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.

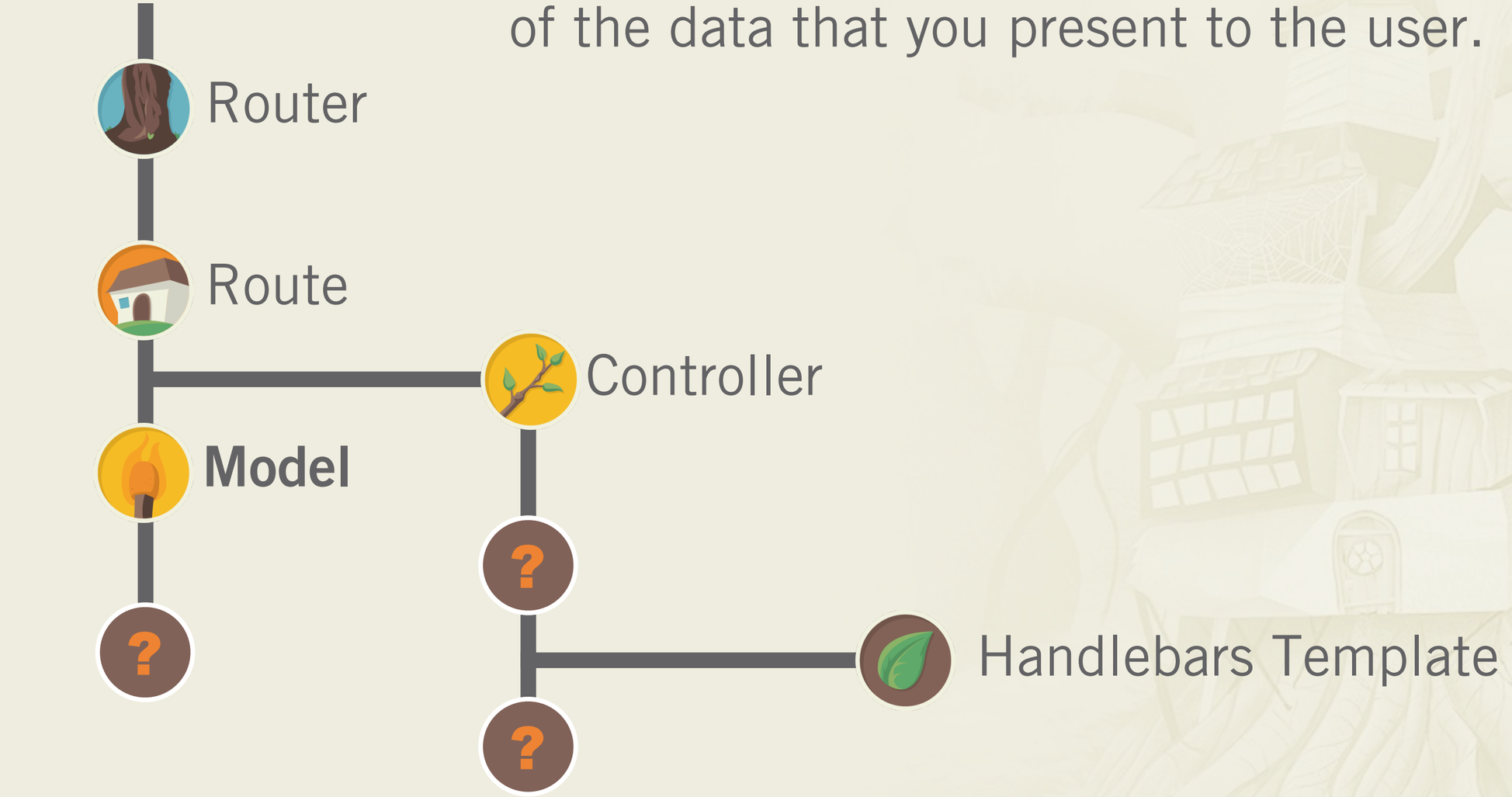
Every Route can have a Model.

Previously our route set the model to an Array.

This is our Model icon

Ember Model

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



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()
});
```

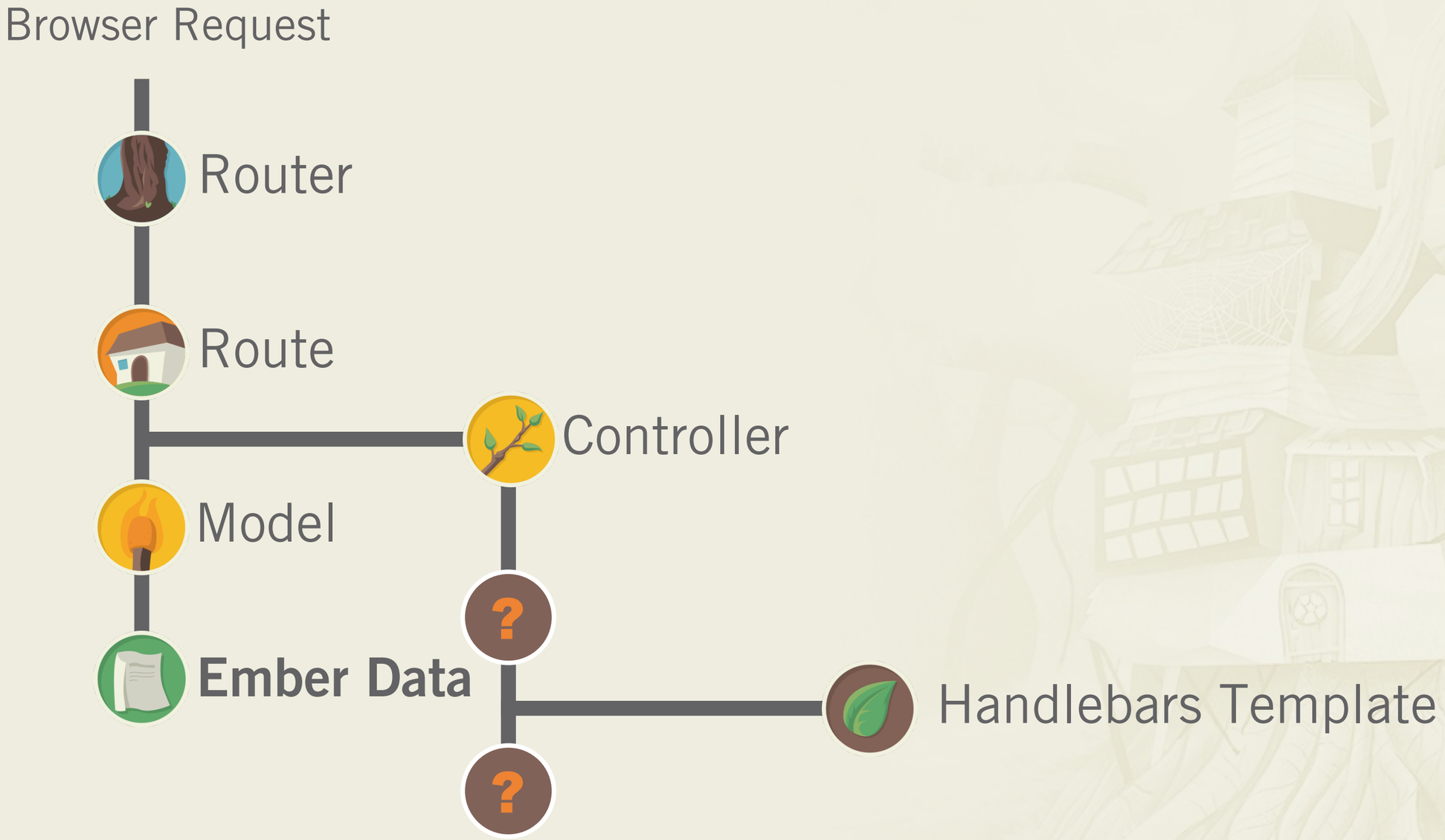
```
{
  title: 'Acorn',
  price: 99,
  description: 'These...' ,
  isOnSale: true,
  isSeasonal: true
}
```

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
Router
Route
Controller
Model
?
Ember Data
Handlebars Template
?

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.

Store

```
App.Product.FIXTURES
```

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);
  }
});
```

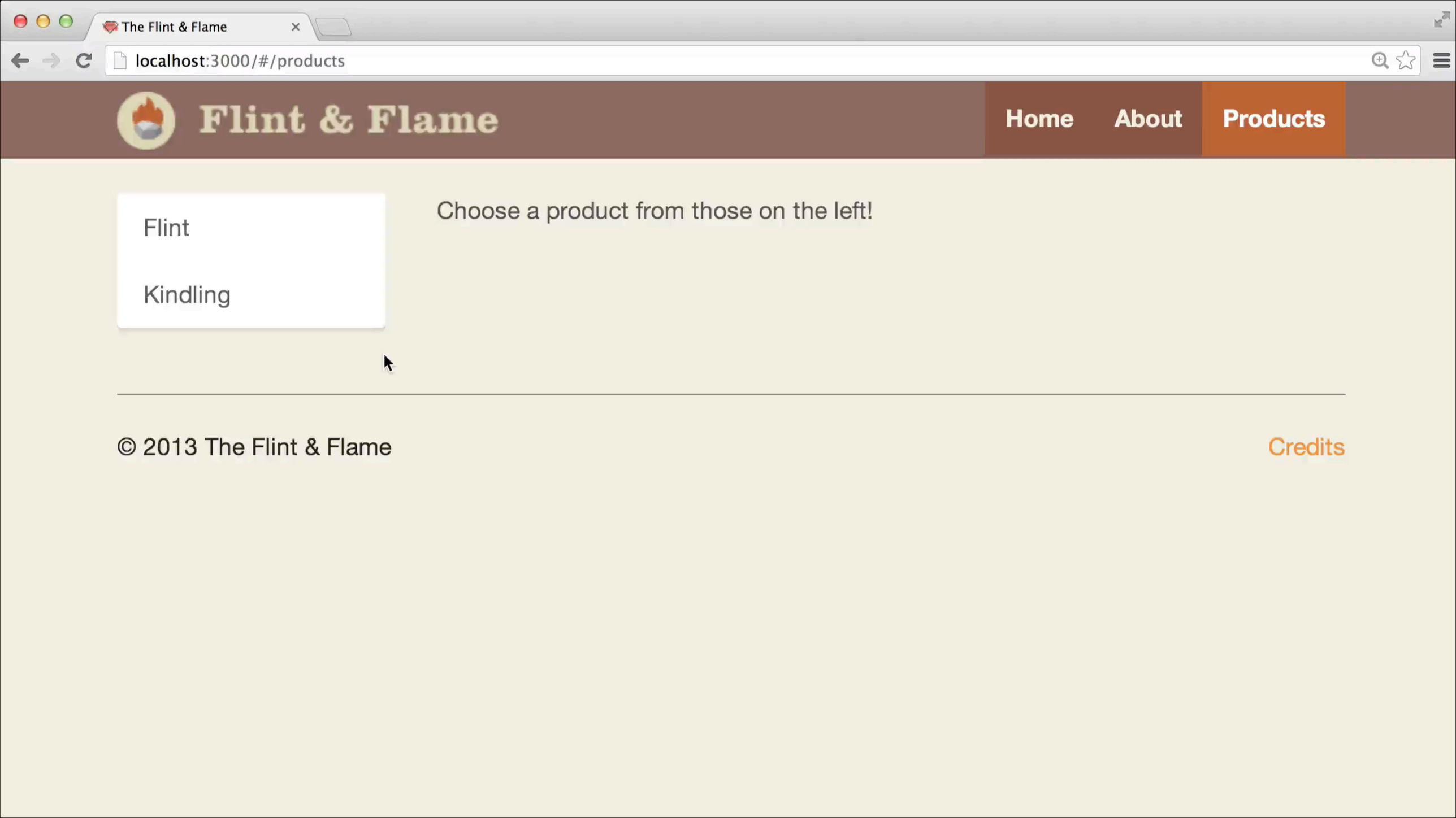
```
App.ProductRoute = Ember.Route.extend({
  model: function(params) {
    return this.store.find('product', params.product_id);
  }
});
```

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!


The Flint & Flame
localhost:3000/#/products
Flint & Flame
Home
About
Products
Flint
Kindling
Choose a product from those on the left!
© 2013 The Flint & Flame
Credits