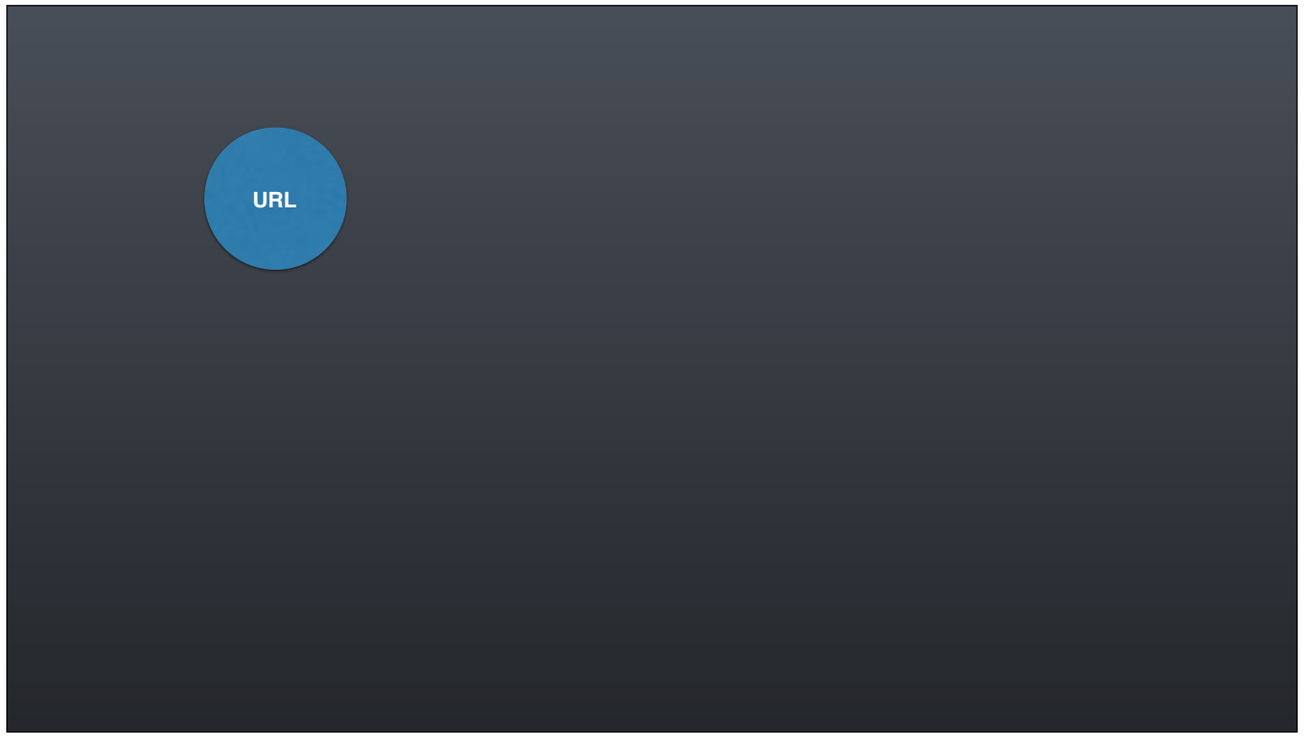

EmberJS
A Fitting Face for a D8 Backend
Taylor Solomon
taylor.solomon
@jtsolomon
http://interactivestrategies.com

2 Years Ago


JS
JS
JS
JS
JS
php
THIS IS FINE.
JS
JS
JS
JS
php

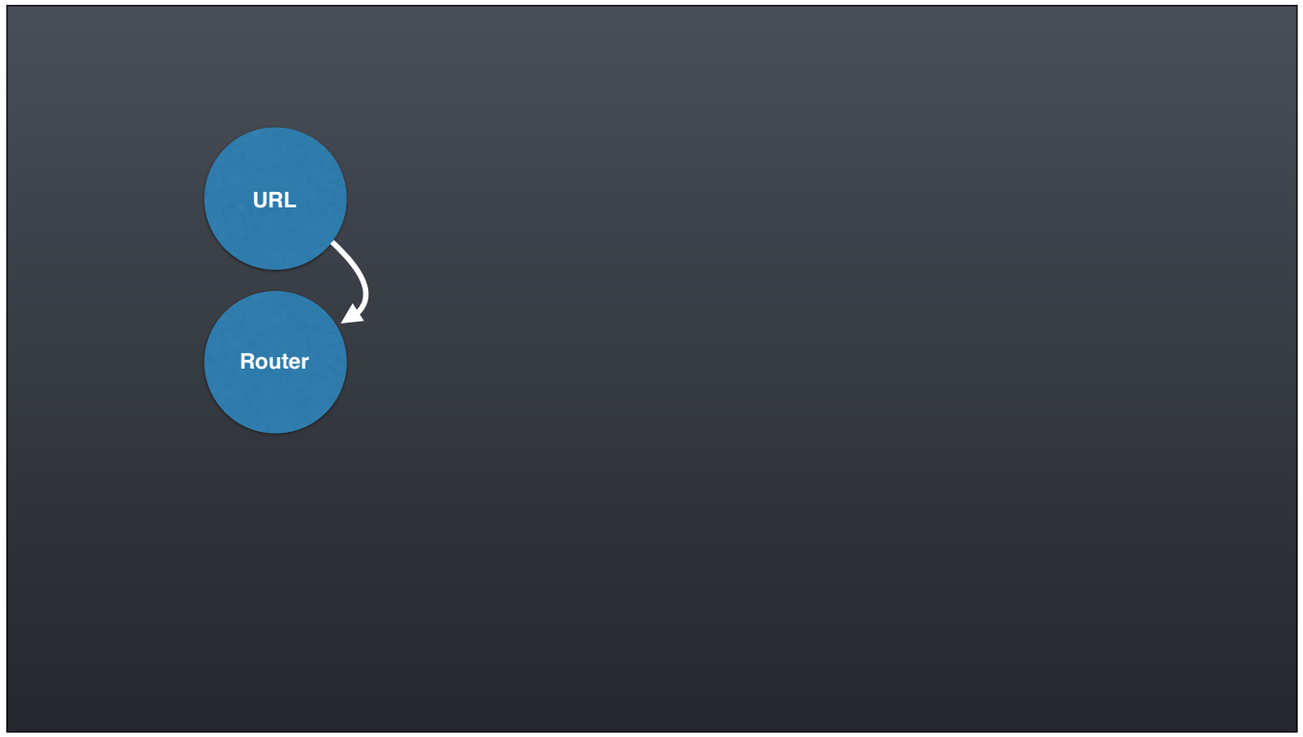

ember



Ember Data assumes a few things.

- Your API format is JSON API
- Your models endpoints are based on the plural form of your model's name.
    - Term => terms
- Check out DS.JSONAPIAdapter here https://emberjs.com/api/data/classes/DS.JSONAPIAdapter.html
- Check out DS.JSONAPISerializer here https://emberjs.com/api/data/classes/DS.JSONAPISerializer.html
    - These are used by default, but might not be in your ember app's structure by default. To override, all you need to do is run `ember generate adapter application` and it will create a new Adapter. You'll see it extends JSONAPIAdapter by default. Same thing for the serializer, but run `ember generate serializer application`

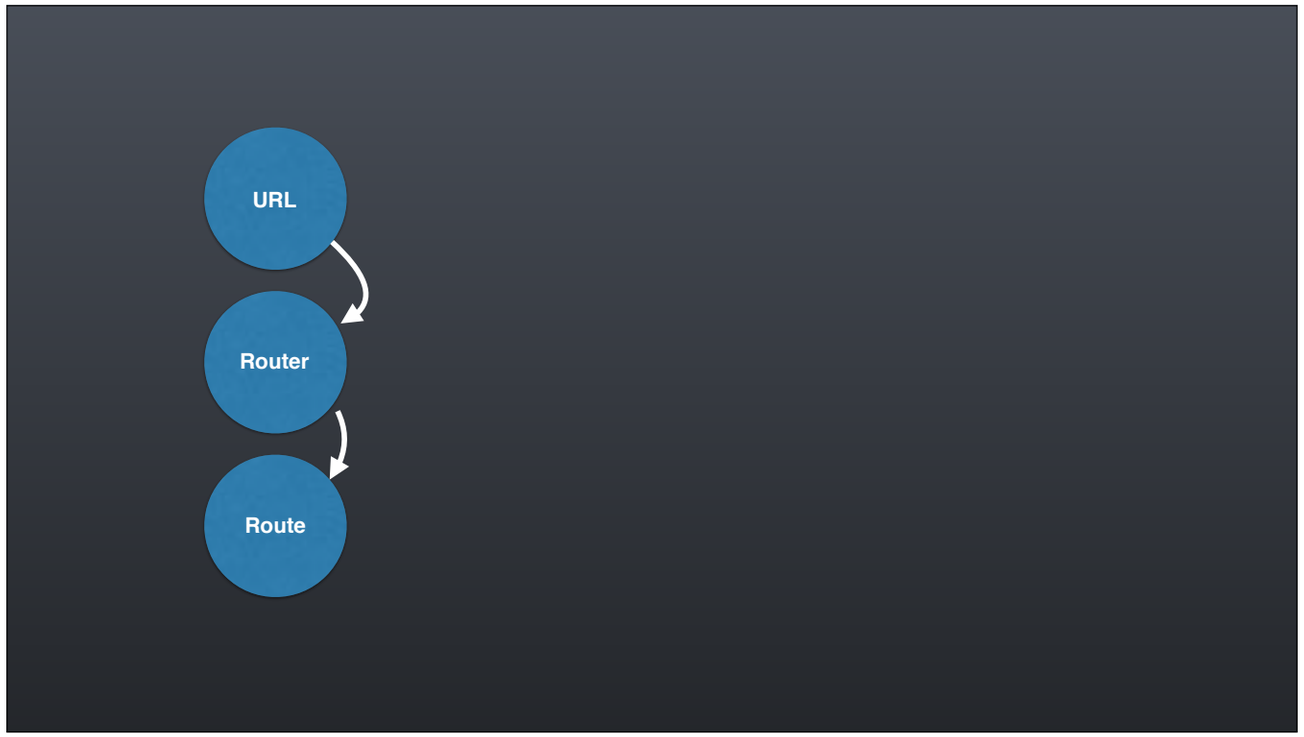
- These defaults respect performance



Ember Data assumes a few things.

- Your API format is JSON API
- Your models endpoints are based on the plural form of your model's name.
    - Term => terms
- Check out DS.JSONAPIAdapter here https://emberjs.com/api/data/classes/DS.JSONAPIAdapter.html
- Check out DS.JSONAPISerializer here https://emberjs.com/api/data/classes/DS.JSONAPISerializer.html
    - These are used by default, but might not be in your ember app's structure by default. To override, all you need to do is run `ember generate adapter application` and it will create a new Adapter. You'll see it extends JSONAPIAdapter by default. Same thing for the serializer, but run `ember generate serializer application`

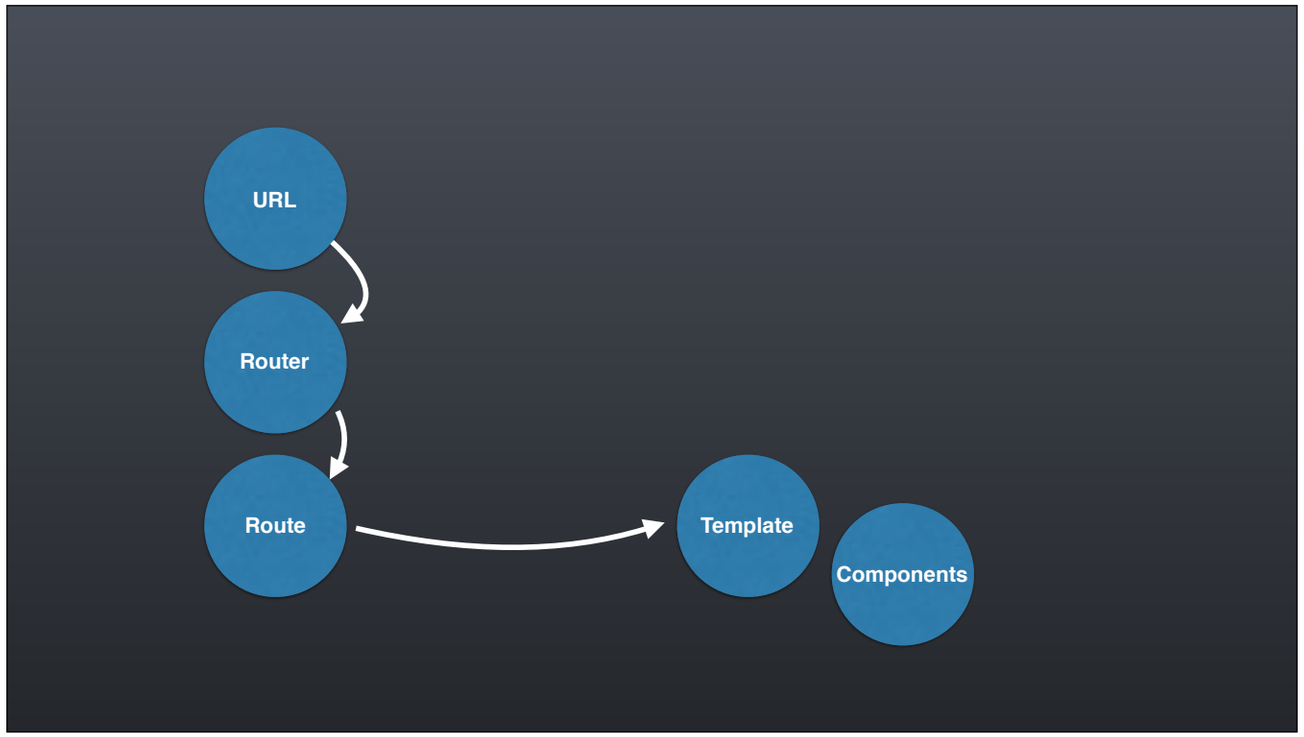
- These defaults respect performance



Ember Data assumes a few things.

- Your API format is JSON API
- Your models endpoints are based on the plural form of your model's name.
    - Term => terms
- Check out DS.JSONAPIAdapter here https://emberjs.com/api/data/classes/DS.JSONAPIAdapter.html
- Check out DS.JSONAPISerializer here https://emberjs.com/api/data/classes/DS.JSONAPISerializer.html
    - These are used by default, but might not be in your ember app's structure by default. To override, all you need to do is run `ember generate adapter application` and it will create a new Adapter. You'll see it extends JSONAPIAdapter by default. Same thing for the serializer, but run `ember generate serializer application`

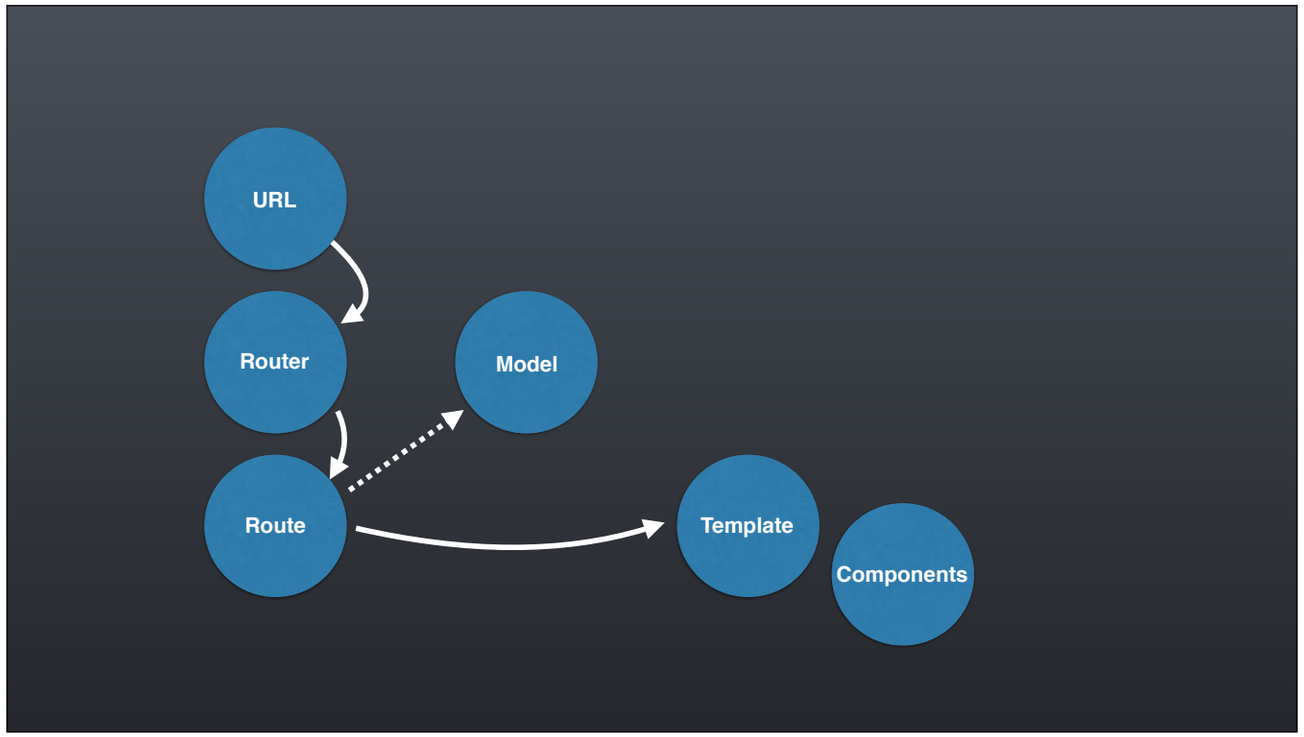
- These defaults respect performance



Ember Data assumes a few things.

- Your API format is JSON API
- Your models endpoints are based on the plural form of your model's name.
    - Term => terms
- Check out DS.JSONAPIAdapter here https://emberjs.com/api/data/classes/DS.JSONAPIAdapter.html
- Check out DS.JSONAPISerializer here https://emberjs.com/api/data/classes/DS.JSONAPISerializer.html
    - These are used by default, but might not be in your ember app's structure by default. To override, all you need to do is run `ember generate adapter application` and it will create a new Adapter. You'll see it extends JSONAPIAdapter by default. Same thing for the serializer, but run `ember generate serializer application`

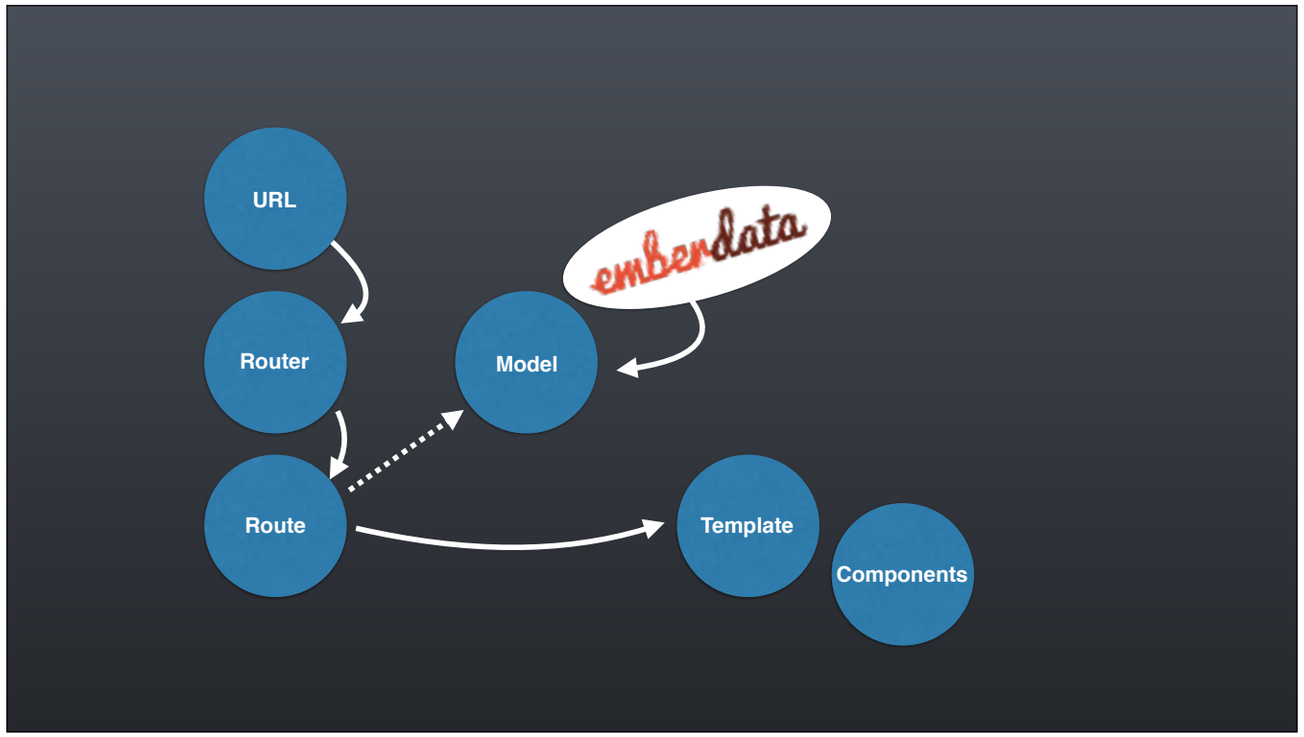
- These defaults respect performance



Ember Data assumes a few things.

- Your API format is JSON API
- Your models endpoints are based on the plural form of your model's name.
    - Term => terms
- Check out DS.JSONAPIAdapter here https://emberjs.com/api/data/classes/DS.JSONAPIAdapter.html
- Check out DS.JSONAPISerializer here https://emberjs.com/api/data/classes/DS.JSONAPISerializer.html
    - These are used by default, but might not be in your ember app's structure by default. To override, all you need to do is run `ember generate adapter application` and it will create a new Adapter. You'll see it extends JSONAPIAdapter by default. Same thing for the serializer, but run `ember generate serializer application`

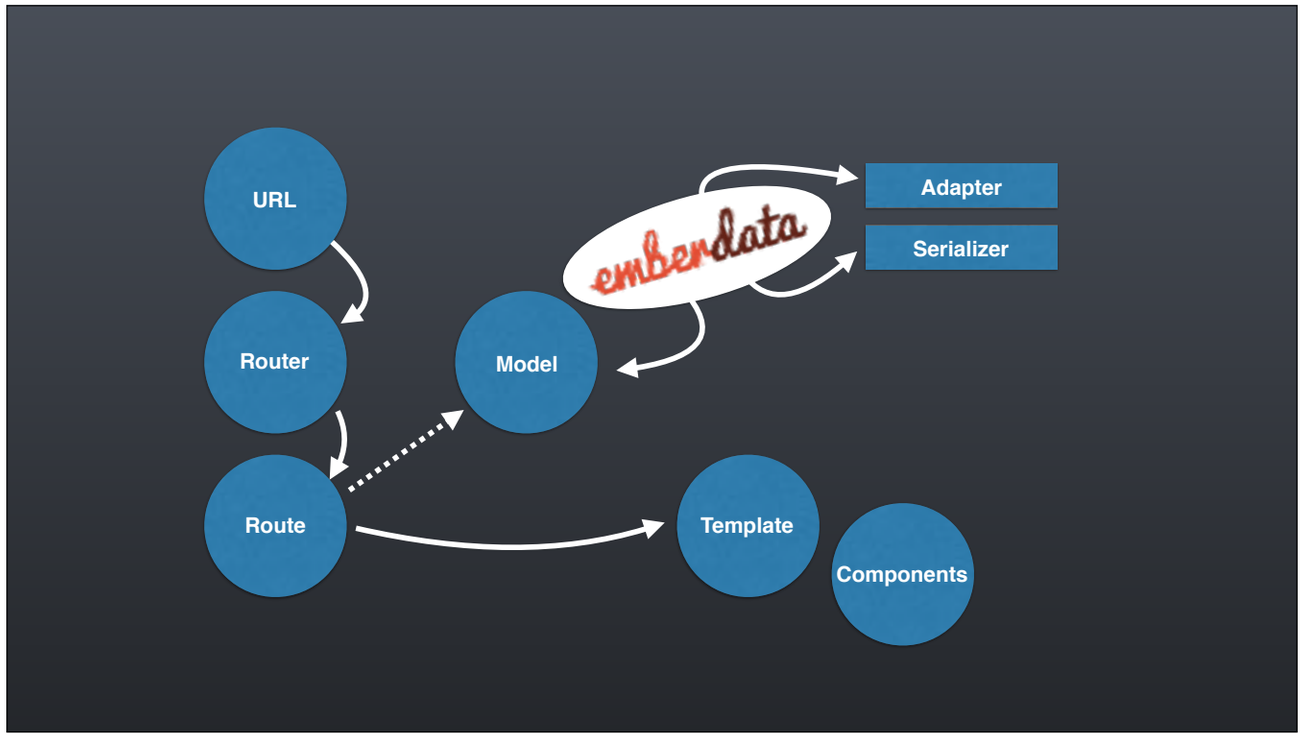
- These defaults respect performance



Ember Data assumes a few things.

- Your API format is JSON API
- Your models endpoints are based on the plural form of your model's name.
    - Term => terms
- Check out DS.JSONAPIAdapter here https://emberjs.com/api/data/classes/DS.JSONAPIAdapter.html
- Check out DS.JSONAPISerializer here https://emberjs.com/api/data/classes/DS.JSONAPISerializer.html
    - These are used by default, but might not be in your ember app's structure by default. To override, all you need to do is run `ember generate adapter application` and it will create a new Adapter. You'll see it extends JSONAPIAdapter by default. Same thing for the serializer, but run `ember generate serializer application`

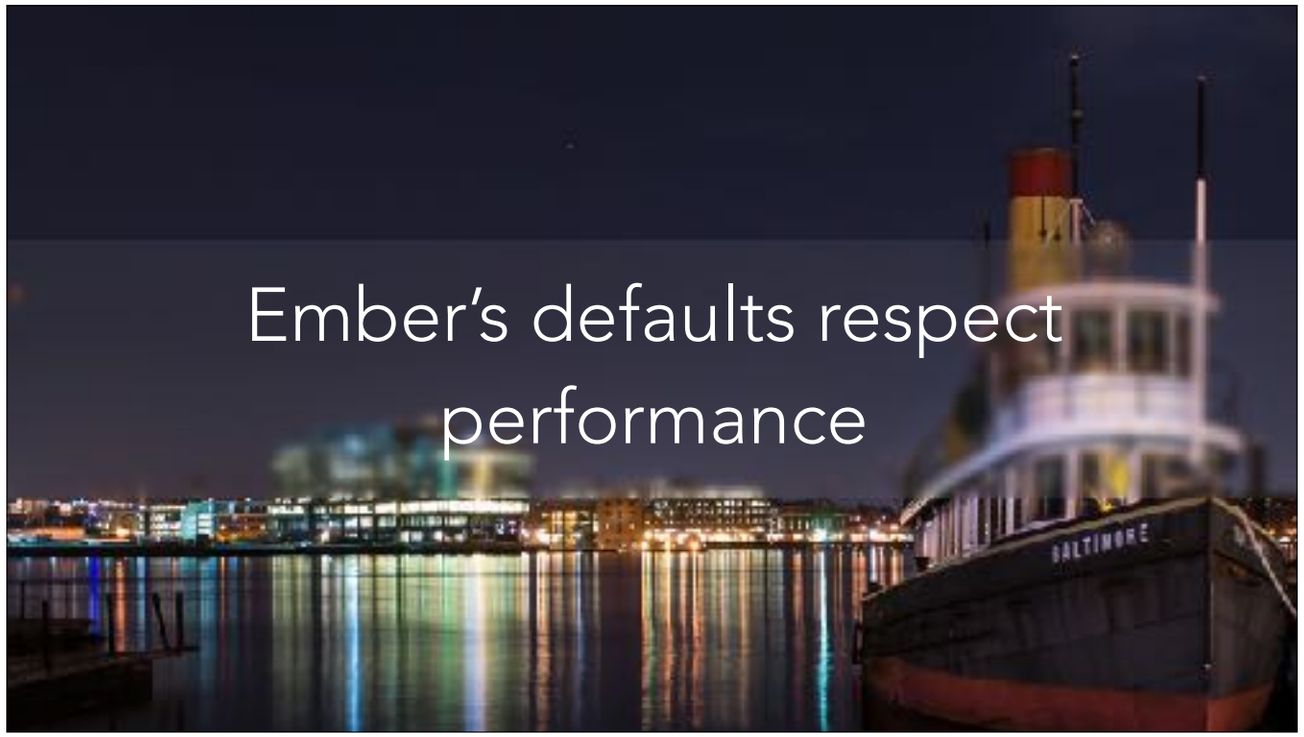
- These defaults respect performance



Ember Data assumes a few things.

- Your API format is JSON API
- Your models endpoints are based on the plural form of your model's name.
    - Term => terms
- Check out DS.JSONAPIAdapter here https://emberjs.com/api/data/classes/DS.JSONAPIAdapter.html
- Check out DS.JSONAPISerializer here https://emberjs.com/api/data/classes/DS.JSONAPISerializer.html
    - These are used by default, but might not be in your ember app's structure by default. To override, all you need to do is run `ember generate adapter application` and it will create a new Adapter. You'll see it extends JSONAPIAdapter by default. Same thing for the serializer, but run `ember generate serializer application`

- These defaults respect performance

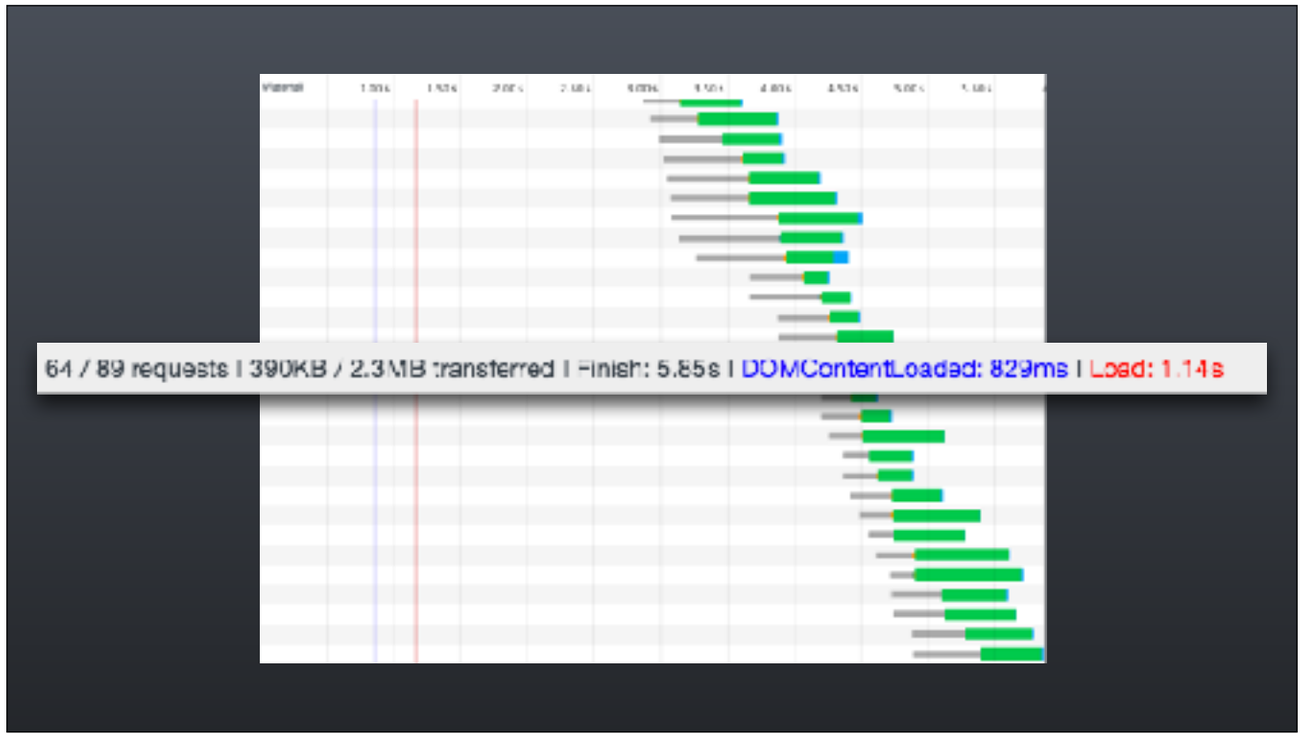


- JSON API is designed to minimize
  - Number of requests
  - The amount of data transmitted between clients and servers.
- Not sacrificing readability

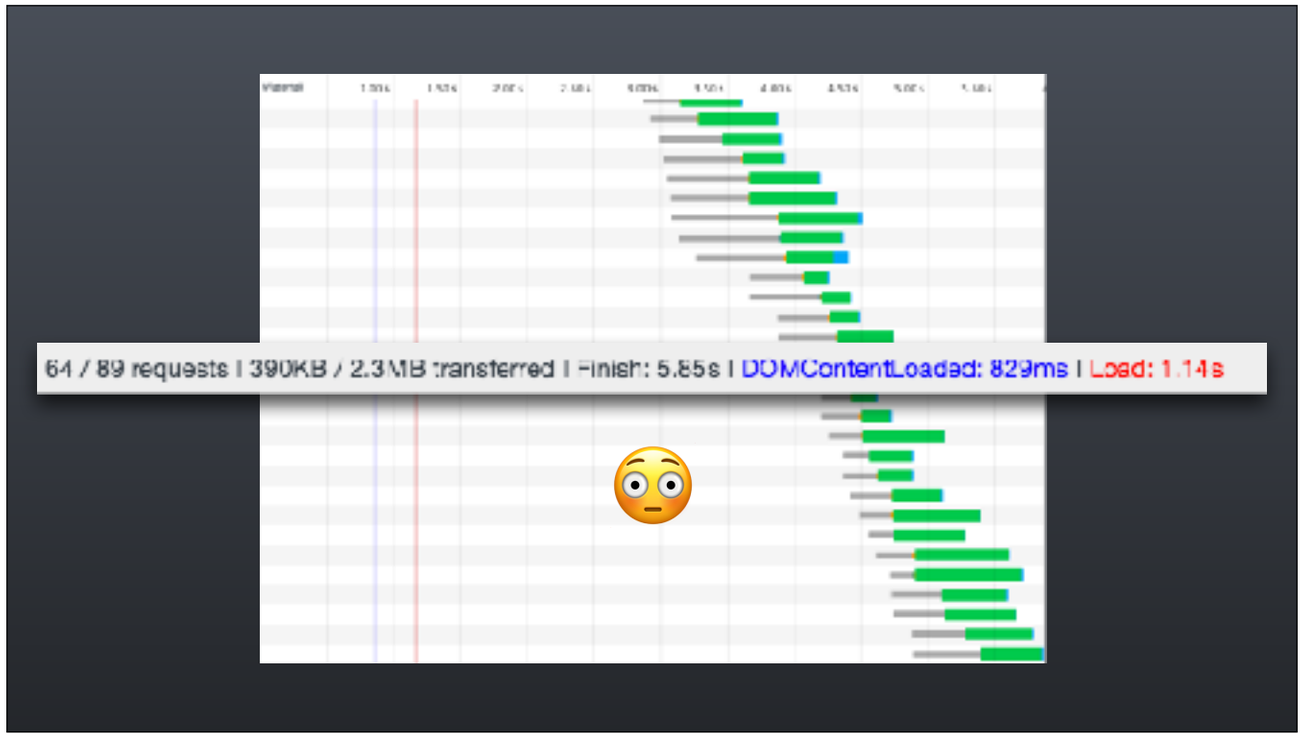
JSON API

- id, type
- attributes
- relationships
- meta
- errors

- Relationships are your Entity References
- By default, Ember Data will know how to find out more about this author guy
- Pretty great, right? So we're really stoked on this, and we build a page with lots of relationships and data in those relationships, and it works! But then we saw how long it took to render out all that data.



- Smart, but slow
- JSON API is designed to minimize requests, namely through allowing additional data.
- This is without gzip, btw.



- Smart, but slow
- JSON API is designed to minimize requests, namely through allowing additional data.
- This is without gzip, btw.

JSON API

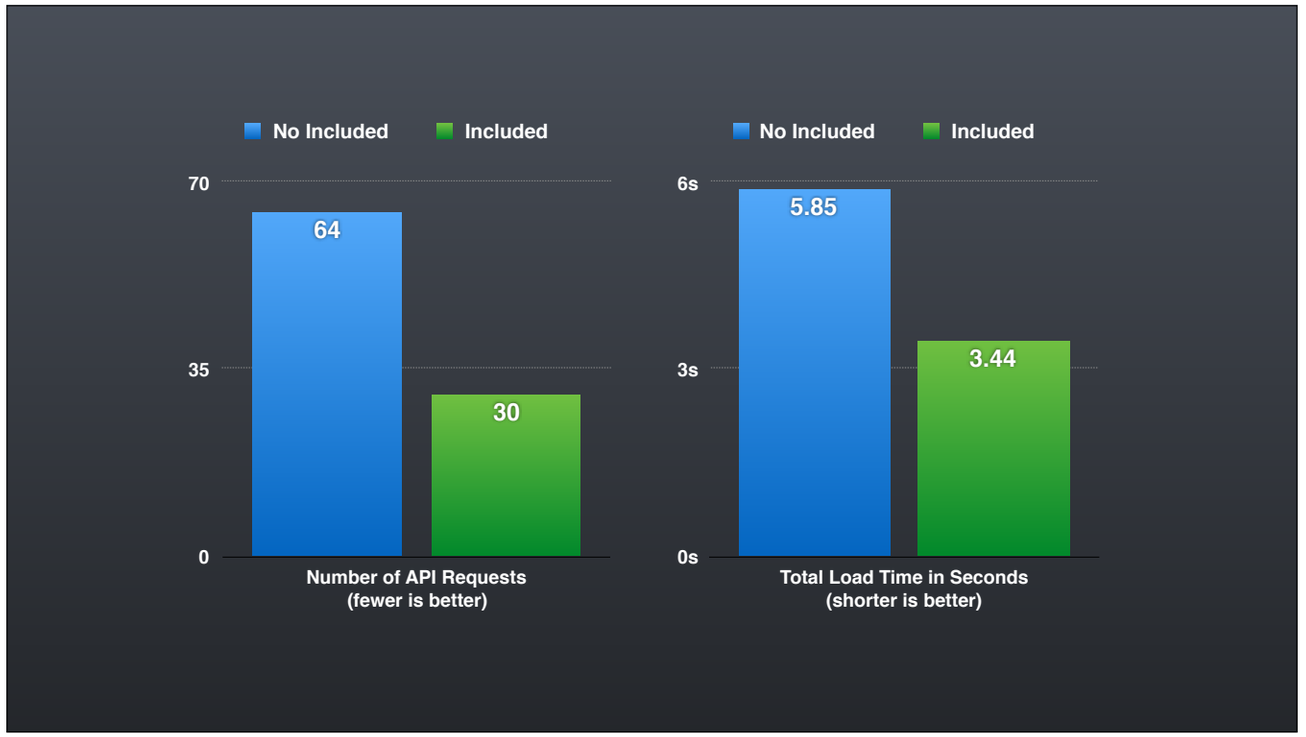
- id, type
- attributes
- relationships
- meta
- errors

- Relationship in the article is thin, but included[] can help define more of that stuff in the relationships
- So that’s what we did

JSON API

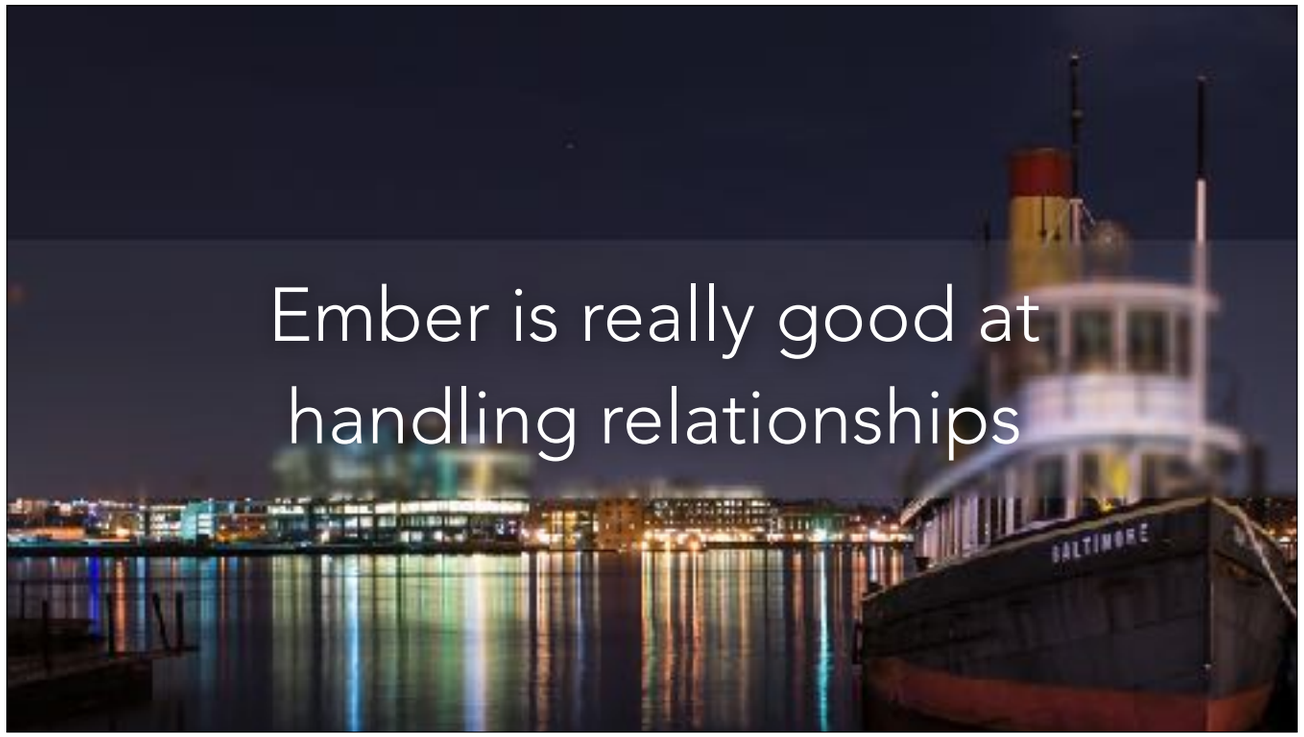
- id, type
- attributes
- relationships
- meta
- errors

- Relationship in the article is thin, but included[] can help define more of that stuff in the relationships
- So that’s what we did

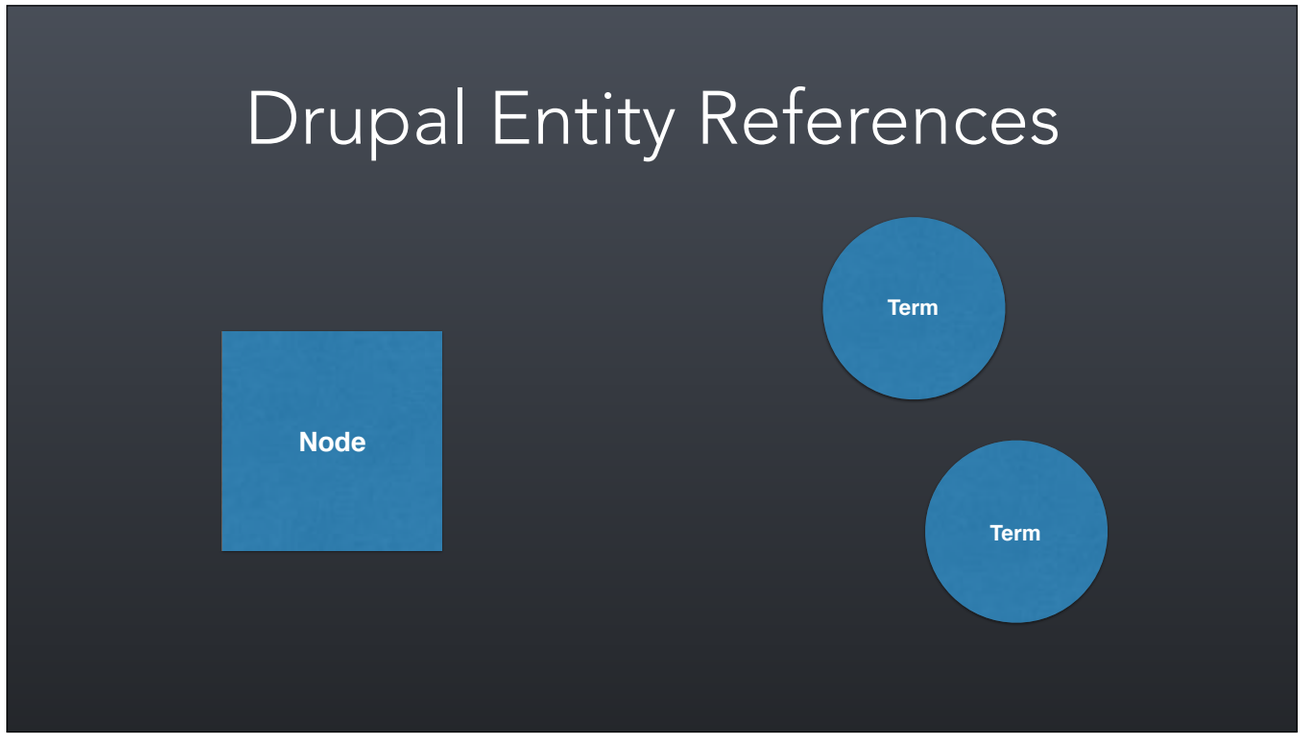


Instead of 64 requests, we only needed to make 30

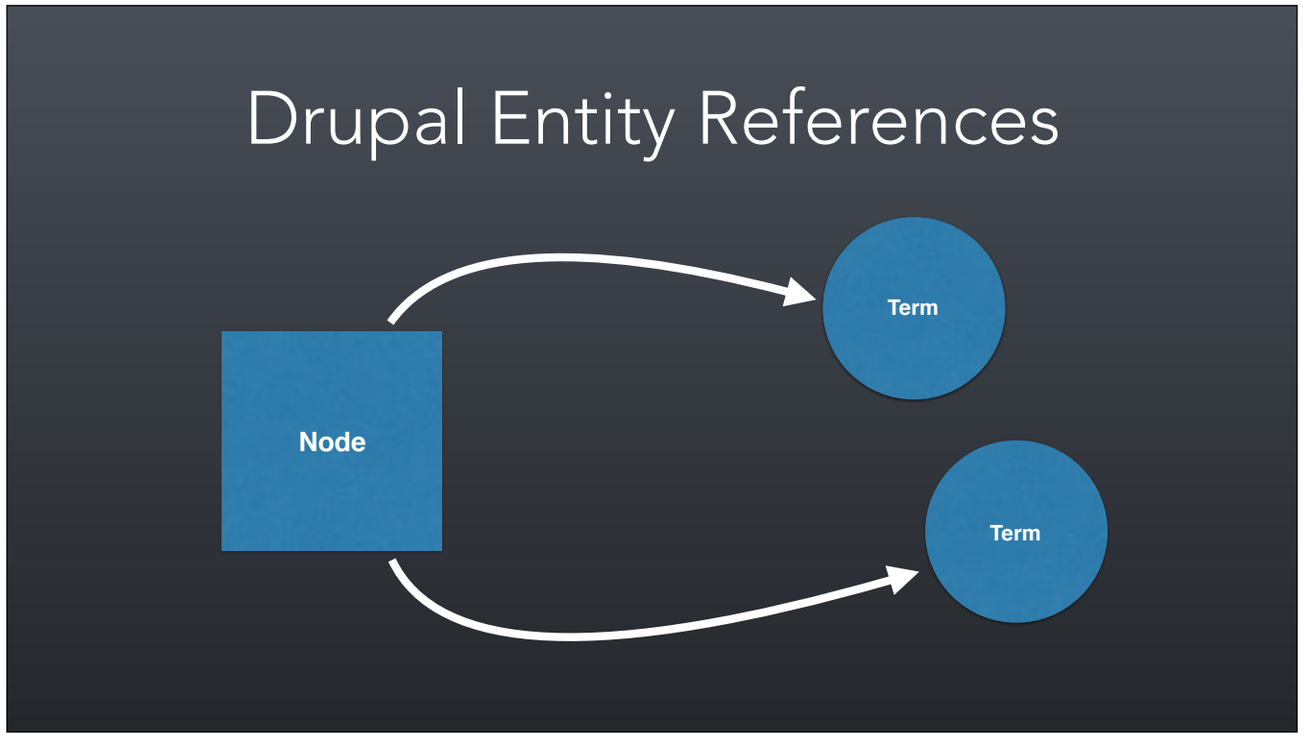
So Ember respects performance through its handling of relationships, but it goes beyond just performance.



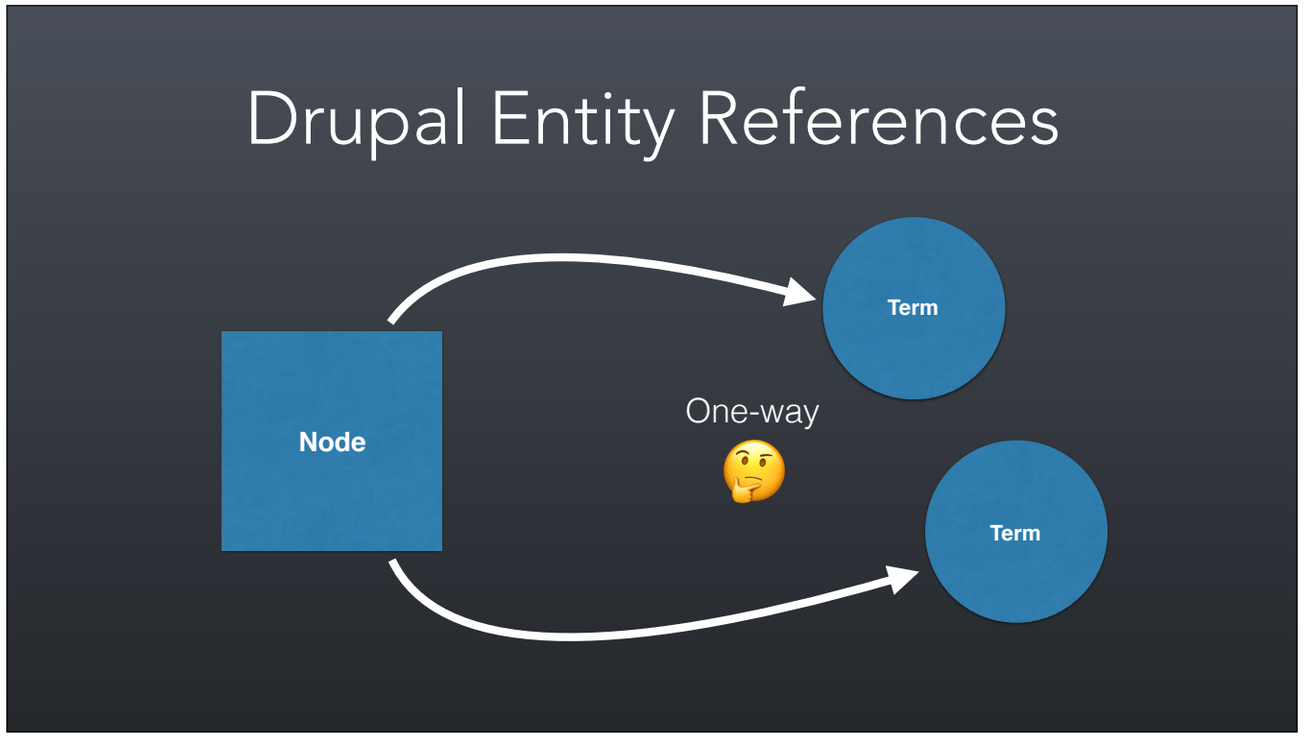
- Beyond performance, Ember allows you to deal with relationships in ways that mean fewer lines of code in your API. It helped us address requirements that would have been complicated to do in just Drupal.
- Drupal is still good at relationships



- Drupal is pretty good at relationships
    - Entityreference is in core
- Key characteristic of entity reference fields is being one way.
- Sometimes you'll need to compute that reverse relationship. Starting with the object rather than the subject.
- How could we do this on the API side?



- Drupal is pretty good at relationships
    - Entityreference is in core
- Key characteristic of entity reference fields is being one way.
- Sometimes you'll need to compute that reverse relationship. Starting with the object rather than the subject.
- How could we do this on the API side?



- Drupal is pretty good at relationships
    - Entityreference is in core
- Key characteristic of entity reference fields is being one way.
- Sometimes you'll need to compute that reverse relationship. Starting with the object rather than the subject.
- How could we do this on the API side?

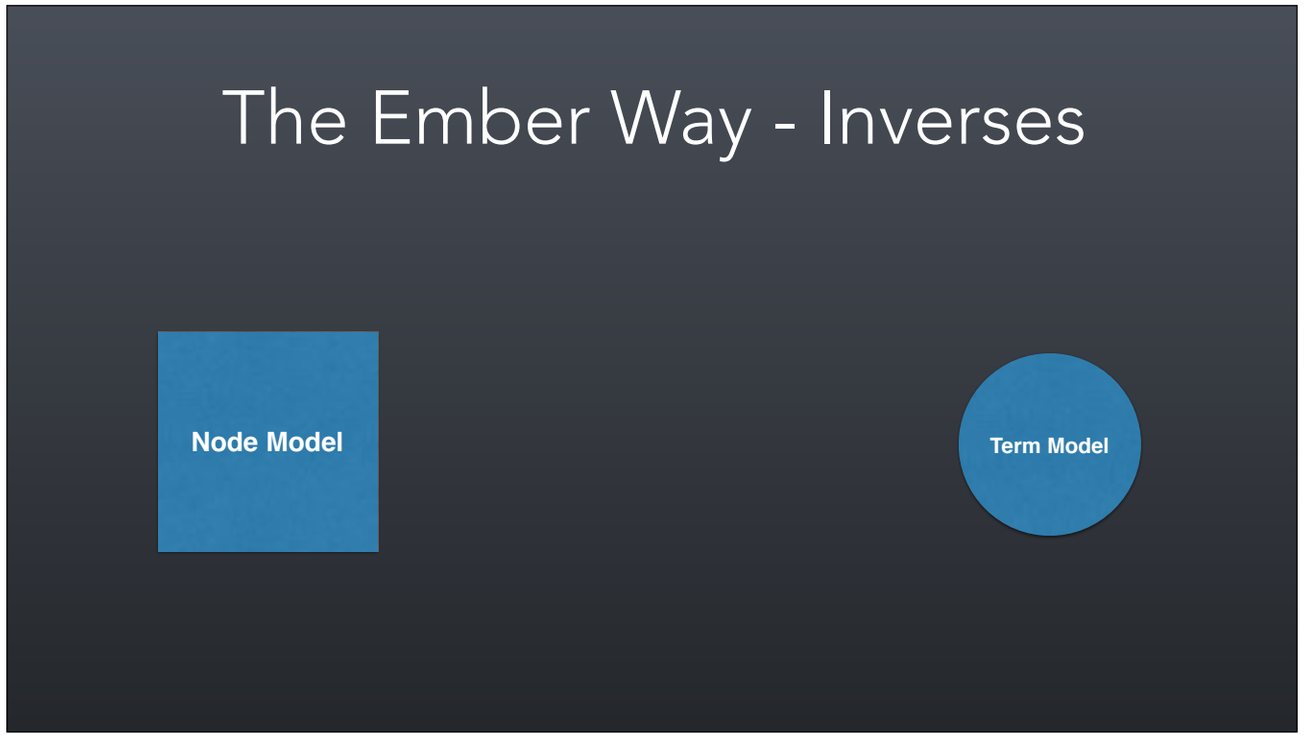
1. Send the reverse relationship in the response.

2. Have Ember query the API for relationship data.

- If our node defines a relationship to a term, maybe our term Response defines that reverse relationship to all its tagged nodes. But that info isn't on the term already.
  - EntityQuery
- But there's a better way.

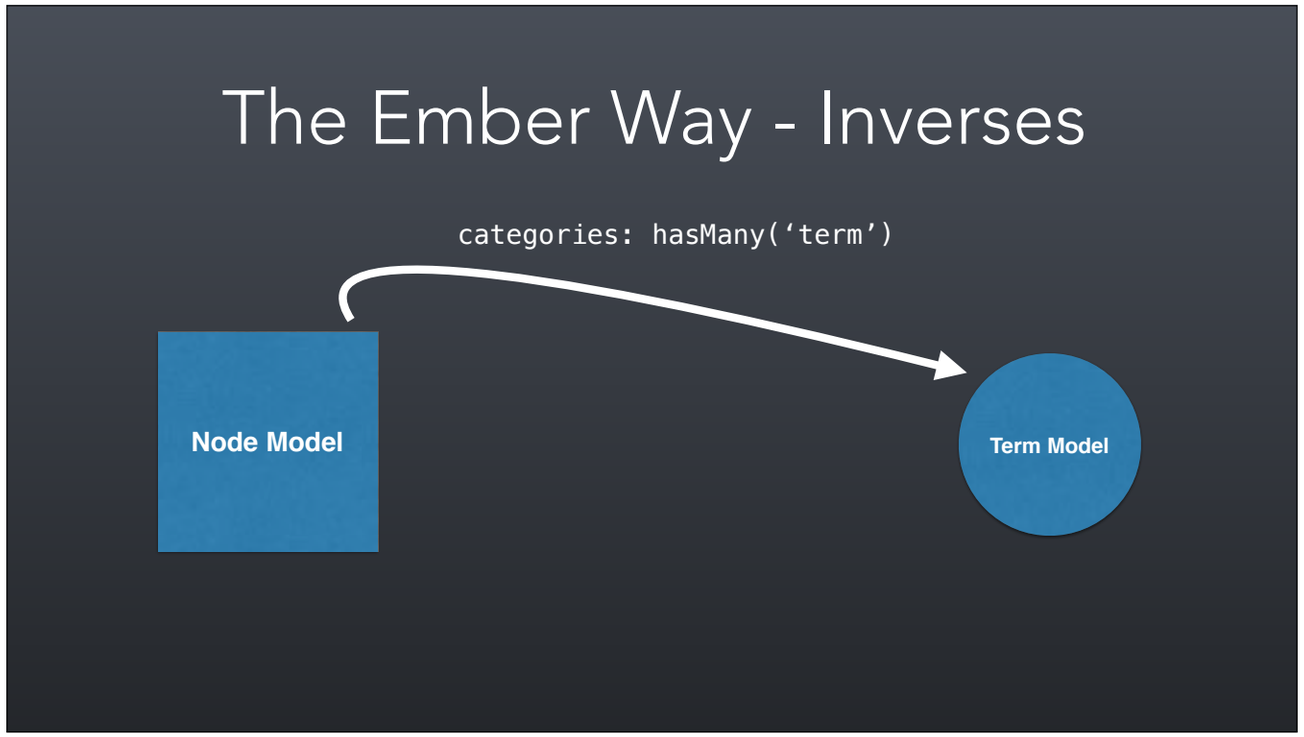
Additionally:

- There was a module in D7 called Relation that created an entity that contained a two way relationship between two other entities. You could use the D8 version I guess to accomplish this, but Ember worked for us.



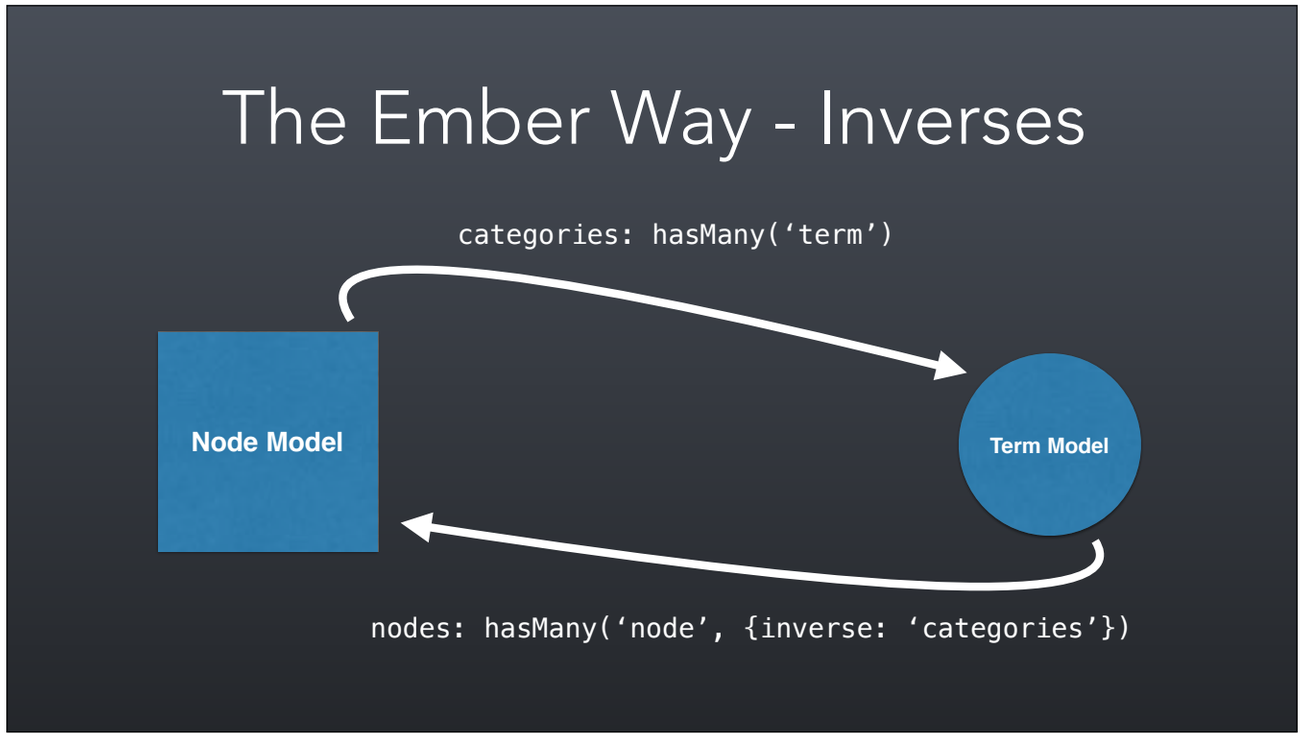
- Can tell Ember that your term model is going to have nodes associated with it.
- If you load 6 nodes all tagged with the same term, you can get that term and using a property defined with an inverse, you can act on that list of 6 nodes as if they were attached to your term.
- Ember can compute that without your data even being in a payload.

Check out https://guides.emberjs.com/v2.12.0/models/relationships/ for more



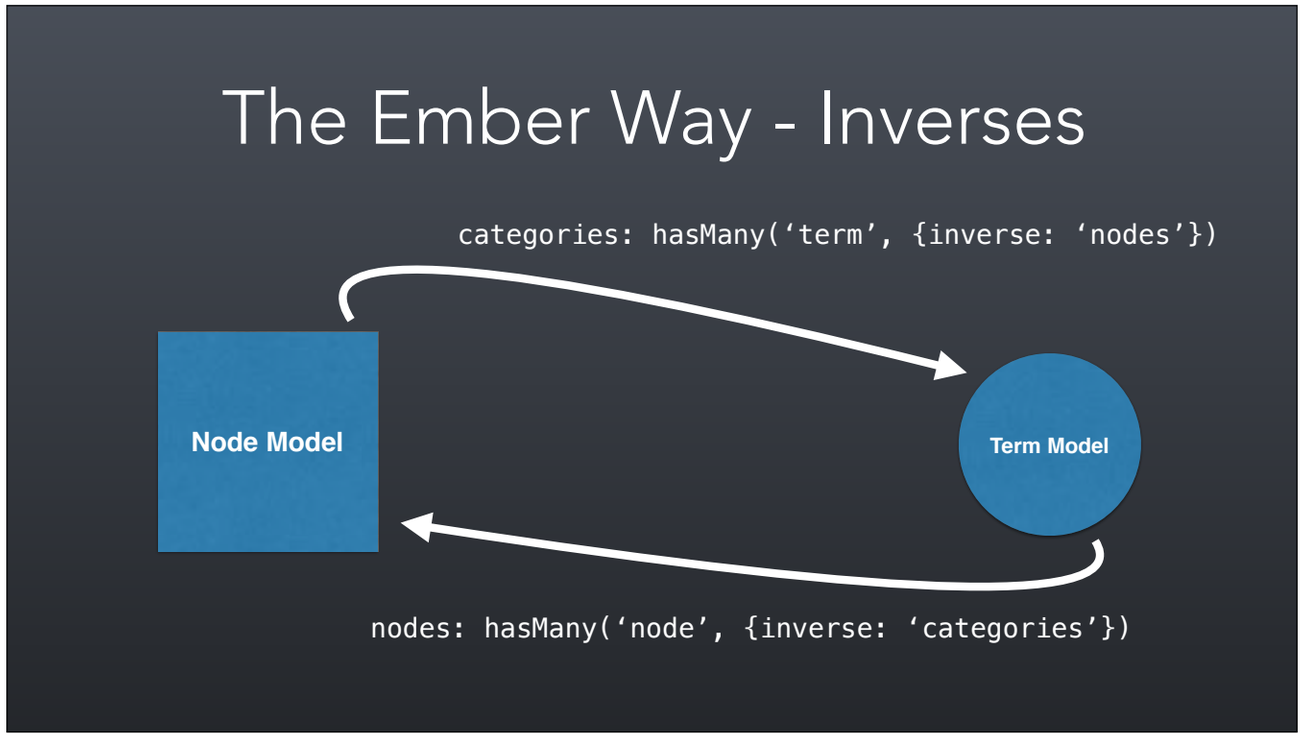
- Can tell Ember that your term model is going to have nodes associated with it.
- If you load 6 nodes all tagged with the same term, you can get that term and using a property defined with an inverse, you can act on that list of 6 nodes as if they were attached to your term.
- Ember can compute that without your data even being in a payload.

Check out https://guides.emberjs.com/v2.12.0/models/relationships/ for more



- Can tell Ember that your term model is going to have nodes associated with it.
- If you load 6 nodes all tagged with the same term, you can get that term and using a property defined with an inverse, you can act on that list of 6 nodes as if they were attached to your term.
- Ember can compute that without your data even being in a payload.

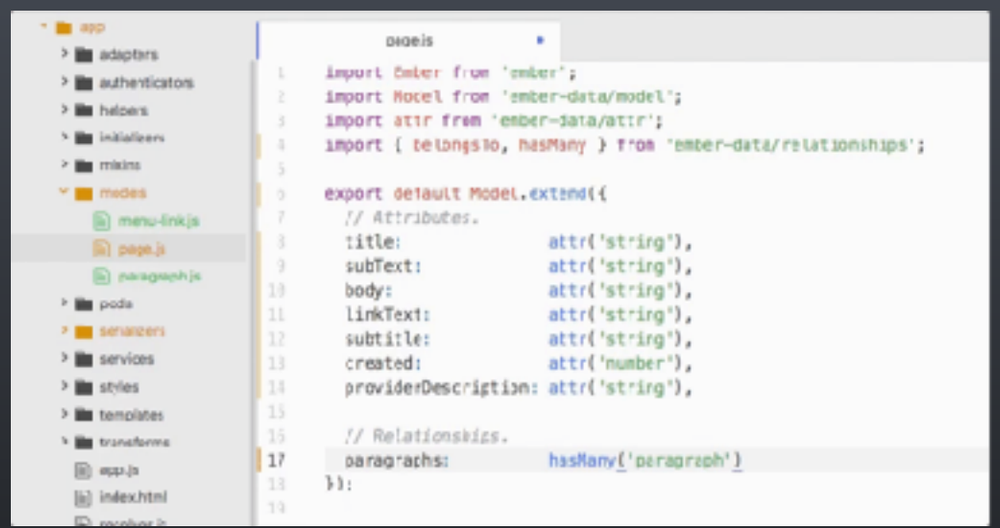
Check out https://guides.emberjs.com/v2.12.0/models/relationships/ for more



- Can tell Ember that your term model is going to have nodes associated with it.
- If you load 6 nodes all tagged with the same term, you can get that term and using a property defined with an inverse, you can act on that list of 6 nodes as if they were attached to your term.
- Ember can compute that without your data even being in a payload.

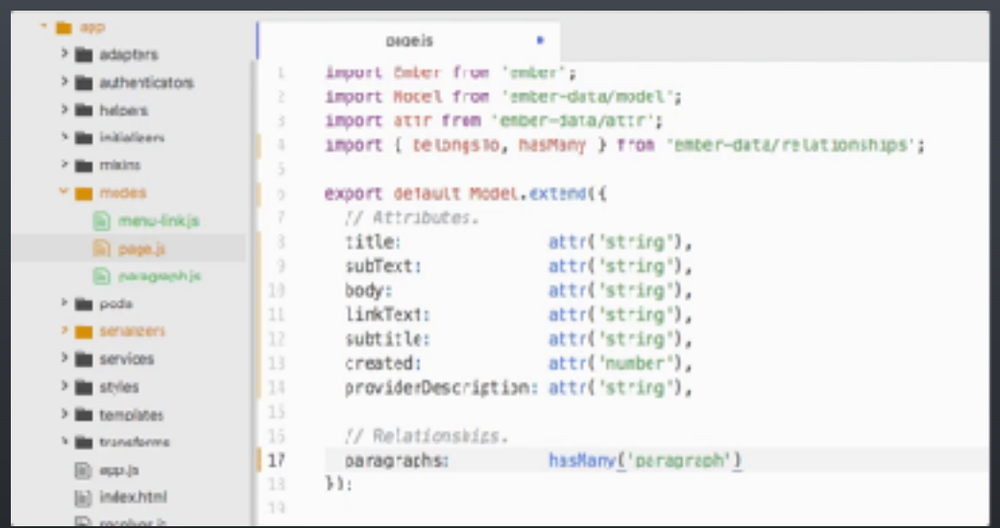
Check out https://guides.emberjs.com/v2.12.0/models/relationships/ for more

Ember Data can support relationships *not* in a payload



- This gives you the ability to do cool things on your front-end that Drupal isn’t set up to do by default.
- Like see what page a certain paragraph is attached to.
- There is no ui for seeing the paragraph entity reference from the perspective of the paragraph in Drupal.

Ember Data can support relationships *not* in a payload



- This gives you the ability to do cool things on your front-end that Drupal isn’t set up to do by default.
- Like see what page a certain paragraph is attached to.
- There is no ui for seeing the paragraph entity reference from the perspective of the paragraph in Drupal.

Ember Data can support reflexive relationships, too

```
import Model from 'ember-data/model';
import attr from 'ember-data/attr';
import { belongsTo, hasMany } from 'ember-data/relationships';

export default Model.extend({
  // Attributes.
  title:    attr('string'),
  link:     attr(),

  // Relationships.
  parent:   belongsTo('menu-link', {inverse: 'children'}),
  children: hasMany('menu-link', {inverse: 'parent'})
});

```

- An Entity relating to its own kind, like menu items.
- Caveat with this - Won't know about stuff you haven't brought over from your API.
    - Got away with this cause we load all our menu items from the API when we boot our Ember app.
    - Otherwise it will need to make requests to the API to gather more information about relationships.

- So this is great. Ember has some robust relationship handling built in.
- Until now, we haven't really talked about static data in Models
    - Like titles, or nids, or created date. Those are static data.
- Ember can cast those to certain primitives like string, number out of the box. But what about the odd stuff? Every API has odd stuff.

Ember Data can support reflexive relationships, too

```
-data/model';
data/attr';
any } from 'ember-data/relationships';

end({

g'),



menu-link', {inverse: 'children'}),
u-link', {inverse: 'parent'})
```

template.hbs

```
<div class="menu-wrapper">
  {{#each mainMenu.children as |levelOne|}}
    <ul>
      {{#link-to params=levelOne.params}}
        <h6>{{levelOne.title}}</h6>
      {{/link-to}}

      {{#each levelOne.children as |levelTwo|}}
        <li>
          {{#link-to params=levelTwo.params}}
            {{help.title}}
          {{/link-to}}
        </li>
      {{/each}}
    </ul>
  {{/each}}
</div>

{{yield}}
```

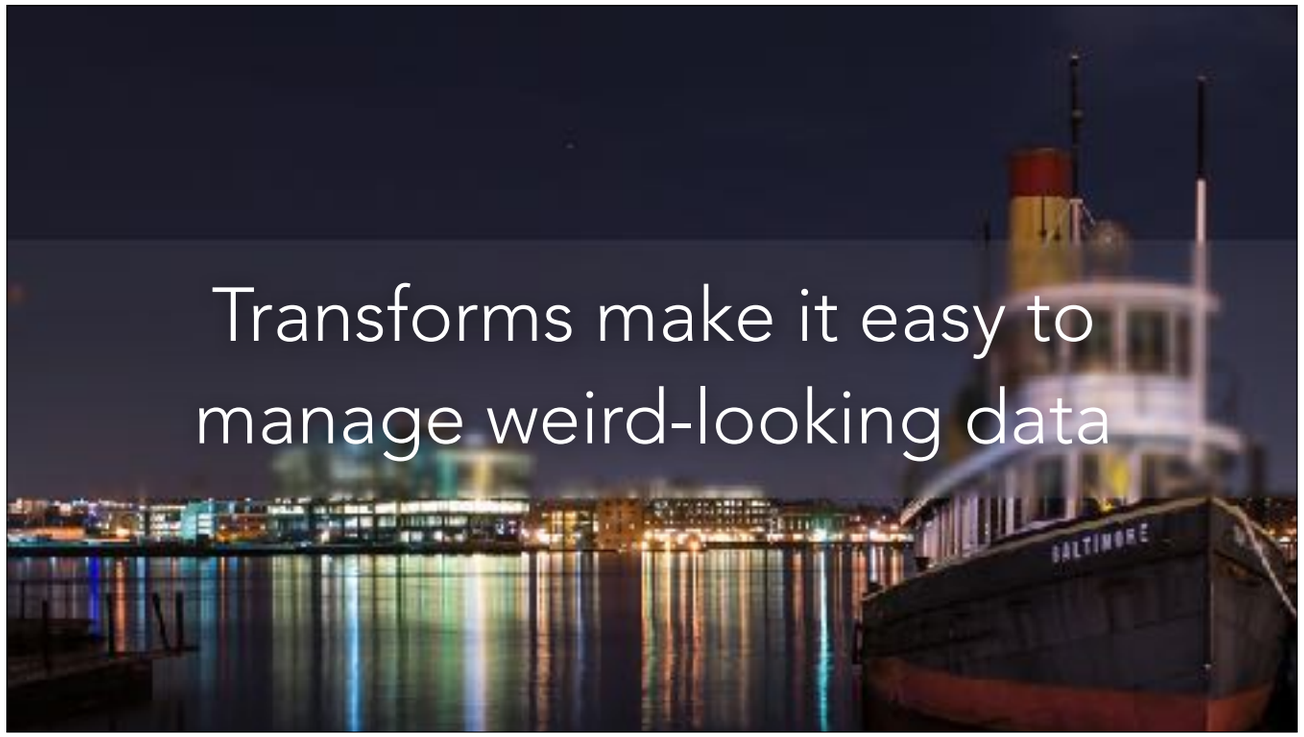
- An Entity relating to its own kind, like menu items.
- Caveat with this - Won't know about stuff you haven't brought over from your API.
    - Got away with this cause we load all our menu items from the API when we boot our Ember app.
    - Otherwise it will need to make requests to the API to gather more information about relationships.

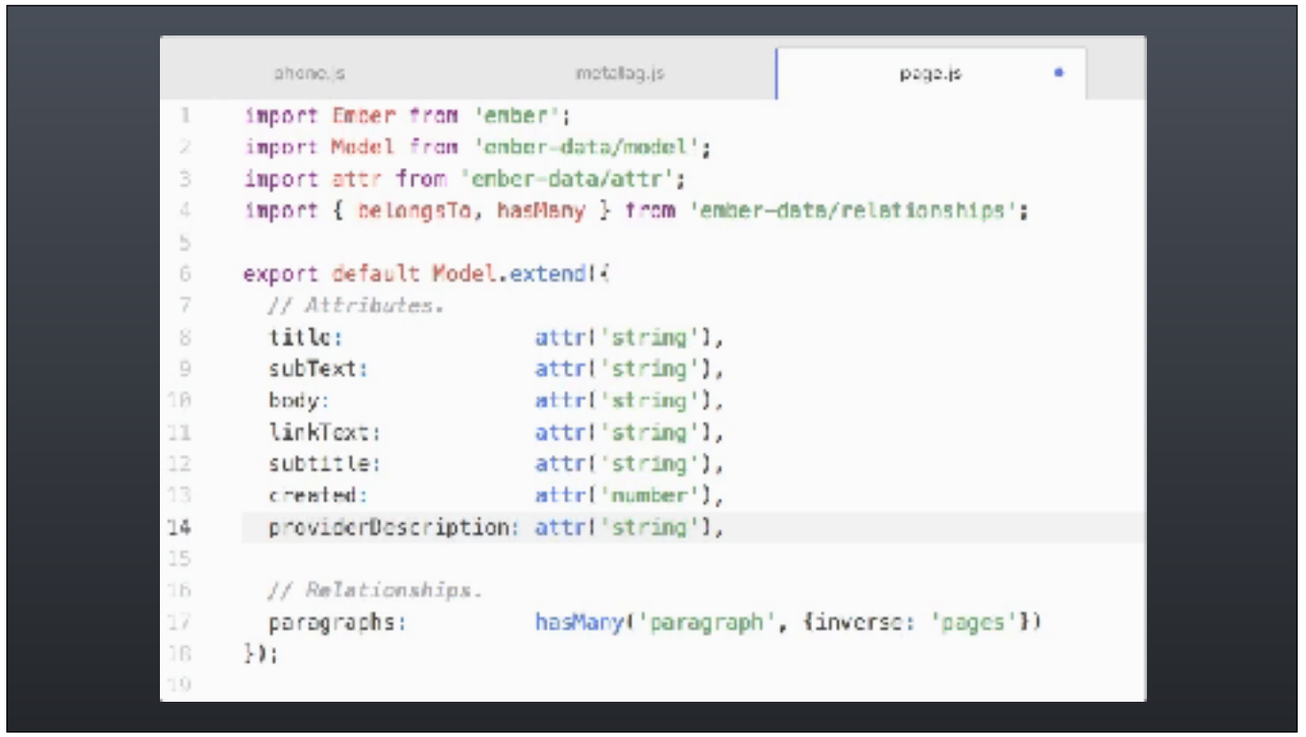
- So this is great. Ember has some robust relationship handling built in.
- Until now, we haven't really talked about static data in Models
    - Like titles, or nids, or created date. Those are static data.
- Ember can cast those to certain primitives like string, number out of the box. But what about the odd stuff? Every API has odd stuff.

“Won’t work cause the data’s weird”

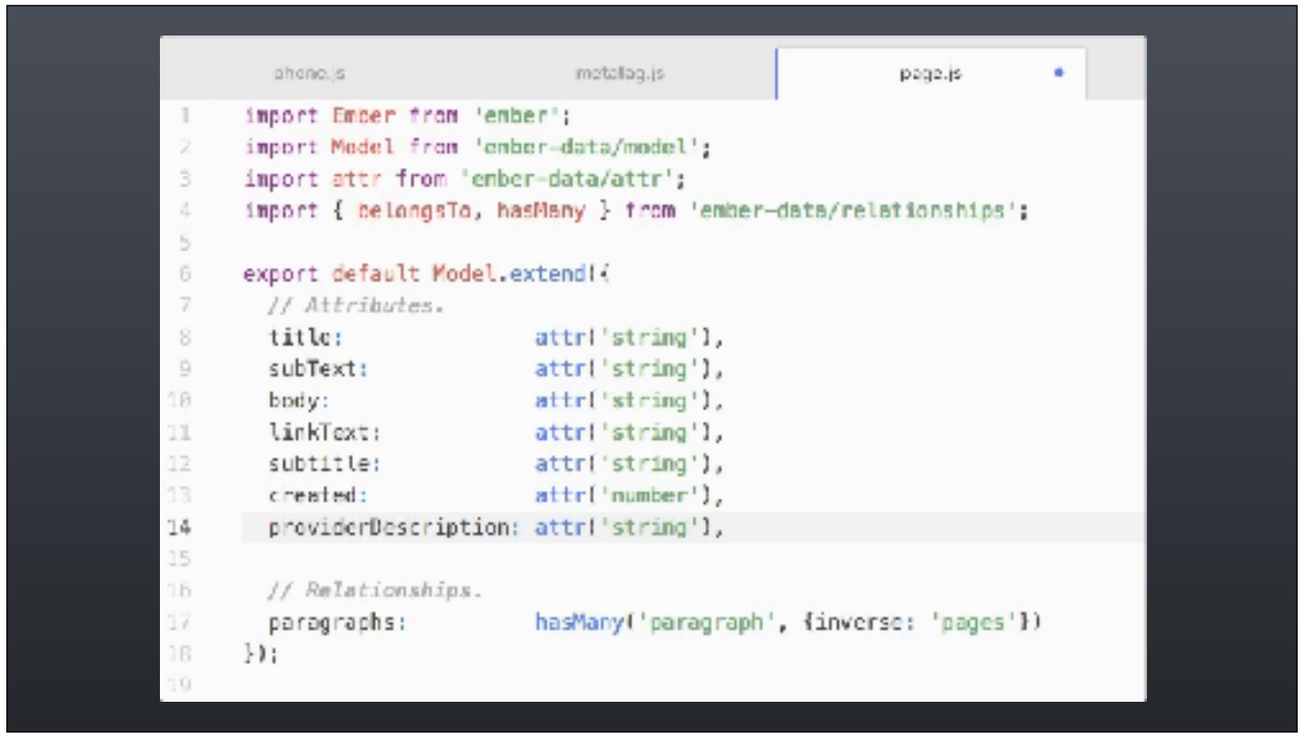
“Won’t work cause the data’s weird”

– All the haters. Also me.


Transforms make it easy to
manage weird-looking data
BALTIMORE



- Ember provides a way to manage the way a response looks (serializer)
- Also provides a way for you to write your own transformations of data at the field level.

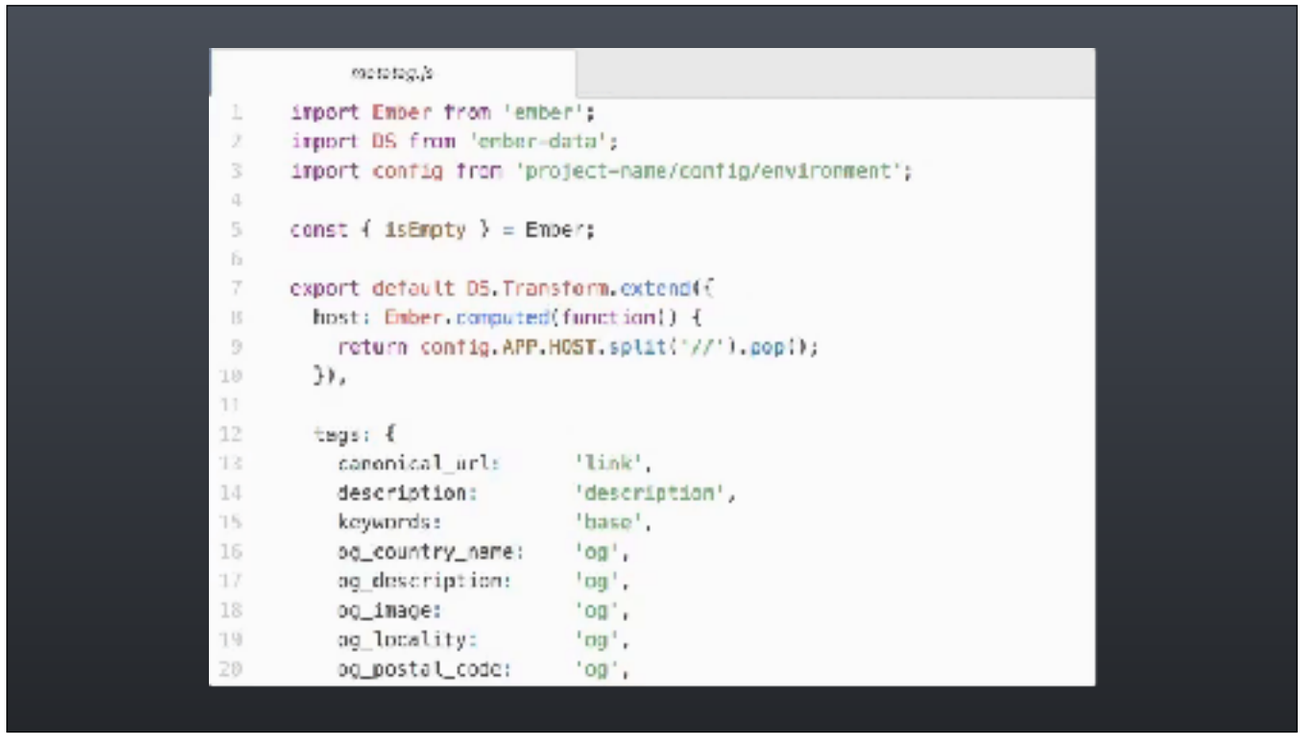


- Ember provides a way to manage the way a response looks (serializer)
- Also provides a way for you to write your own transformations of data at the field level.

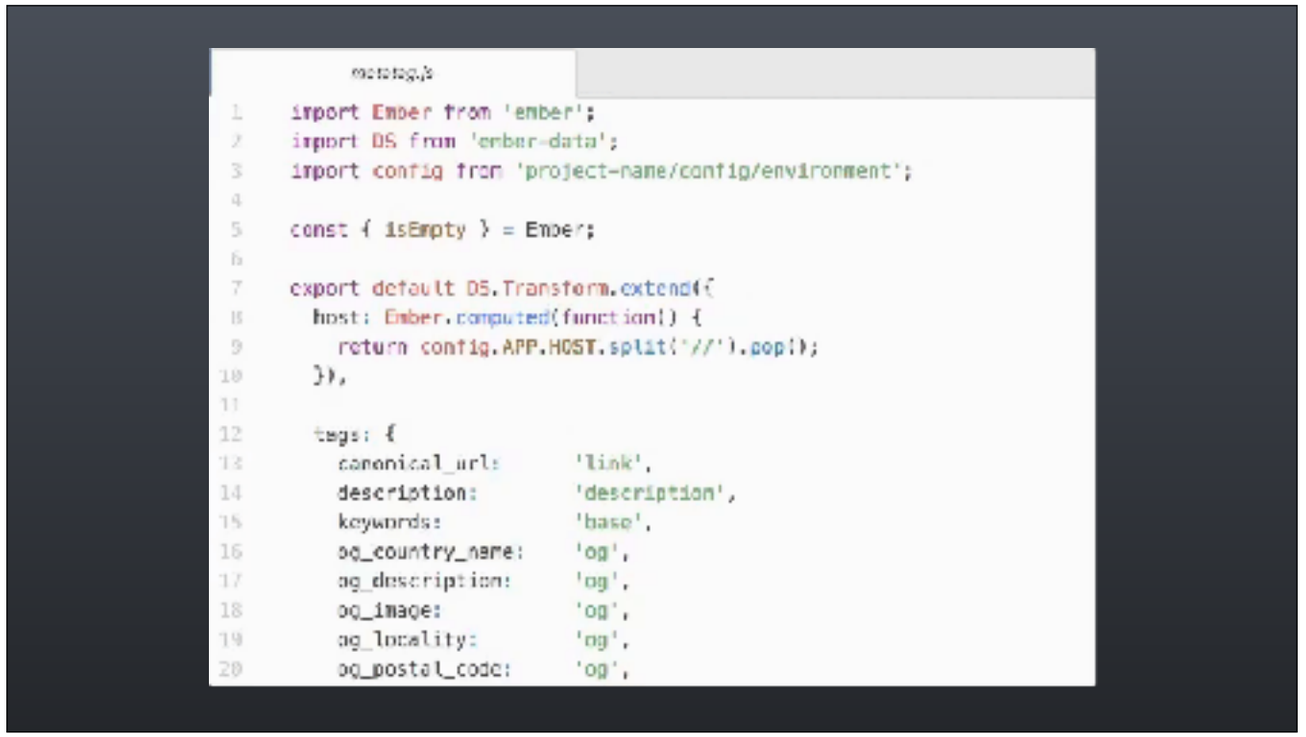
phone.js

```
import DS from 'ember-data';
import Ember from 'ember';

export default DS.Transform.extend({
  deserialize(serialized) {
    return Ember.isEmpty(serialized) ? null :
      serialized.replace(/(\d{3})(\d{3})(\d{4})/, '($1) $2-$3');
  },

  serialize(deserialized) {
    return Ember.isEmpty(deserialized) ? null :
      deserialized.replace(/[^0-9.]/g, '');
  }
});
```

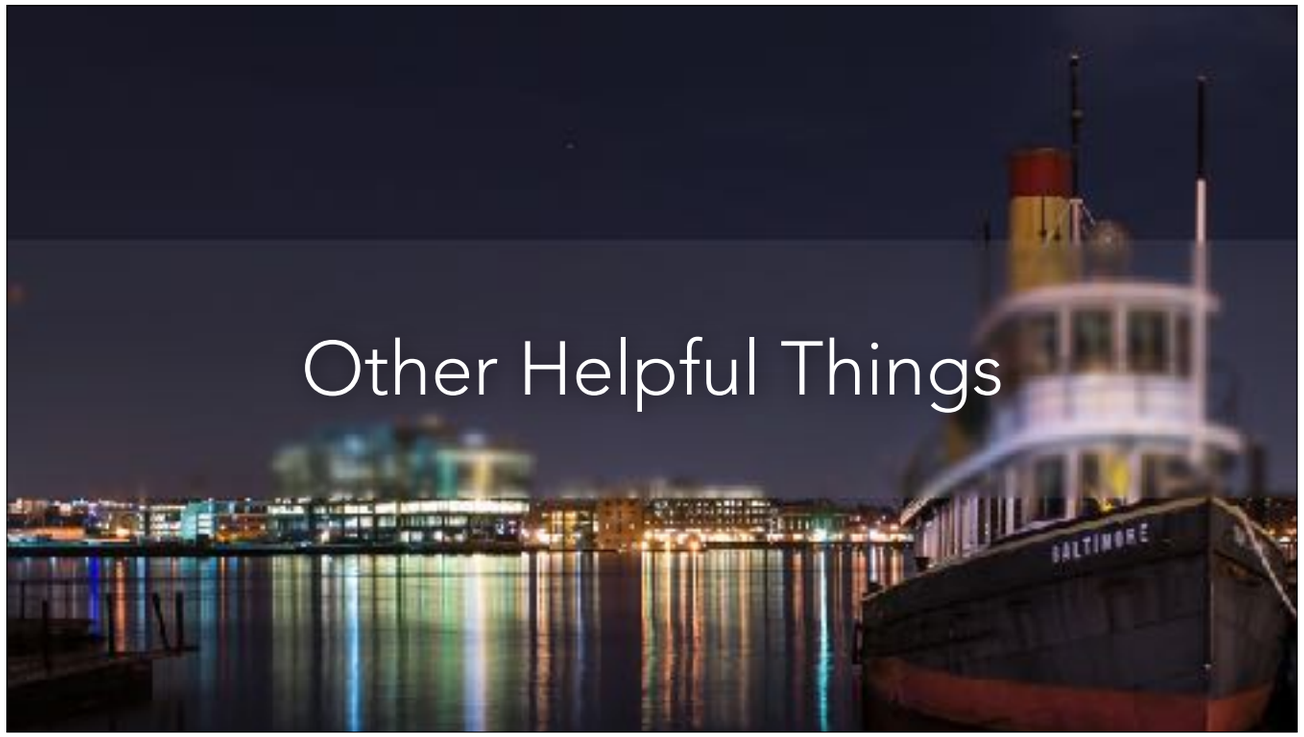
- Don’t have to necessarily rely on your content managers to enter data in a certain format.
- You can use transforms to homogenize data on the front-end rather than having to do it at your api level.



Transforms also allow you to change your data based on information only available once you’re on the client side.



Transforms also allow you to change your data based on information only available once you’re on the client side.



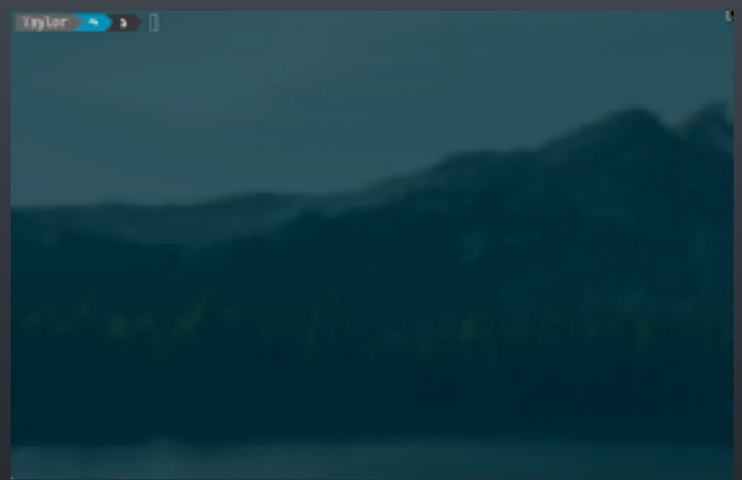
Other Helpful Things

Ember CLI

Scaffold code: Similar to Drupal Console. Transforms, models, routes, adapters, serializers

Ember CLI

- npm install -g ember-cli
- ember new
- Generates scaffold code!
- Automated testing
- Abstracts package.json management
- Live reloading + Linting



Scaffold code: Similar to Drupal Console. Transforms, models, routes, adapters, serializers

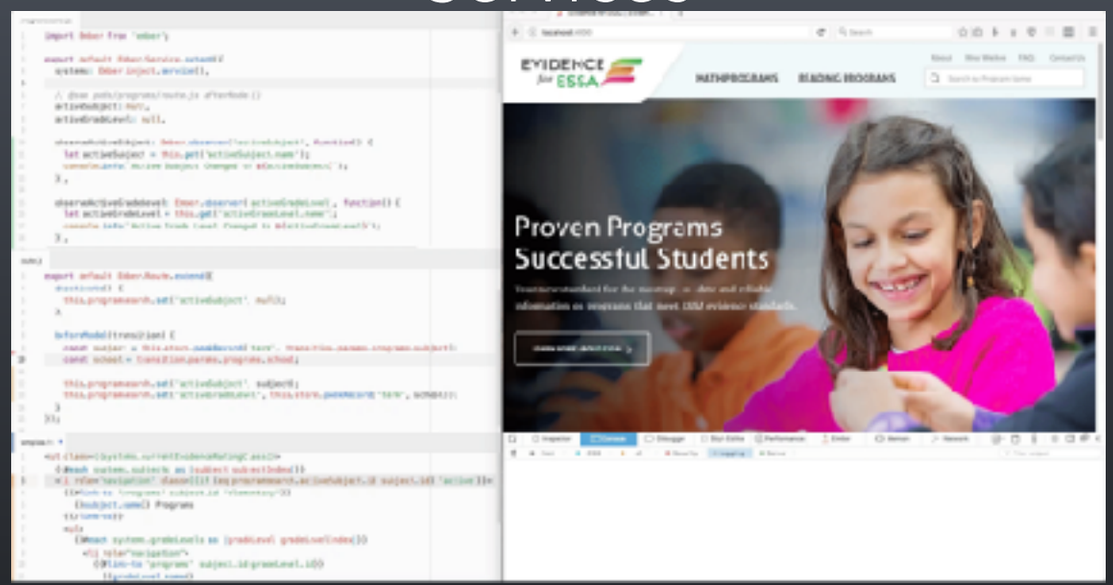
Services

Services are good for data that isn’t really worth putting in a Model.

We used it for navigation selections.
As transition between routes in our app, the service can be used to modify this global idea of active menu items.

Other uses: breadcrumbs, Handling third party api requests, websocket connections

Services



Services are good for data that isn’t really worth putting in a Model.

We used it for navigation selections.
As transition between routes in our app, the service can be used to modify this global idea of active menu items.

Other uses: breadcrumbs, Handling third party api requests, websocket connections

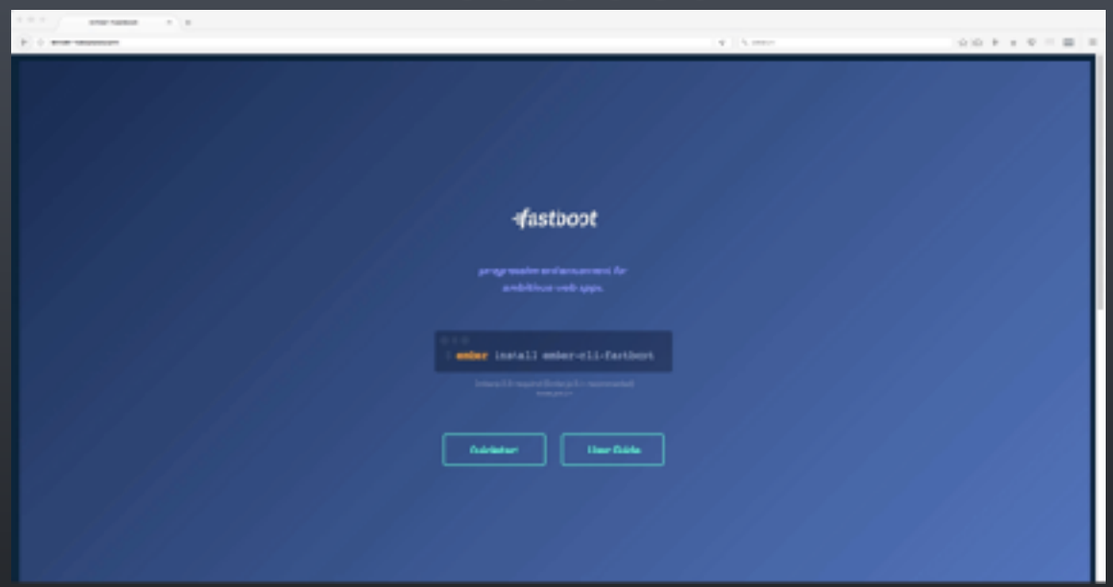
Fastboot

If you’re hesitating to jump into the JS MVC space because of the big question about SEO, Ember community has an answer.

This is the Ember community’s version of the idea of isomorphic javascript.

SPAs: Page source is blank, but gets filled in with markup on the client side…

Fastboot



If you’re hesitating to jump into the JS MVC space because of the big question about SEO, Ember community has an answer.

This is the Ember community’s version of the idea of isomorphic javascript.

SPAs: Page source is blank, but gets filled in with markup on the client side…

Other Helpful Things

- API Mocking with ember-cli-mirage
- A11y community is 👍
- Ember Engines - Divide and Conquer

API Mocking

- We didn’t know what our api was going to look like.
- Third party APIs could be mocked while they were being built
- Drive tests without a live API
- Concurrent FE and BE development

A11y

- More helpful and active
- Data Binding

Ember Engines

- Construct a User experience from several apps that all work together
- Lazy Loading, load only specific parts of app when needed.

Work In Progress

- Robust, performant Image Styles
- Redirects
- A synchronized path structure

Image styles

- Worth sending alongside our API data, or should we query them when needed?

Redirects - Use nginx maps or .htaccess right now.

Path structure: Drupal’s got pathauto

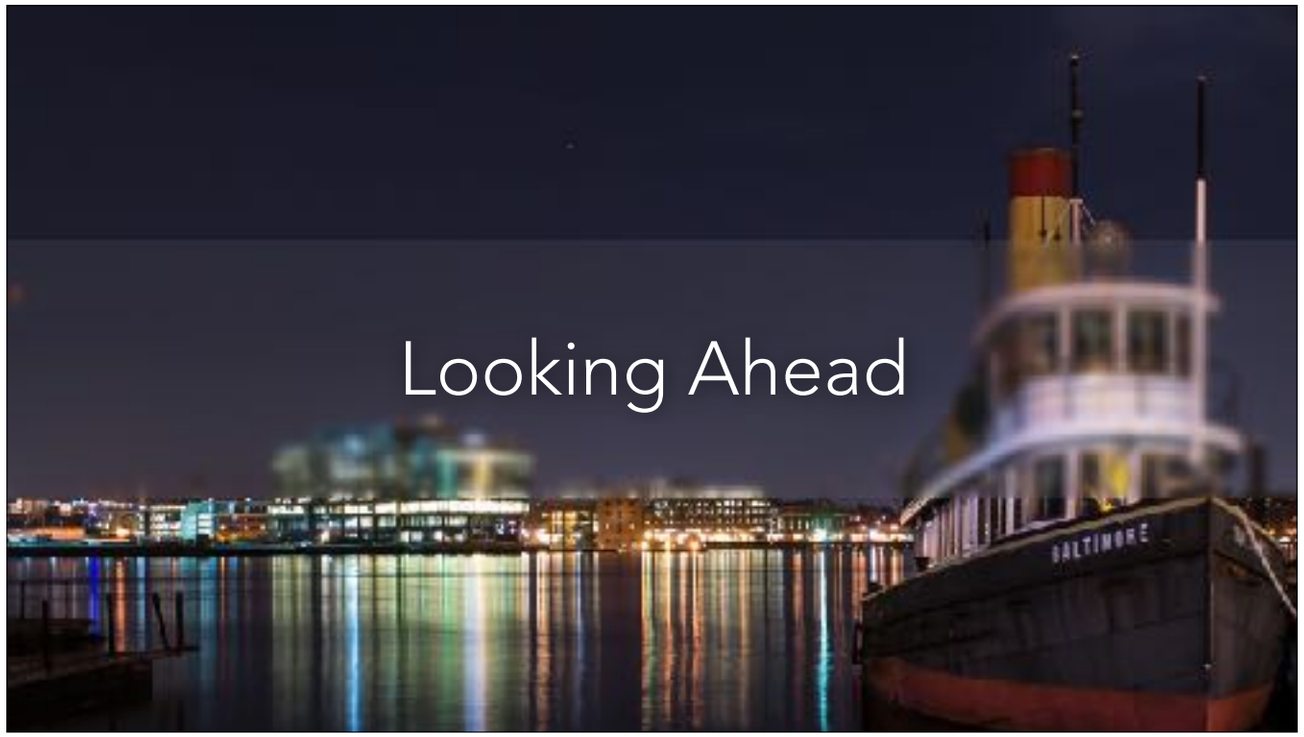
```
28    project_module.pages_id:
29      path: '/api/v0/pages/{id}'
30      defaults:
31        _controller: '\Drupal\ember_helper\Controller\EntityController::nodes'
32        type: page
33      requirements:
34        _permission: 'access content'
35        _method: 'GET|OPTIONS'
36      options:
37        _auth: ['token_bearer']
38        parameters:
39          id:
40            type: title_converter
41          alias_path: '/page/{id}'
```

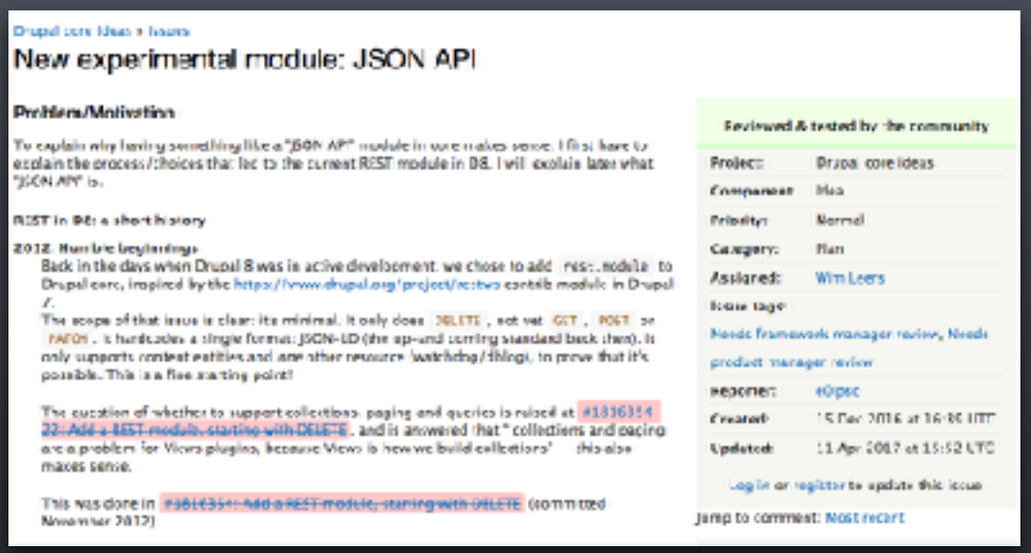
Our ID is actually the pathauto-if-ied version of our node title. And using a custom route option we added called “alias_path”, we’re able to inform our ParamConverter about the full pathauto url structure based on the node bundle we’re dealing with.

```php
<?php

namespace Drupal\ember_helper\ParamConverter;

use Drupal\Core\ParamConverter\ParamConverterInterface;
use Symfony\Component\Routing\Route;

class EmberHelperTitleConverter implements ParamConverterInterface {

  public function convert($value, $definition, $name, array $defaults) {
    // Compute path based on alias_path so we can handle wildcards and stuff.
    $path = $this->constructPath($value, $defaults);

    $connection = \Drupal::database();
    $q = "SELECT * FROM url_alias WHERE alias LIKE '%" . $path . "'";
    $result = $connection->query($q);
    if ($row = $result->fetchAssoc()) {
      $source = explode('/', $row['source']);
      $entity_type = empty($source[1]) ? '' : $source[1];
      $id = empty($source[2]) ? '' : $source[2];
```

Then we look up the path in the pathauto table and grab the nid there.

This would be simpler if the path for an entity could be a field attached to the entity itself.


Looking Ahead
BALTIMORE


Drupal core ideas » Issues

New experimental module: JSON API

Problem/Motivation

To explain why having something like a "JSON API" module in core makes sense, I first have to explain the process/choices that led to the current REST module in D8. I will explain later what "JSON API" is.

REST in D8: a short history

2012. Humble beginnings

Back in the days when Drupal 8 was in active development, we chose to add `rest.module` to Drupal core, inspired by the https://www.drupal.org/project/restws contrib module in Drupal 7.

The scope of that issue is clear: its minimal. It only does `DELETE`, not yet `GET`, `POST` or `PATCH`. It hardcodes a single format: JSON-LD (the up-and-coming standard back then). It only supports content entities and one other resource (watchdog/things), to prove that it's possible. This is a fine starting point!

The question of whether to support collections, paging and queries is raised at #1336314 22: Add a REST module, starting with DELETE, and is answered that " collections and paging are a problem for Views plugins, because Views is how we build collections" this also makes sense.

This was done in #1861354: Add a REST module, starting with DELETE (committed November 2012)

Reviewed & tested by the community

Project:	Drupal core ideas
Component:	Rest
Priority:	Normal
Category:	Plan
Assigned:	Wim Leers
Issue tags:	Needs framework manager review, Needs product manager review
Reporter:	e0ipso
Created:	15 Dec 2016 at 16:46 UTC
Updated:	11 Apr 2017 at 15:52 UTC

Log in or register to update this issue

Jump to comment: Most recent

https://www.drupal.org/node/2836165

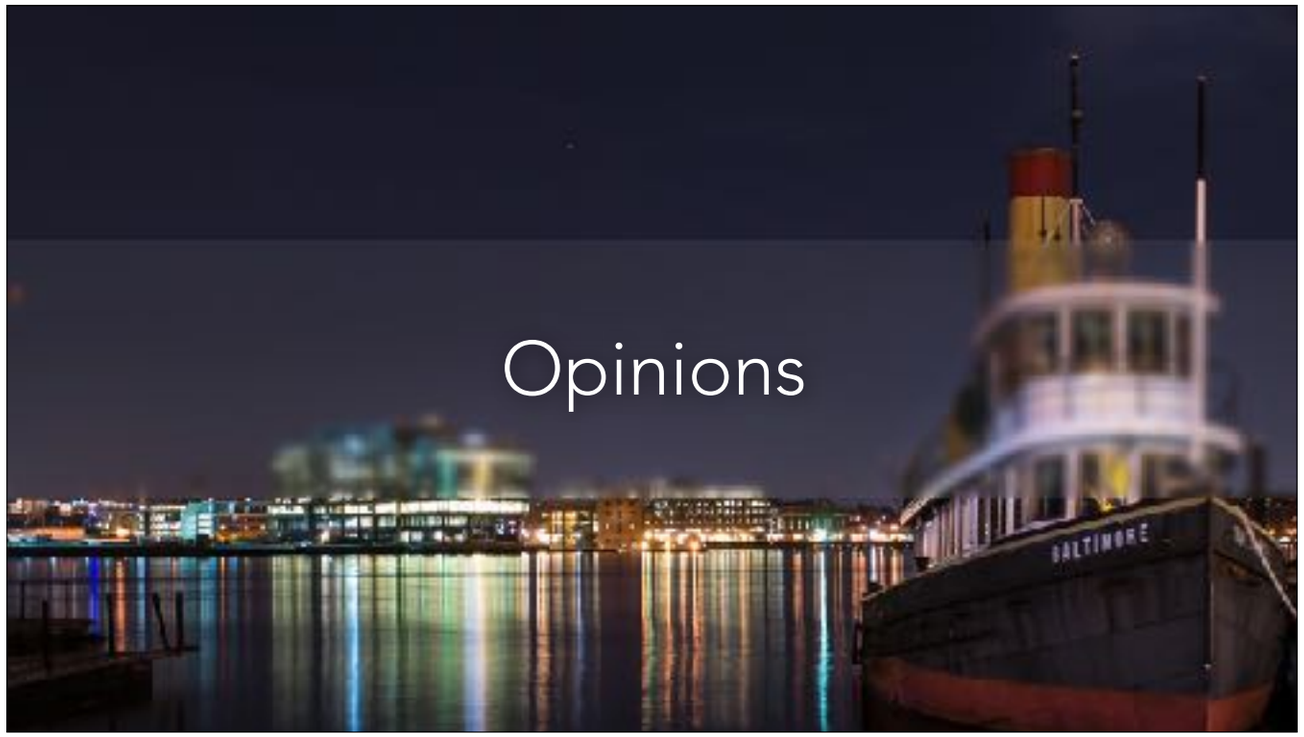
Ember data support for Drupal 8 CMS
(via JSON API module)
https://github.com/boztek/ember-data-drupal

Cardstack Project
http://cardstack.io/

Questions?

taylor@interactivestrategies.com

Feel free to hit me up via email!



- JS Framework space filled with opinions.
- In our case, opinions actually helped.
- We needed something that made enough opinions for us so we can ship something, but not so many that we were fighting it the whole way.
- Release Schedules
- Community-driven