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?

The Flint & Flame

localhost:3000/#/products

Flint & Flame

Home About Products

- Flint
- Kindling
- Matches
- Bow Drill
- Tinder
- Birch Bark Shaving

Choose a product from those on the left!

How could we sort products?

© 2013 The Flint & Flame

Credits

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
});
```

The Flint & Flame

localhost:3000/#/products

Flint & Flame

Home About Products

- Birch Bark Shaving
- Bow Drill
- Flint
- Kindling
- Matches
- Tinder

Choose a product from those on the left!

Products are now sorted by name

© 2013 The Flint & Flame

Credits

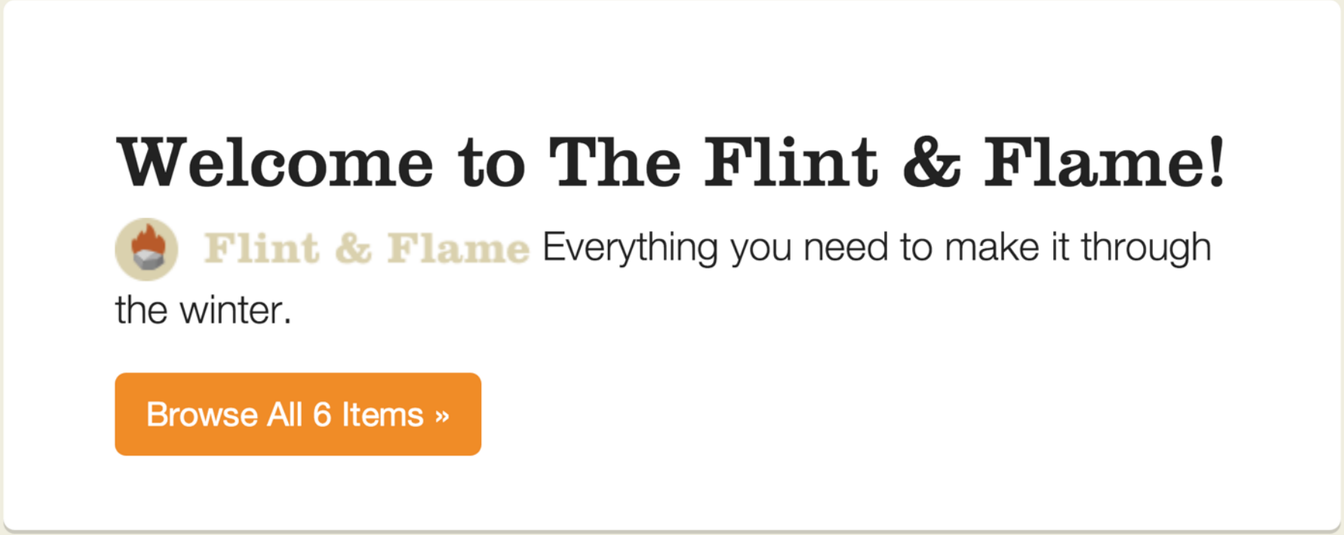
Refactoring IndexController

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

Using Ember Data

app.js

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



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

The model is an array

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

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/#/

& 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

The Flint & Flame

localhost:3000/#/

Flint & Flame

Home About Products

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’}}
           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

The Flint & Flame

localhost:3000/#/

Flint

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

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

The parent route of ProductsOnsale is Products

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

```
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
products.onsale	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

The Flint & Flame

localhost:3000/#/

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

© 2013 The Flint & Flame

Credits

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">
<ul class='list-unstyled'>
  {{#each}}
    <li class='row'>
      <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'}}
           Buy for ${{price}}{{/link-to}}</p>
      </div>
    </li>
  {{/each}}
</ul>
</script>
```

Show all products in the model

Same product details as on the homepage

The Flint & Flame

localhost:3000/#/products/onsale

Flint & Flame

Home About Products

- 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

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

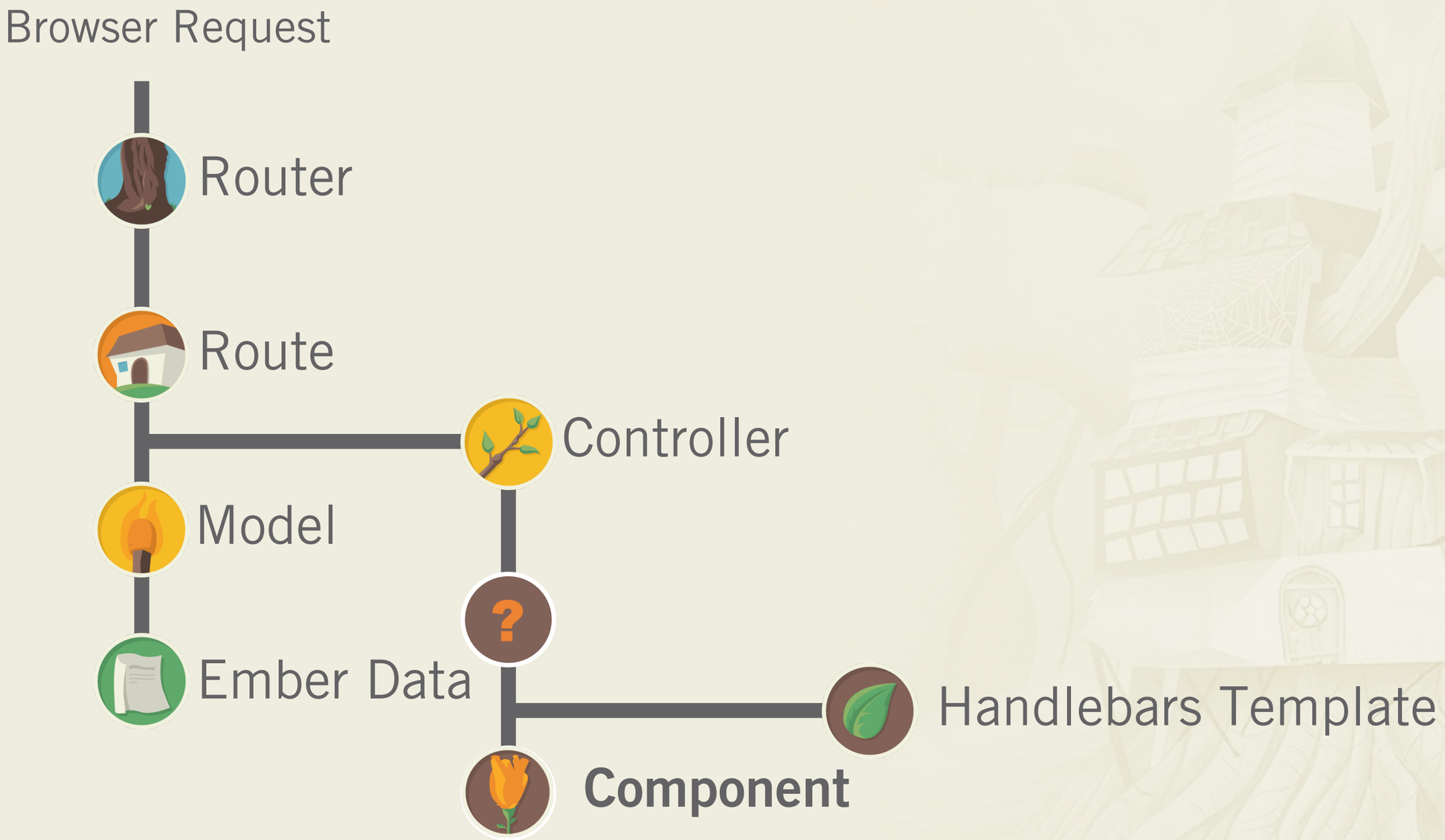
Similar implementation to Web Components

Split out parts of your page to reusable components

Can wrap user interactivity

Each Ember Component will use this icon

Ember Component



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>

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

Must start with components/

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

The Flint & Flame

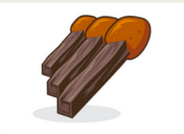
localhost:3000/#/

div#ember387.ember-view.col-sm-4 250px × 292px

Flint

On Sale

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



Matches

On Sale

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

Tinder

On Sale

Tinder is easily combustible material used to ignite fires by rudimentary

Elements Resources Network Sources Timeline Profiles Audits Console Ember

```
<script id="metamorph-3-start" type="text/x-
placeholder"></script>
▶ <div id="ember387" class="ember-view col-sm-4">
…</div>
<script id="metamorph-3-end" type="text/x-
placeholder"></script>
<script id="metamorph-4-start" type="text/x-
placeholder"></script>
▶ <div id="ember406" class="ember-view col-sm-4">
```

Styles Computed »

```
element.style {
}
media="screen"                    (index)
      bootstrap.css?body=1:880
@media (min-width: 768px)
      bootstrap.css?body=1:911
```

… div div div div#ember387.ember-view.col-sm-4 … 11

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 × 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>
```

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`

then renders

Component Template

`components/product-details`

This is where we can set properties!

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'}}
```

uses the passed in product

app.js

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

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

The Flint & Flame

localhost:3000

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

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

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

© 2013 The Flint & Flame

Credits

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

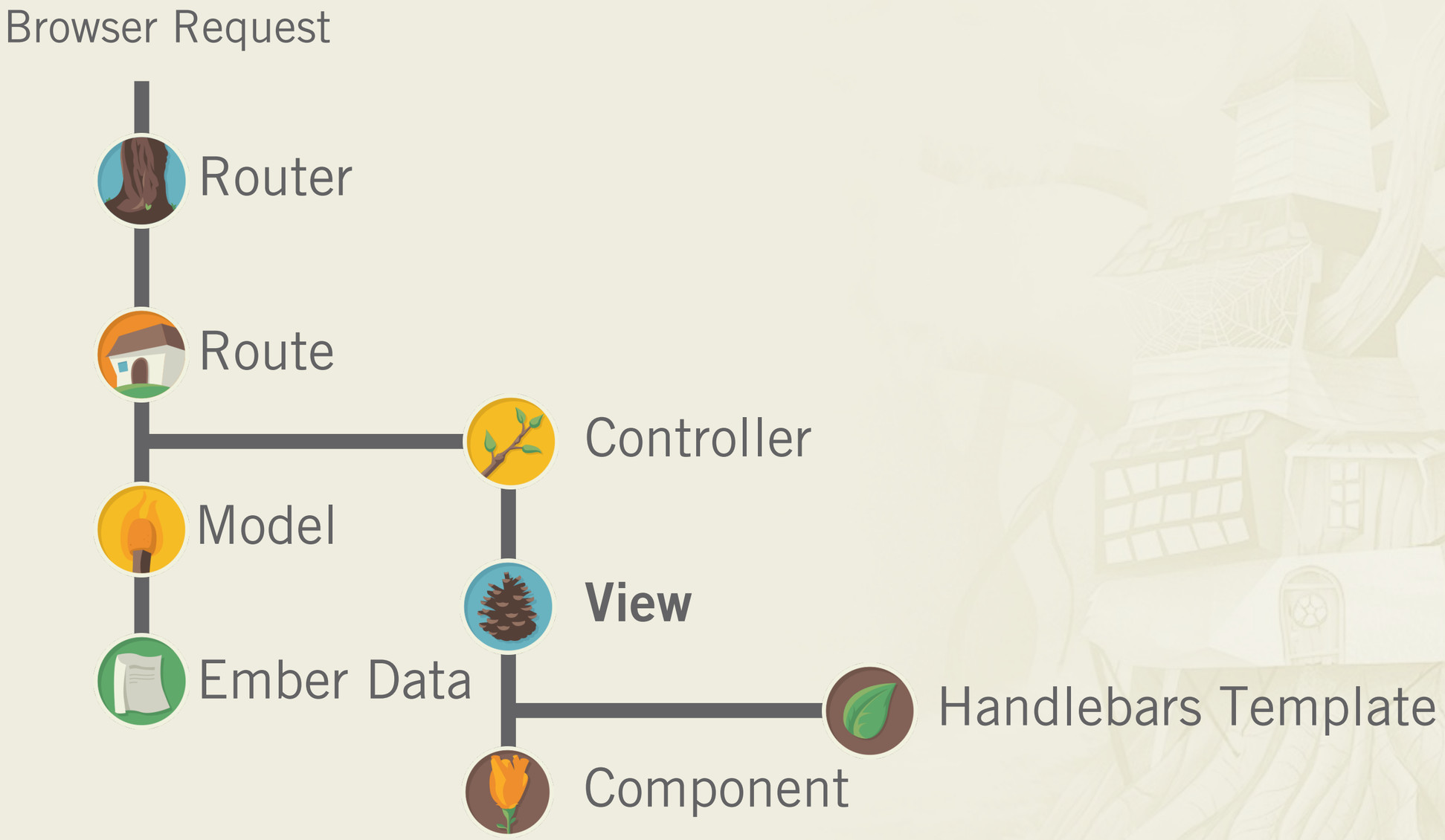
Responsible for encapsulating HTML content

Registering and responding to user-initiated events

Ember Component Extends from Ember View

Each Ember View will use this icon

Ember View


Browser Request
Router
Route
Controller
Model
View
Ember Data
Handlebars Template
Component

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?

output

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

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

The Flint & Flame

localhost:3000/#/products/1

Flint & Flame

Home About Products

div#ember369.ember-view.row.is-on-sale 705px × 256px

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-3">…</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>
```

Styles Computed »

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

… div div div#ember369.ember-view.row.is-on-sale … 11

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>

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

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

Prefix the template name with an underscore

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

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

© 2013 The Flint & Flame

Credits

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

Controller Access in Partials

Product Controller

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

```
data-template-name='_reviews'
```

Both will use the ProductController

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

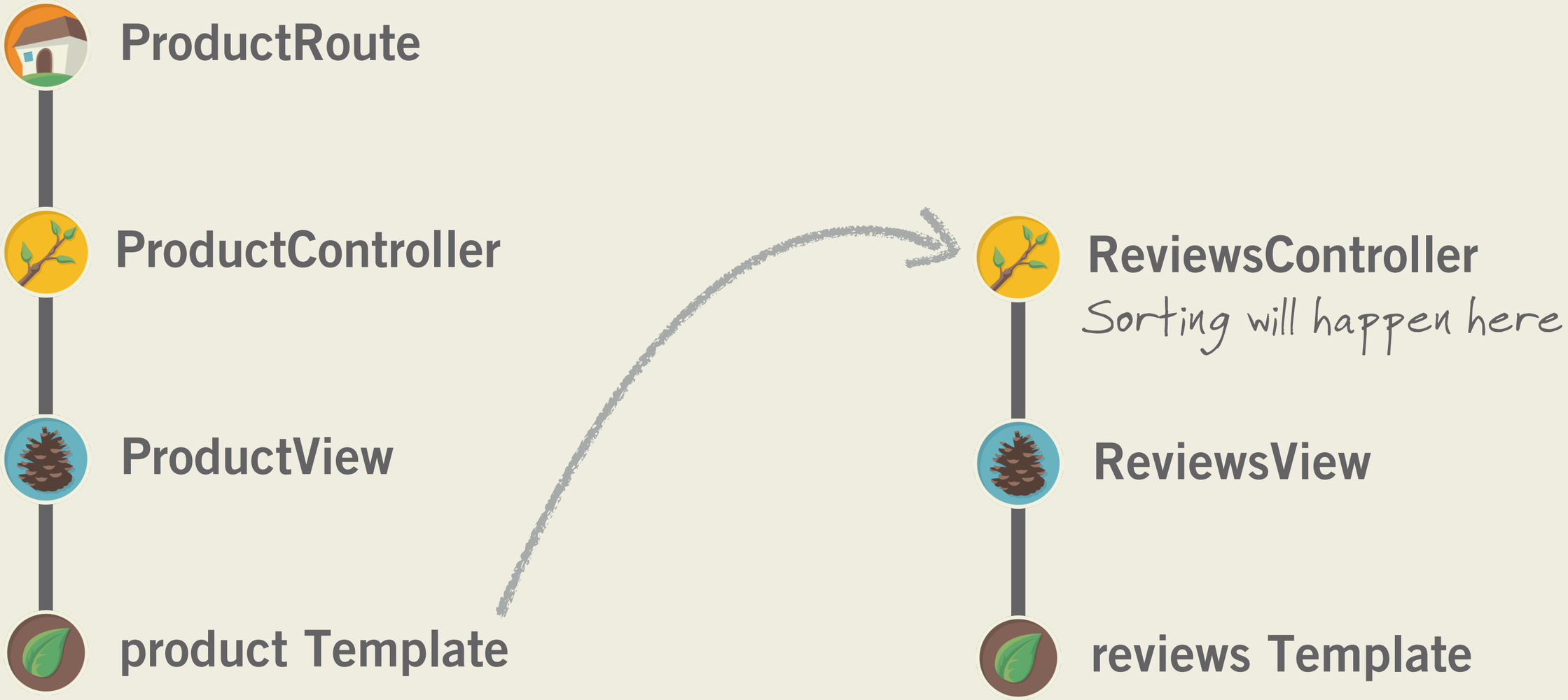
index.html

```
<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}}
</script>

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

The object passed in will become the model

We will create App.ReviewsController to do the sorting

Controller Access with Render

ProductRoute

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

ProductController

ReviewsController

ProductView

ReviewsView

product Template

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'>
```

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!

© 2013 The Flint & Flame

Credits

Showing the newest reviews first

Rendering Differences

Route

- Route
- Controller
- View
- Template

{{render}}

- Controller
- 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>

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

The object passed in will become the model

Controller View Template

{{render}}

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

Let's add the ability to create new reviews!

© 2013 The Flint & Flame

Credits

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'}}
  </div>
</script>
```

Lets print this property out to the screen

Just like any other property

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

© 2013 The Flint & Flame

Credits

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

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> All 11

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()

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

Will add it to our local fixtures. Would do a POST to a server if using the REST adapter

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

Review

© 2013 The Flint & Flame

Credits

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
- Flint
- Kindling
- Matches
- Tinder

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

Review

© 2013 The Flint & Flame

Credits

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

Create a new review object for this product

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

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

Create a new review object for this product

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

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

© 2013 The Flint & Flame

Credits

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')
});
```

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

Ember Data Model States

isNew	Has not been saved
isDirty	Has unsaved changes
isLoading	While fetching from a server
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

© 2013 The Flint & Flame

Credits

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);
      });
    }
  }
});
```

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

- 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');
  }
});
```

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

Ember View Events

`click`	`submit`	`keyDown`
`mouseDown`	`change`	`keyUp`

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

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

Add a class if this is a long review

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

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

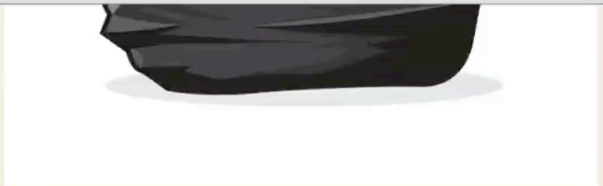
max min

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

Read More

Not the brightest flame, but warm!

Started a fire in no time!

Review Flint

Review

© 2013 The Flint & Flame

Credits