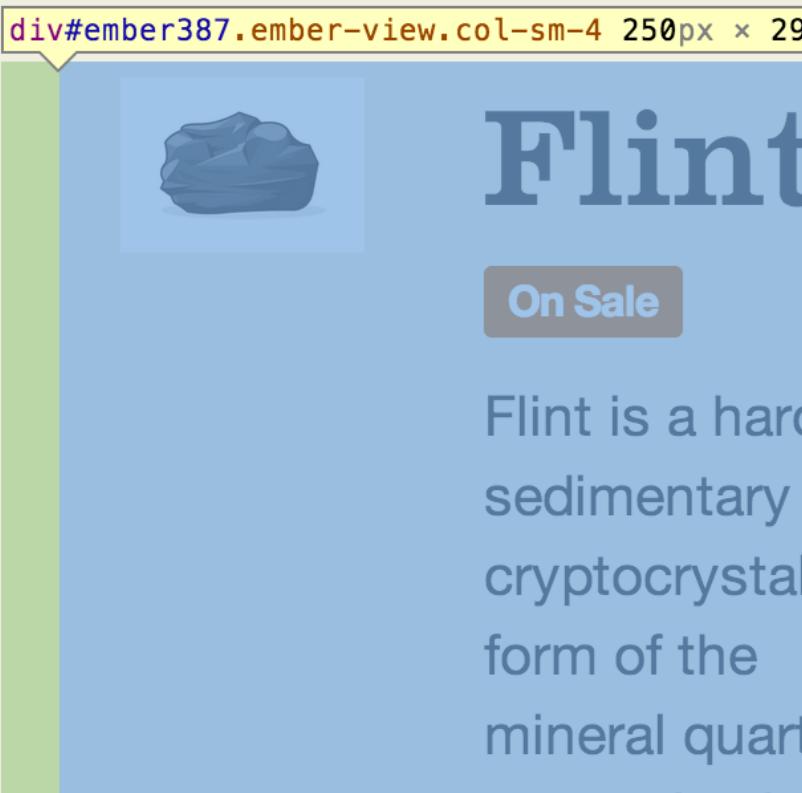
index.html

```
<script type='text/x-handlebars' data-template-name='index'>
  {{#each onSale}}
    {{product-details product=this classNames='col-sm-4'}}
  {{/each}}
</script>
<script type='text/x-handlebars' data-template-name='components/product-details'>
...
<h2>{{product.title}}</h2>
...
</script>
```

We can pass in a class name for our wrapper div

The component will render into a new  
div element with our given class





# Matches



# Tinder

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz,

One end is coated with a material that can be ignited by frictional

Tinder is easily combustible material used to ignite fires by rudimentary

# Updating products/onsale

index.html

```
<script type='text/x-handlebars' data-template-name='products/onsale'>
  <ul>
    {{#each}}
      <li class='row'>
        <!-- product details -->
      </li>
    {{/each}}
  </ul>
</script>
```



We'll need to use the component here too



# Passing in the Product

index.html

```
<script type='text/x-handlebars' data-template-name='products/onsale'>
  <ul>
    {{#each}}
      <li class='row'>
        {{product-details product=this}}
      </li>
    {{/each}}
  </ul>
</script>
```



# Changing the Root Element

index.html

```
<script type='text/x-handlebars' data-template-name='products/onsale'>
  <ul>
    {{#each}}
      {{product-details product=this tagName='li' classNames='row'}}
    {{/each}}
  </ul>
</script>
```



Pass in tag, like we did with link-to

Add the class for the  
wrapper element



The Flint & Flame

localhost:3000/#/products/onsale

Bow Drill

Flint

Kindling

Matches

Tinder

li#ember477.ember-view.row 562px x 192px

# Flint

On Sale



Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

Buy for \$99

Elements Resources Network Sources Timeline Profiles Audits Console Ember

<script id="metamorph-38-start" type="text/x-placeholder"></script>

> <li id="ember477" class="ember-view row">...</li>

<script id="metamorph-38-end" type="text/x-placeholder"></script>

<script id="metamorph-39-start" type="text/x-placeholder"></script>

> <li id="ember481" class="ember-view row">...</li>

<script id="metamorph-39-end" type="text/x-

Styles Computed »

element.style { }

media="screen" (index)

h2 application.css?body=1:78 { margin-top: 5px;

html body #ember311 div div div ul #ember477 div h2 ! 11

# Warming Up With ember.js

Level 6 - The Template Forest

Component Objects



# Review

index.html

```
<script type='text/x-handlebars' data-template-name='products/onsale'>
  <ul>
    {{#each}}
      {{product-details product=this tagName='li' classNames='row'}}
    {{/each}}
  </ul>
</script>

<script type='text/x-handlebars' data-template-name='components/product-details'>
  ...
  <h2>{{product.title}}</h2>
  ...
</script>
```



We split out a product into a component and reused it



# Adding Functionality to our Template

index.html

```
<script type='text/x-handlebars' data-template-name='components/product-details'>
...
{{#if hasReviews}}
  <p class='text-muted'>Read all reviews {{reviewsCount}}.</p>
{{/if}}
</script>
```



How do we add properties for **hasReviews** and **reviewsCount** to our component?

# Component Objects

index.html

```
 {{product-details product=this tagName='li' classNames='row'}}
```



first looks for



Component Object

App.ProductDetailsComponent



This is where we can set properties!

then renders



Component Template

components/product-details



# Component Object Properties

We can create properties on the component object, just like a controller.

index.html

```
{{product-details product=this tagName='li' classNames='row'}}
```

app.js

```
App.ProductDetailsComponent = Ember.Component.extend({
  reviewsCount: Ember.computed.alias('product.reviews.length'),
  hasReviews: function() {
    return this.get('reviewsCount') > 0;
  }.property('reviewsCount')
});
```

uses the passed in product

Watch reviewsCount property, and if it changes  
run this function and update the template



# Using Properties In Component Templates

app.js

```
App.ProductDetailsComponent = Ember.Component.extend({
  reviewsCount: Ember.computed.alias('product.reviews.length'),
  hasReviews: function() {
    return this.get('reviewsCount') > 0;
  }.property('reviewsCount')
});
```



index.html

```
<script type='text/x-handlebars' data-template-name='components/product-details'>
  ...
  {{#if hasReviews}}
    <p class='text-muted'>Read all reviews {{reviewsCount}}.</p>
  {{/if}}
</script>
```



The template will look in the component for properties

# Welcome to The Flint & Flame!

Everything you need to make it through the winter.

[Browse All 6 Items »](#)

## Flint

**On Sale**

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

[Read all reviews \(2\).](#)[Buy for \\$99](#)

## Matches

**On Sale**

One end is coated with a material that can be ignited by frictional heat generated by striking the match against a suitable surface.

[Read all reviews \(1\).](#)[Buy for \\$499](#)

## Tinder

**On Sale**

Tinder is easily combustible material used to ignite fires by rudimentary methods.

[Buy for \\$499](#)

# Passing Component Objects Properties

index.html

```
{{product-details product=this tag='li' class='row'}}
```



We can also set attributes inside this new class

index.html

```
{{product-details product=this}}
```



app.js

```
App.ProductDetailsComponent = Ember.Component.extend({  
  tagName: 'li',  
  classNames: ['row']  
});
```



# Warming Up With ember.js

## Level 6 - The Template Forest

View Objects





Birch Bark Shaving

Bow Drill

Flint

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

## Reviews

- Started a fire in no time!
- Not the brightest flame, but warm!



No indicator this product is on sale. How about adding CSS class and doing some styling?

# Product Template

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='row'>
    <div class='col-sm-7'>
      <h2>{{title}}</h2>
      ...
    </div>
  </div>
</script>
```

How can we add the class “is-on-sale” to this div, if the product is on sale



# Ember View



Each Ember View will use this icon

Responsible for encapsulating HTML content

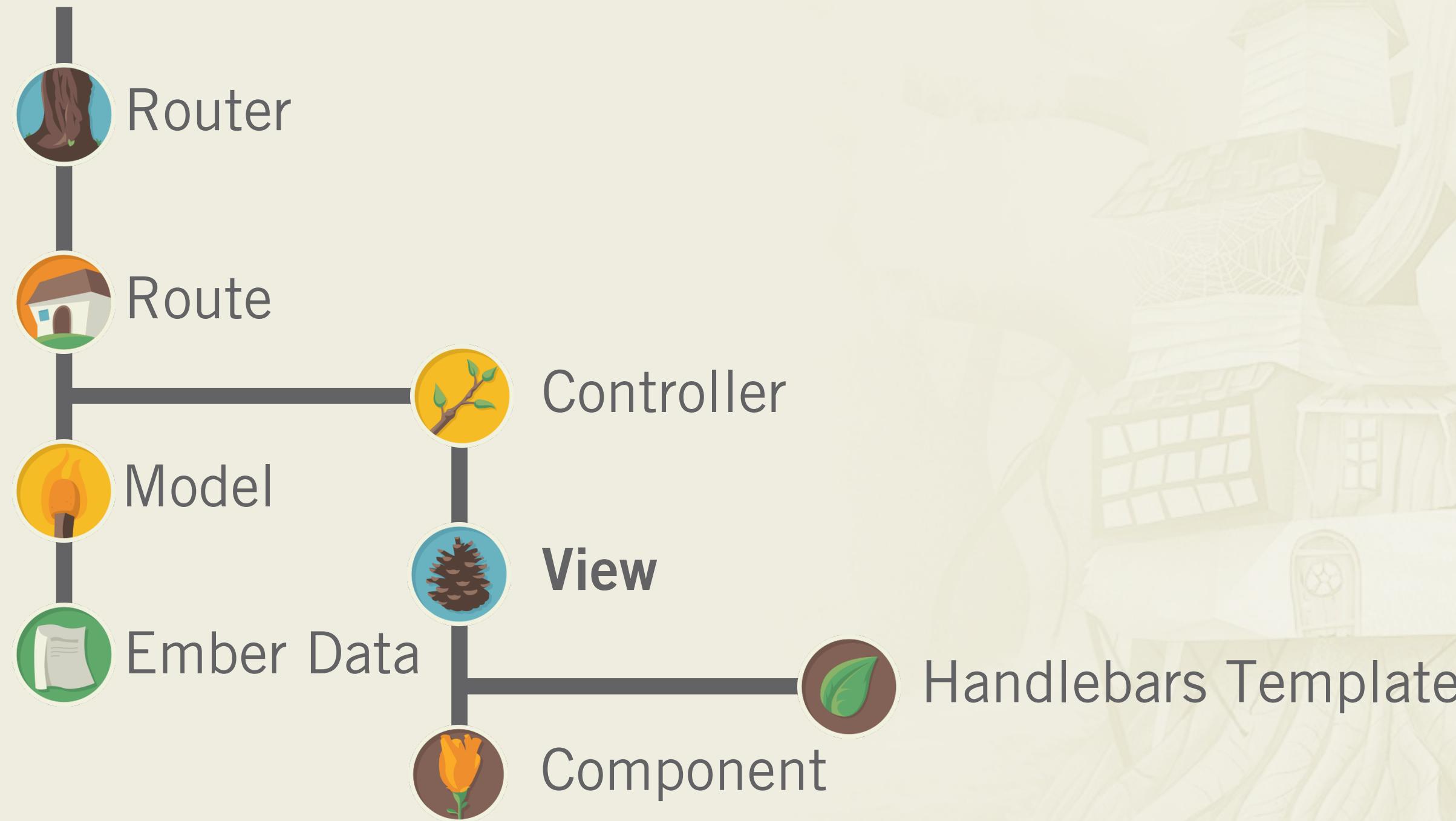
Registering and responding to user-initiated events

Ember Component Extends from Ember View



# Ember View

Browser Request



# Ember View In a Request



Route

```
App.ProductRoute = Ember.Route.extend({});
```



Controller

```
App.ProductController = Ember.Controller.extend({});
```



View

```
App.ProductView = Ember.View.extend({});
```



Template

```
data-template-name='product'
```



# Product Root Element

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='row'>
    ...
  </div>
</script>
```



Remember how the template is creating a div for us?

```
<div class='ember-view' id='ember432'>
  <div class='row'>
    ...
  </div>
</script>
```

output

We can define a View to set the proper  
classes on a template's div

"row" and optionally "is-on-sale"



# Product View Object

index.html

```
<script type='text/x-handlebars' data-template-name='product'>  
...  
</script>
```

app.js

```
App.ProductView = Ember.View.extend({  
  classNames: ['row']  
});
```

Can alter properties including tag, class, which template it should render and more



# View Class Names

Looks for `isOnSale` property on the controller,  
doesn't find it, so looks in the model

Views can access the model through the controller

app.js

```
App.ProductView = Ember.View.extend({
  classNames: ['row'],
  classNameBindings: ['isOnSale'],
  isOnSale: Ember.computed.alias('controller.isOnSale')
});
```



Will check the `isOnSale` view property, and if  
true add the class of `is-on-sale` to the div





## Birch Bark Shaving

## Bow Drill

Flint

Kindling

## Matches

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

Reviews



[Elements](#) [Resources](#) [Network](#) [Sources](#) [Timeline](#) [Profiles](#) [Audits](#) [Console](#) [Ember](#)

```
▶ <div class="col-sm-9">...</div>
▼ <div class="col-sm-9">
    <script id="metamorph-15-start" type="text/x-
placeholder"></script>
```

```
►<div id="ember369" class="ember-view row is-on-sale">  
  </div>
```

```
<script id="metamorph-15-end" type="text/x-placeholder"></script>
```

```
</div>
```

```
element.style { }  
media="screen" (index)  
.row bootstrap.css?body=1:694  
{  
margin-left: -15px;
```

# Warming Up With ember.js

## Level 6 - The Template Forest

Partials



# Review

```
App.ProductDetailsComponent = Ember.Component.extend({
  reviewsCount: Ember.computed.alias('product.reviews.length'),
  hasReviews: function() {
    return this.get('reviewsCount') > 0;
  }.property('reviewsCount')
});
```

```
App.ProductView = Ember.View.extend({
  classNames: ['row'],
  classNameBindings: ['isOnSale'],
  isOnSale: Ember.computed.alias('controller.isOnSale')
});
```

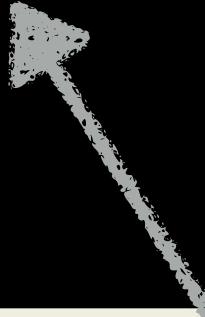


# Our Product Template

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <h2>{{title}}</h2>

  <h3>Reviews</h3>
  <ul>
    {{#each reviews}}
      <li><p>{{text}}</p></li>
    {{else}}
      <li><p class='text-muted'>
        <em>No reviews yet. Be the first to write one!</em>
      </p></li>
    {{/each}}
  </ul>
</script>
```



What if we also showed Comments? Related products? Products customers also bought?



# Our Product Template

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <h2>{{title}}</h2>

  <h3>Reviews</h3>
  <ul>
    {{#each reviews}}...{{/each}}
  </ul>
</script>
```



# Moving out to a partial

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <h2>{{title}}</h2>
</script>

<script type='text/x-handlebars' data-template-name='?'>
  <h3>Reviews</h3>
  <ul>
    {{#each reviews}}...{{/each}}
  </ul>
</script>
```

We could use a component for this, but it might be overkill



# Partial Naming

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <h2>{{title}}</h2>
  {{partial 'reviews'}} ←
</script>
```

No need to pass anything in, the partial has access to everything

```
<script type='text/x-handlebars' data-template-name='_reviews'>
  <h3>Reviews</h3>
  <ul>
    {{#each reviews}}...{{/each}}
  </ul>
</script>
```

Prefix the template name with an underscore





Birch Bark Shaving

Bow Drill

Flint

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

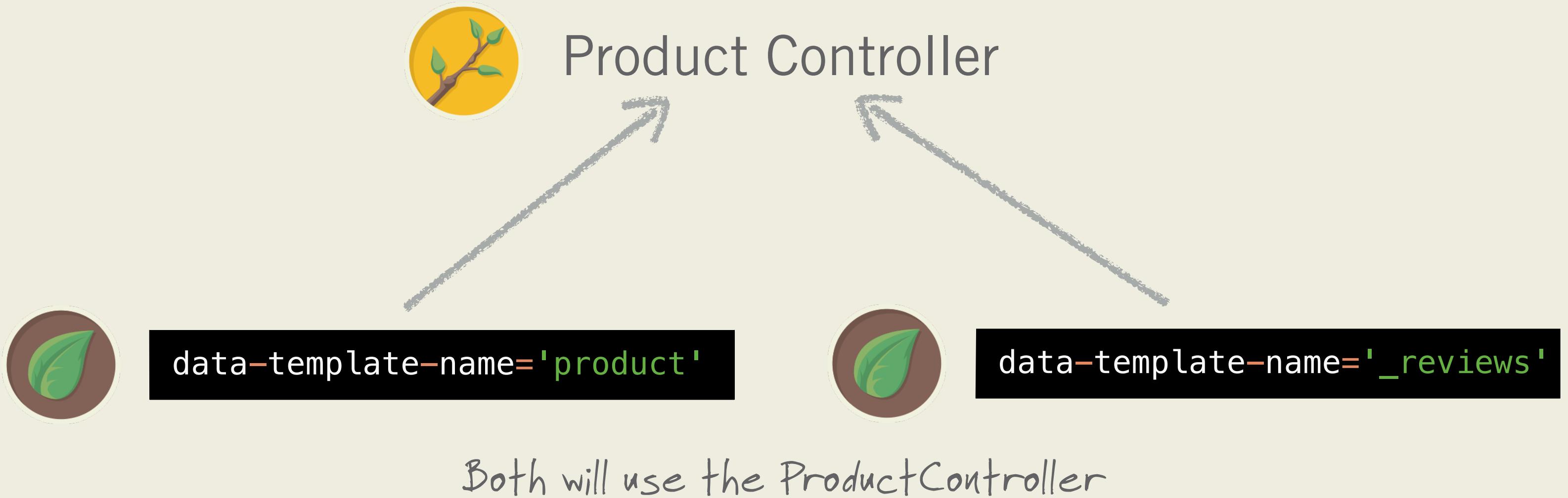
## Reviews

- Started a fire in no time!
- Not the brightest flame, but warm!



We're showing all reviews in the order  
they were defined in our fixtures

# Controller Access in Partials



# Rendering Differences

Route



Route

Controller



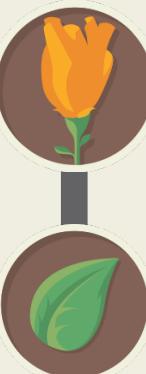
View



Template



`{{component}}`



Component

Template

`{{partial}}`



Template



# Warming Up With ember.js

## Level 6 - The Template Forest

Using {{render}}



# Review

index.html

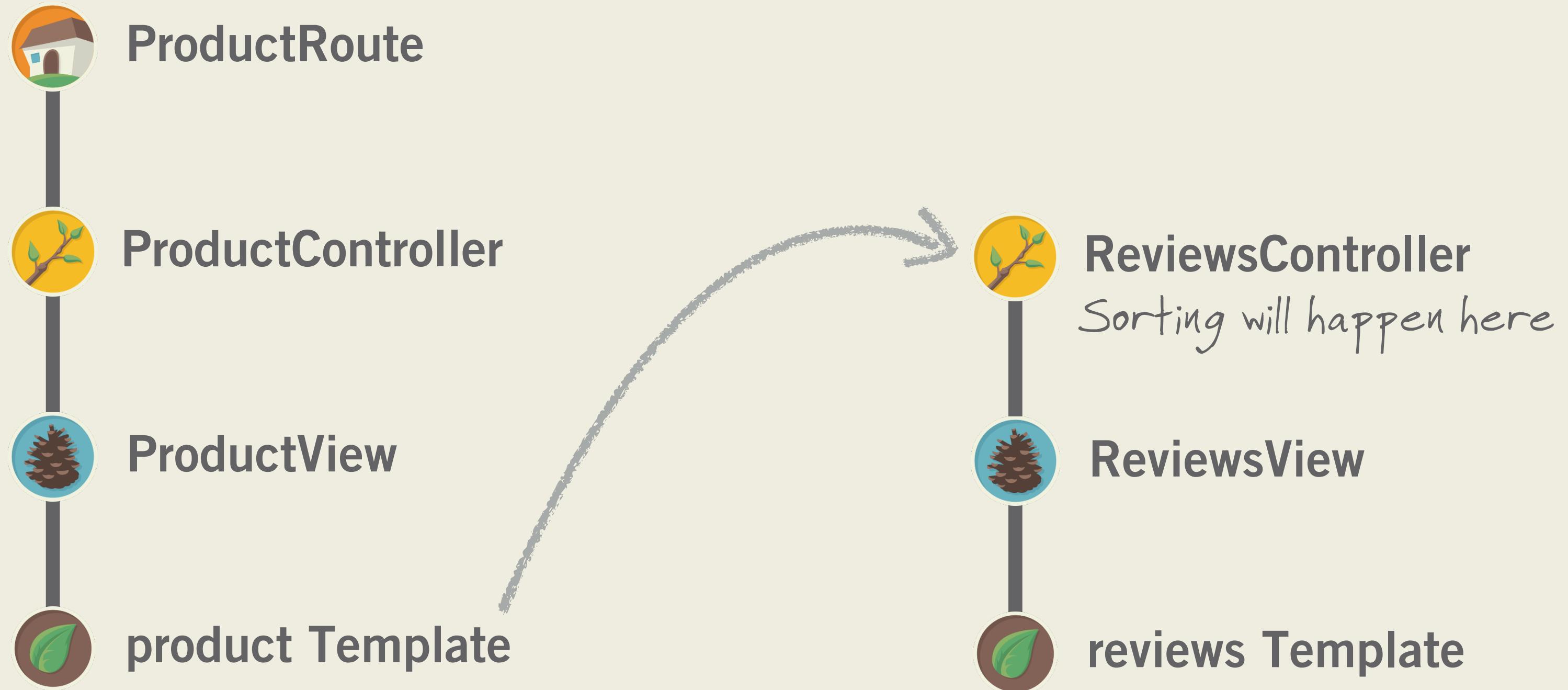
```
<script type='text/x-handlebars' data-template-name='product'>
  <h2>{{title}}</h2>
  {{partial 'reviews'}}
</script>

<script type='text/x-handlebars' data-template-name='_reviews'>
  <h3>Reviews</h3>
  <ul>
    {{#each reviews}}...{{/each}}
  </ul>
</script>
```

What if we wanted to show the most recent reviews first and we want to do this client side?



# Creating a Controller from a Template



# Using {{render}}

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <h2>{{title}}</h2>
  {{render 'reviews' reviews}} ← The object passed in will
</script>                                become the model
```

```
<script type='text/x-handlebars' data-template-name='reviews'>
  <h3>Reviews</h3>
  <ul>
    {{#each}}...{{/each}}
  </ul>
</script>
```

We will create App.ReviewsController to do the sorting



# Controller Access with Render



ProductRoute



ProductController

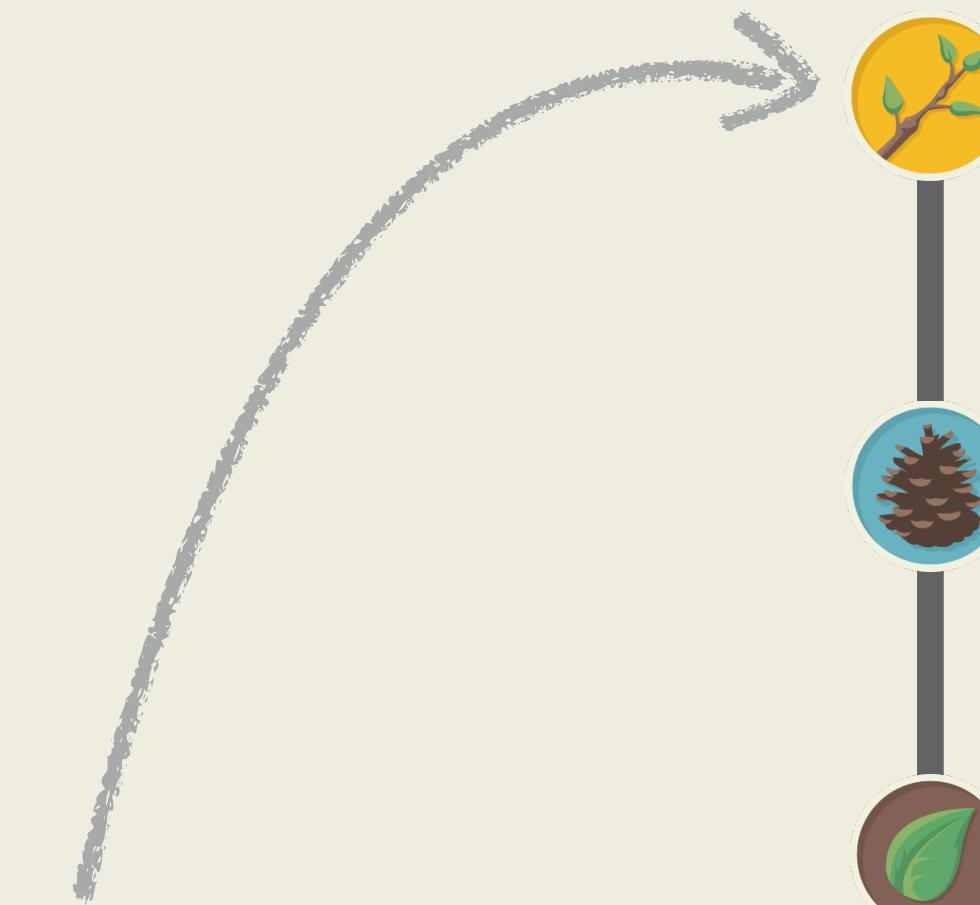


ProductView



product Template

Tells Ember to go look for a ReviewsController,  
with reviews as the model



ReviewsController



ReviewsView



review Template

```
 {{render 'reviews' reviews}}
```



# Create our Controller with Sorting

app.js

```
App.ReviewsController = Ember.ArrayController.extend({  
  sortProperties: ['reviewedAt'],  
  sortAscending: false  
});
```

Uses the reviews passed in as its model

```
{{render 'reviews' reviews}}
```

Renders out the reviews template

```
<script type='text/x-handlebars' data-template-name='reviews'>
```



Birch Bark Shaving

Bow Drill

Flint

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

## Reviews

- Not the brightest flame, but warm!
- Started a fire in no time!



# Rendering Differences

Route



Route

Controller



`{{render}}`



Controller

View



`View`

Template



`{{component}}`



Component

`Template`



`{{partial}}`



Template



# Warming Up With ember.js

Level 7 - Acting on Instinct

Controller Actions



# Review

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <h2>{{title}}</h2>
  {{render 'reviews' reviews}} ←
</script>
```

The object passed in will become the model

```
<script type='text/x-handlebars' data-template-name='reviews'>
  <h3>Reviews</h3>
  <ul>
    {{#each}}...{{/each}}
  </ul>
</script>
```

Controller   View   Template

`{{render}}`





Birch Bark Shaving

Bow Drill

Flint

Kindling

Matches

Tinder

# Flint

\$99

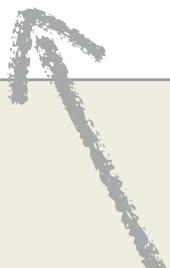
Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

## Reviews

- Not the brightest flame, but warm!
- Started a fire in no time!

## Review Flint

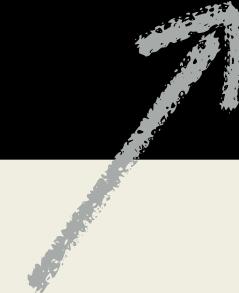
Review



# valueBinding for the textarea

index.html

```
<script type='text/x-handlebars' data-template-name='product'>  
  ...  
  <div class='new-review'>  
    <h3>Review {{title}}</h3>  
    {{textarea valueBinding='text'}}  
  </div>  
</script>
```



Binds the value of the text area to the text  
property on the controller



# Outputting a Property

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='new-review'>
    <h3>Review {{title}}</h3>
    <p class='text-muted'>{{text}}</p>
    {{textarea valueBinding='text'}}<br/>
  </div>
</script>
```

Lets print this property out to the screen

Just like any other property





Birch Bark Shaving

Bow Drill

Flint

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

## Reviews

- Not the brightest flame, but warm!
- Started a fire in no time!



## Review Flint

# Next Steps

1. Only show “review” paragraph if {{text}} has a value.
2. Create the submit button.
3. Write the code which receives the review on submit.

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='new-review'>
    <h3>Review {{title}}</h3>
    <p class='text-muted'>{{text}}</p>
    {{textarea valueBinding='text'}}
  </div>
</script>
```



# First Two Steps

Only show the text if they've started typing

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='new-review'>
    <h3>Review {{title}}</h3>
    {{#if text}}
      <p class='text-muted'>{{text}}</p>
    {{/if}}
    {{textarea valueBinding='text'}}
      <button {{action 'createReview'}} class='btn-primary'>Review</button>
    </div>
</script>
```

Call the createReview function  
on the controller when clicked.



# Controller Action

```
<script type='text/x-handlebars' data-template-name='product'>
  {{textarea valueBinding='text'}}
  <button {{action 'createReview'}} class='btn-primary'>Review</button>
</script>
```

app.js

```
App.ProductController = Ember.ObjectController.extend({
  text: '',
  actions: {
    createReview: function() {
      console.log('createReview Called');
    }
  }
});
```

If left off, the property would  
be set on the model!

Clicking our button triggers the  
createReview function to be run

The Flint & Flame

localhost:3000/#/products/1

# Review Flint

Started a new review.

Review

---

© 2013 The Flint & Flame

Credits

Elements Resources Network Sources Timeline Profiles Audits Console Ember

```
DEBUG: Ember      : 1.2.0          ember.js?body=1:3231
DEBUG: Ember Data : 1.0.0-beta.5+pre.69cb8b87  ember.js?body=1:3231
DEBUG: Handlebars : 1.1.2          ember.js?body=1:3231
DEBUG: jQuery     : 1.10.2         ember.js?body=1:3231
DEBUG: -----        -----        ember.js?body=1:3231
```

>

<top frame>

<page context>

# Saving with Ember Data

app.js

```
App.ProductController = Ember.ObjectController.extend({
  text: '',
  actions: {
    createReview: function() {
      console.log('createReview Called');

      // Step 1: Build a new Review object
      // Step 2: Save the Review
      // Step 3: Clear out the text variable
    }
  }
});
```



We'll need to complete these steps in order to save this review



# Building a new Review Object

app.js

```
App.ProductController = Ember.ObjectController.extend({
  text: '',
  actions: {
    createReview: function() {
      // Step 1: Build a new Review object
      var review = this.store.createRecord('review', {
        text: this.get('text'),
        product: this.get('model'),
        reviewedAt: new Date()
      });

      // Step 2: Save the Review
      // Step 3: Clear out the text variable
    }
  }
});
```

Sets up the new review object  
locally, but hasn't touched the server



# Saving a Review

app.js

```
App.ProductController = Ember.ObjectController.extend({
  text: '',
  actions: {
    createReview: function() {
      // Step 1: Build a new Review object
      var review = this.store.createRecord('review', {
        text: this.get('text'),
        product: this.get('model'),
        reviewedAt: new Date()
      });

      // Step 2: Save the Review
      review.save() ← Will add it to our local fixtures. Would
                        do a POST to a server if using the
                        REST adapter
      // Step 3: Clear out the text variable
    }
  }
});
```



# The save() Promise

app.js

```
App.ProductController = Ember.ObjectController.extend({
  text: '',
  actions: {
    createReview: function() {
      var review = this.store.createRecord('review', {
        text: this.get('text'),
        product: this.get('model'),
        reviewedAt: new Date()
      });
      var controller = this; ←
      review.save().then(function(review) {
        // Will be called when the save call finishes
      });
    }
  }
});
```

Need to be able to reference the controller in the save callback



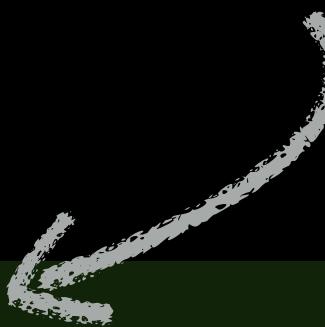
# Clear out the reviewText Variable

app.js

```
App.ProductController = Ember.ObjectController.extend({
  text: '',
  actions: {
    createReview: function() {
      var review = this.store.createRecord('review', {
        text: this.get('text'),
        product: this.get('model'),
        reviewedAt: new Date()
      });
      var controller = this;

      review.save().then(function(review) {
        controller.set('text', '');
        controller.get('model.reviews'). addObject(review);
      });
    }
  }
});
```

Clear out reviewText and add the  
review to products.reviews



The Flint & Flame

localhost:3000/#/products/1

# Flint & Flame

Home About Products

Birch Bark Shaving

Bow Drill

**Flint**

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



## Reviews

- Not the brightest flame, but warm!
- Started a fire in no time!

## Review Flint

I

Review

# Warming Up With ember.js

Level 7 - Acting on Instinct

Data binding with Models



# Review

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='new-review'>
    <h3>Review {{title}}</h3>
    {{#if text}}
      <p class='text-muted'>{{text}}</p>
    {{/if}}
    {{textarea valueBinding='text'}}
    <button {{action 'createReview'}} class='btn-primary'>Review</button>
  </div>
</script>
```



# Review

app.js

```
App.ProductController = Ember.ObjectController.extend({
  text: '',
  actions: {
    createReview: function() {
      var review = this.store.createRecord('review', {
        text: this.get('text'),
        product: this.get('model'),
        reviewedAt: new Date()
      });
      var controller = this;

      review.save().then(function(review) {
        controller.set('text', '');
        controller.get('model.reviews'). addObject(review);
      });
    }
  }
});
```



The Flint & Flame

localhost:3000/examples/7-10#/products/4

# Flint & Flame

Home About Products

Birch Bark Shaving

**Bow Drill**

\$999

The bow drill is an ancient tool. While it was usually used to make fire, it was also used for primitive woodworking and dentistry.

## Reviews

- I don't even have arms, how would I operate this?

### Review Bow Drill

|

X

Review



# Controller Reuse

This same controller object is being reused!

app.js

```
App.ProductController = Ember.ObjectController.extend({  
  text: ''  
});
```

The only part that's changing is the model  
property, representing the product.



# Using A New Model

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='new-review'>
    <h3>Review {{title}}</h3>
    {{#if text}}
      <p class='text-muted'>{{text}}</p>
    {{/if}}
    {{textarea valueBinding='text'}}
    <button {{action 'createReview'}} class='btn-primary'>Review</button>
  </div>
</script>
```



What if we bind this functionality to a  
“new review” object?

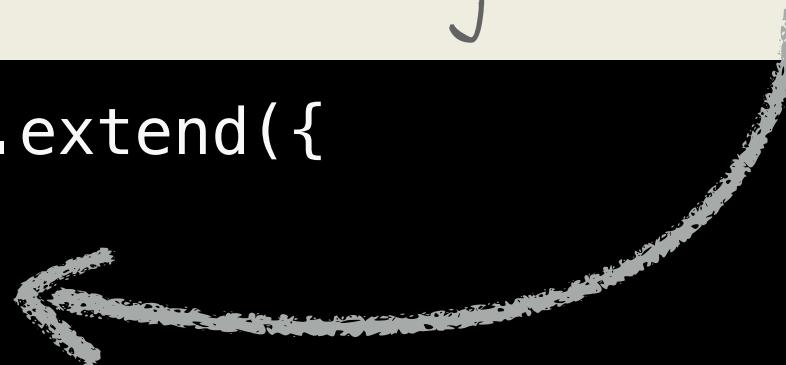


# Two Things to Fix

app.js

```
App.ProductController = Ember.ObjectController.extend({  
  text: '',  
  actions: {  
    createReview: function() {  
      ...  
    }  
  }  
});
```

Create a new review object for this product



1. Create a ‘review’ property that returns a new Review model, and we’ll set the text on the review model.
2. Create a new (blank) ‘review’ every time the product model changes.



# A New Review

app.js

```
App.ProductController = Ember.ObjectController.extend({
  review: function() {
    return this.store.createRecord('review', {
      product: this.get('model')
    });
  }.property('model'),
  actions: {
    createReview: function() {
      ...
    }
  }
});
```

Create a new review object for this product

Watch the model, and when it changes  
immediately invoke this function.



# Now let's use the Review model

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='new-review'>
    <h3>Review {{title}}</h3>
    {{#if text}}
      <p class='text-muted'>{{text}}</p>
    {{/if}}
    {{textarea valueBinding='text'}}
    <button {{action 'createReview'}} class='btn-primary'>Review</button>
  </div>
</script>
```



# Template Updates for Review

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  <div class='new-review'>
    <h3>Review {{title}}</h3>
    {{#if review.text}}
      <p class='text-muted'>{{review.text}}</p>
    {{/if}}
    {{textarea valueBinding='review.text'}}
    <button {{action 'createReview'}} class='btn-primary'>Review</button>
  </div>
</script>
```

Bind directly to the text  
property on our review object



# Let's Update the createReview

app.js

```
App.ProductController = Ember.ObjectController.extend({
  review: function() { ... }.property('model'),
  actions: {
    createReview: function() {
      var review = this.store.createRecord('review', {
        text: this.get('text'),
        product: this.get('model'),
        reviewedAt: new Date()
      });

      var controller = this;

      review.save().then(function(review) {
        controller.set('text', '');
        controller.get('model.reviews'). addObject(review);
      });
    }
  }
});
```



We no longer need to  
create a model.



# Updating createReview

app.js

```
App.ProductController = Ember.ObjectController.extend({
  review: function() { ... }.property('model'),
  actions: {
    createReview: function() {
      var controller = this;
      this.get('review').set('reviewedAt', new Date());
      this.get('review').save().then(function(review) {
        controller.get('model.reviews'). addObject(review);
      });
    }
  }
});
```



The Flint & Flame

localhost:3000/#/products/1

# Flint & Flame

Home About Products

Birch Bark Shaving

Bow Drill

**Flint**

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



## Reviews

- Not the brightest flame, but warm!
- Started a fire in no time!

## Review Flint

Review

# Hiding after Review Creation

How do we hide this form after a review is created?

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  ...
<div class='new-review'>
  <h3>Review {{title}}</h3>
  {{#if review.text}}
    <p class='text-muted'>{{review.text}}</p>
  {{/if}}
  {{textarea valueBinding='review.text'}}
  <button {{action 'createReview'}} class='btn-primary'>Review</button>
</div>

</script>
```



# Hiding the Form

We'll need to create this property on our ProductController

index.html

```
<script type='text/x-handlebars' data-template-name='product'>  
  ...  
  {{#if isNotReviewed}} <div class='new-review'>  
    <h3>Review {{title}}</h3>  
    {{#if review.text}}  
      <p class='text-muted'>{{review.text}}</p>  
    {{/if}}  
    {{textArea valueBinding='review.text'}}  
    <button {{action 'createReview'}} class='btn-primary'>Review</button>  
  </div>  
  {{/if}}
```



# isNotReviewed Property

app.js

```
App.ProductController = Ember.ObjectController.extend({
  review: function() { ... }.property('model'),
  actions: { ... },
  isNotReviewed: Ember.computed.alias('review.isNew')
});
```



## Ember Data Model States

See the Ember API Docs for all states at <http://emberjs.com/>

**isNew** Has not been saved

**isLoading** While fetching from a server

**isDirty** Has unsaved changes

**isLoaded** After fetching from a server



The Flint & Flame

localhost:3000/#/products/1

# Flint & Flame

Home About Products

Birch Bark Shaving

Bow Drill

**Flint**

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



## Reviews

- Not the brightest flame, but warm!
- Started a fire in no time!

## Review Flint

Review

# Warming Up With ember.js

## Level 7 - Acting on Instinct

Handling Events in Views



# Review

index.html

```
<script type='text/x-handlebars' data-template-name='product'>
  {{#if isNotReviewed}}
    <div class='new-review'>
      <h3>Review {{title}}</h3>
      {{#if review.text}}
        <p class='text-muted'>{{review.text}}</p>
      {{/if}}
      {{textarea valueBinding='review.text'}}
      <button {{action 'createReview'}} class='btn-primary'>Review</button>
    </div>
  {{/if}}
</script>
```



We have a form to create reviews



# Review

app.js

```
App.ProductController = Ember.ObjectController.extend({
  isNotReviewed: Ember.computed.alias('review.isNew'),
  review: function() {
    return this.store.createRecord('review', {
      product: this.get('model')
    });
  }.property('model'),
  actions: {
    createReview: function() {
      var controller = this;
      this.get('review').set('reviewedAt', new Date());
      this.get('review').save().then(function(review) {
        controller.get('model.reviews'). addObject(review);
      });
    }
  }
});
```





Birch Bark Shaving

Bow Drill

Flint

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

## Reviews

- This is some amazing Flint! It lasts forever and works even when damp! I still remember the first day when I was only a little fire sprite and got one of these in my flame stalking for treemas. My eyes lit up the moment I tried it! Years later I'm still using the same one. That's the biggest advantage of this -- it doesn't run out easily like matches. As long as you have something to strike it against, you can start a fire anywhere you have something to burn!
- Not the brightest flame, but warm!
- Started a fire in no time!



These reviews are getting long! What about only showing a sample with a “read more...” link



What about allowing them to enter Markdown?

# Viewing Our Reviews

index.html

```
<script type='text/x-handlebars' data-template-name='reviews'>
  <h3>Reviews</h3>
  <ul>
    {{#each}}
      <li><p>{{text}}</p></li>
    {{else}}
      <li>
        <p class='text-muted'>
          <em>No reviews yet. Be the first to write one!</em>
        </p>
      </li>
    {{/each}}
  </ul>
</script>
```



We can abstract out a “review” into a View

# Content Within a View

index.html

```
<script type='text/x-handlebars' data-template-name='reviews'>
  <h3>Reviews</h3>
  <ul>
    {{#each}}
      {{#view tag='li'}}
        <div class='content'>{{text}}</div>
      {{/view}}
    {{else}}
      ...
    {{/each}}
  </ul>
</script>
```

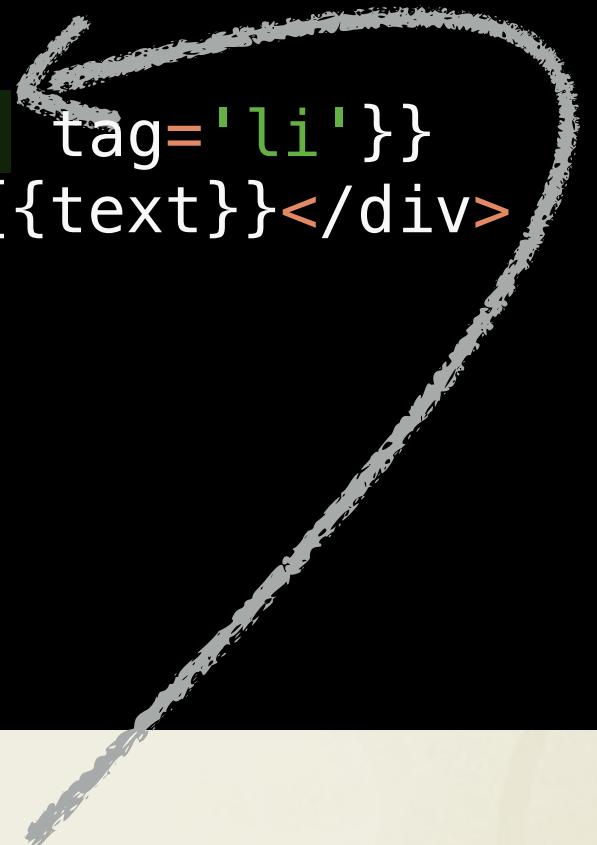
Creates a generic view with a li tag rather than a div



# Using a View Class

index.html

```
<script type='text/x-handlebars' data-template-name='reviews'>
  <h3>Reviews</h3>
  <ul>
    {{#each}}
      {{#view 'App.ReviewView' tag='li'}}
        <div class='content'>{{text}}</div>
      {{/view}}
    {{else}}
      ...
    {{/each}}
  </ul>
</script>
```



Will use the App.ReviewView object as  
the view, let's create it!



# Creating a View Class

app.js

```
App.ReviewView = Ember.View.extend({  
  ???  
});
```

## TODO

1. We need a class ‘isExpanded’ that gets added to the ‘li’ when the text is expanded.
2. Need a click event, which toggle’s this class.



# Add the Proper Class

app.js

```
App.ReviewView = Ember.View.extend({  
  isExpanded: false,  
  classNameBindings: ['isExpanded']  
});
```

Will NOT have a class of `is-expanded` initially.

How can we add it when the review is clicked?



# Listening for Click Events

When the review li element is clicked, change the expanded property  
app.js

```
App.ReviewView = Ember.View.extend({  
  isExpanded: false,  
  classNameBindings: ['isExpanded'],  
  click: function() {  
    this.toggleProperty('isExpanded'); ←  
  }  
});
```



## Ember View Events

**click**

**mouseDown**

**submit**

**change**

**keyDown**

**keyUp**

See the Ember API Docs for all events at <http://emberjs.com/>



The Flint & Flame

localhost:3000/#/products/1

Home About Products



# Flint & Flame

Birch Bark Shaving

Bow Drill

Flint

Kindling

Matches

Tinder

# Flint

\$99

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.

## Reviews

This is some amazing Flint! It lasts forever and works even when damp! I still remember



Not the brightest flame, but warm!

# Our Problem

Need to add a class ‘readMore’ to the reviews that have more than 140 characters, so we can add “Read More”.

## Our Solution

1. Send our ‘text.length’ value into the View.
2. Create a function called ‘readMore’ which will check if string is > 140 characters.
3. Add the readMore className.
4. Add a “Read More” / “Read Less” link



# Pass in the length

index.html

```
<script type='text/x-handlebars' data-template-name='reviews'>
  <h3>Reviews</h3>
  <ul>
    {{#each}}
      {{#view 'App.ReviewView' tag='li' length=text.length}}
        <div class='content'>{{text}}</div>
      {{/view}}
    {{else}}
      ...
    {{/each}}
  </ul>
</script>
```

Sets the length property on the ReviewView



# Add the readMore Function

app.js

```
App.ReviewView = Ember.View.extend({
  isExpanded: false,
  classNameBindings: ['isExpanded'],
  click: function() {
    this.toggleProperty('isExpanded');
  },
  readMore: function() {
    return this.get('length') > 140;
  }.property('length')
});
```



# Add the ClassName

app.js

```
App.ReviewView = Ember.View.extend({
  isExpanded: false,
  classNameBindings: ['isExpanded', 'readMore'],
  click: function() {
    this.toggleProperty('isExpanded');
  },
  readMore: function() {
    return this.get('length') > 140;
  }.property('length')
});
```

Add a class if this is a long review

index.html

```
{{#view 'App.ReviewView' tag='li' length=text.length}}
  <div class='content'>{{text}}</div>
  <span class='expand text-success'>
    Read {{#if view.isExpanded}}Less{{else}}More{{/if}}
  </span>
{{/view}}
```



# Ember Computed

app.js

```
App.ReviewView = Ember.View.extend({
  readMore: function() {
    return this.get('length') > 140;
  }.property('length')
});
```

Functionally the same!

```
App.ReviewView = Ember.View.extend({
  readMore: Ember.computed.gt('length', 140)
});
```

**Ember.computed**

**alias**

**gt      gte**

**lt      lte**

**map      any      filter**

**max      min**



The Flint & Flame

localhost:3000/#/products/1

Flint

Kindling

Matches

Tinder

Flint is a hard, sedimentary cryptocrystalline form of the mineral quartz, categorized as a variety of chert.



## Reviews

This is some amazing Flint! It lasts forever and works even when damp! I still remember

[Read More](#)

---

Not the brightest flame, but warm!

---

Started a fire in no time!



## Review Flint

[Review](#)

# Defining Handlebars Helpers

app.js

```
Ember.Handlebars.registerBoundHelper('markdown', function(text) {  
  return text;  
});
```

Defines a handlebars helper named “markdown”

index.html

```
<script type='text/x-handlebars' data-template-name='reviews'>  
  {{#view 'App.ReviewView' tag='li' length=text.length}}  
    <div class='content'>{{markdown text}}</div>  
  {{/view}}  
</script>
```

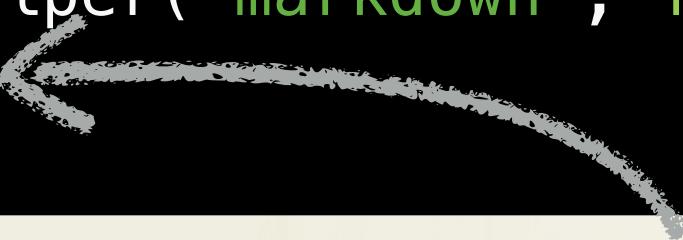
Use our handlebars helper, passing in the review



# Using External Libraries

app.js

```
Ember.Handlebars.registerBoundHelper('markdown', function(text) {  
  return markdown.toHTML(text);  
});
```



We're going to use an external Markdown library  
which provides a `markdown.toHTML()` method

index.html

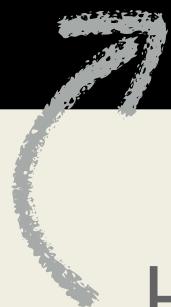
```
<script type='text/x-handlebars' data-template-name='reviews'>  
  {{#view 'App.ReviewView' tag='li' length=text.length}}  
    <div class='content'>{{markdown text}}</div>  
  {{/view}}  
</script>
```



# Turning Off Escaping

app.js

```
Ember.Handlebars.registerBoundHelper('markdown', function(text) {  
  return new Handlebars.SafeString(markdown.toHTML(text));  
});
```



Handlebars will escape text by default, unless its a SafeString

index.html

```
<script type='text/x-handlebars' data-template-name='reviews'>  
  {{#view 'App.ReviewView' tag='li' length=text.length}}  
    <div class='content'>{{markdown text}}</div>  
  {{/view}}  
</script>
```



The Flint & Flame

localhost:3000/#/products/1

Matches

Tinder

## Reviews

This is some amazing Flint! It lasts **forever** and works even when damp! I still remember the first day when I was only a little fire novice and got one of these in my

[Read More](#)

---

Not the brightest flame, but warm!

---

Started a fire in no time!

## Review Flint

[Review](#)