May 25, 2018

Mike North

Ember Fundamentals

© Mike.Works, Inc. 2017. All rights reserved

```
npm install -g ember-cli
```

```
git clone https://github.com/mike-works/ember-fundamentals
```

https://docs.mike.works/ember-fundamentals-slides

Slack: #train-ember

VS Code Live: http://go/mnorth-live

May 25, 2018

Mike North

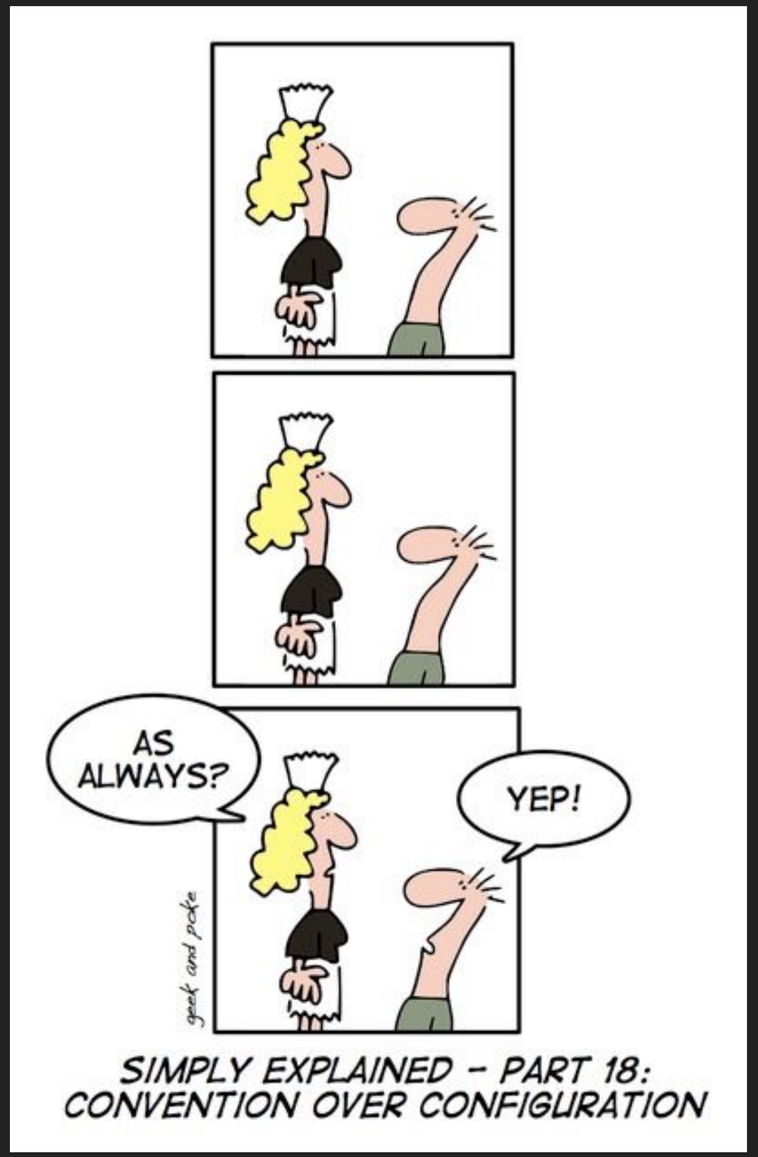
MIKE WORKS

Ember Fundamentals

© Mike.Works, Inc. 2017. All rights reserved

Philosophy of Ember

- Designed for “ambitious apps”
- Focused on productivity & ergonomics
- Aligned with web standards
- A complete, wholistic solution
- Constantly growing & improving



© Mike.Works, Inc. 2017. All rights reserved

Ecosystem Libraries

Framework	Ember.js
Data Persistence	Ember Data
Build Toolchain	Ember-cli
Animations	Liquid Fire
Virtualized List	List View

© Mike.Works, Inc. 2017. All rights reserved

Ecosystem Microlibraries

Prioritized, Batched Queue	backburner.js
URL handling	route-recognizer.js
SPA routing	router.js
A+ Promises	rsvp.js
Template Engine	handlebars / HTMLbars
Asset Pipeline	broccoli.js

© Mike.Works, Inc. 2017. All rights reserved

JavaScript Objects

© Mike.Works, Inc. 2017. All rights reserved

{ }

© Mike.Works, Inc. 2017. All rights reserved

```
{
    name: 'Mike',
    age: 34
}
```

© Mike.Works, Inc. 2017. All rights reserved

```
{
  name: 'Mike',
  age: 34,
  toString: function() {
    return `${this.name} - ${this.age}`;
  }
}
```

© Mike.Works, Inc. 2017. All rights reserved

```
{
  name: 'Mike',
  age: 34,
  toString() {
    return `${this.name} - ${this.age}`;
  }
}
```

© Mike.Works, Inc. 2017. All rights reserved

```javascript
function Person() {}

Person.prototype = {
  toString() {
    return `${this.name} - ${this.age}`;
  }
}

var mike = new Person();
```

© Mike.Works, Inc. 2017. All rights reserved

Properties can exist on prototypes or instances

PROTOTYPE

INSTANCE

© Mike.Works, Inc. 2017. All rights reserved

Properties can exist on prototypes or instances

PROTOTYPE

```
Person.prototype = {
  name: 'Mike'
};
```

INSTANCE

```
let mike = new Person();
mike.name = 'Mike';
```

© Mike.Works, Inc. 2017. All rights reserved

What happens when we access a property?

```
let mike = new Person();

 ...

mike.name = ?
```

INSTANCE

PROTOTYPE

© Mike.Works, Inc. 2017. All rights reserved

What happens when we access a property?

Do you <u>own</u> a property called "name"

INSTANCE

```
mike.name = ?
```

PROTOTYPE

© Mike.Works, Inc. 2017. All rights reserved

What happens when we access a property?

```
let mike = new Person();

...
```

Do you <u>own</u> a property called "name"

INSTANCE

PROTOTYPE

© Mike.Works, Inc. 2017. All rights reserved

What happens when we access a property?

```
function Person() {}
Person.prototype = {
  name: 'Default Name'
};
var mike = new Person();

mike.hasOwnProperty('name');
```

FALSE

© Mike.Works, Inc. 2017. All rights reserved

What happens when we access a property?

```
function Person() {}
Person.prototype = {
  name: 'Default Name'
};
var mike = new Person();
mike.name = 'Mike North';

mike.hasOwnProperty('name');
```

TRUE

© Mike.Works, Inc. 2017. All rights reserved

Anatomy of an Ember App

© Mike.Works, Inc. 2017. All rights reserved

Where do the files go?

Folder	Purpose
/app	Application source code
/vendor	Last resort for assets
/public	Pass-through in build pipeline
/tests	Tests
/config	App configuration
/lib	In-repo adding and engines

© Mike.Works, Inc. 2017. All rights reserved

/app/templates	Handlebars templates
/app/templates/components	Handlebars templates for
/app/routes	Routes
/app/components	Components
/app/helpers	Handlebars helpers
/app/models	Models
/app/styles	CSS, SASS, LESS
/app/initializers	Initializers - for customizing app
/app/services	Singletons
/app/router.js	Router
/app/index.html	Start-up HTML
/app/app.js	Ember Application object

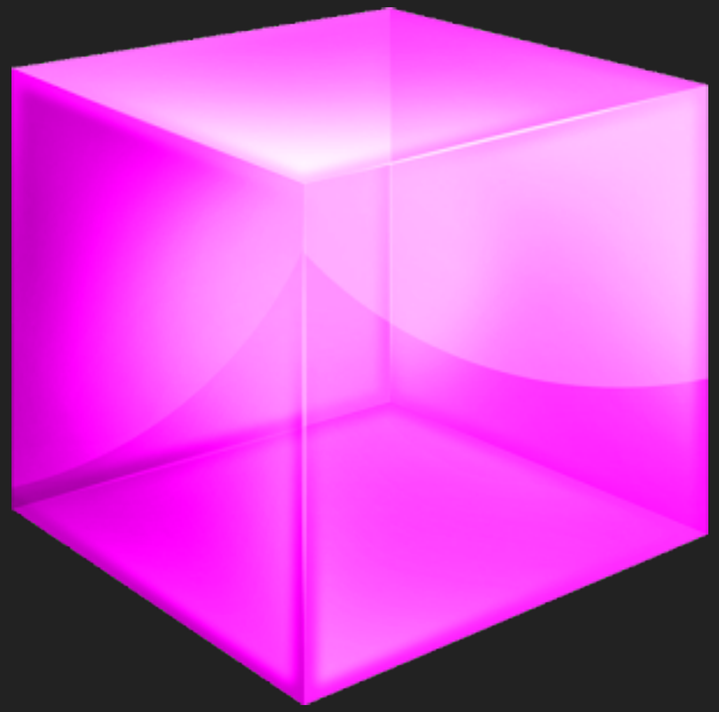
© Mike.Works, Inc. 2017. All rights reserved

Ember Objects

© Mike.Works, Inc. 2017. All rights reserved

Ember.(Core)Object

- Key-Value Observable API
  - ⚠️ Observers ⚠️
  - Concatenated & Merged properties
- Inheritance
- Some parallels to classical and prototypical inheritance
- Basic Lifecycle



© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API

```
let me = Ember.Object.create({name : 'Mike'});

me.get('name'); // 'Mike'
me.set('name', 'Michael');
me.get('name'); // 'Michael'
```

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Observers

```
let me = Ember.Object.create({name : 'Mike'});

me.addObserver('name', () => {
  console.log('Name changed!');
});

me.set('name', 'Michael');
```

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Observers

```
let me = Ember.Object.create({name : 'Mike'});

me.addObserver('name', () => {
  console.log('Name changed!');
});

me.notifyPropertyChange('name');
```

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Observers

```
let me = Ember.Object.create({name : 'Mike', age: 34});

me.addObserver('name', () => { console.log('name'); });
me.addObserver('age', () => { console.log('age'); });

me.set('name', 'Michael');
console.log('middle');
me.set('age', 35);
```

NAME

AGE

MIDDLE

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Observers

```
let me = Ember.Object.create({name : 'Mike', age: 34});

me.addObserver('name', () => { console.log('name'); });
me.addObserver('age', () => { console.log('age'); });

me.beginPropertyChanges();
me.set('name', 'Michael');
console.log('middle');
me.set('age', 35);
me.endPropertyChanges();
```

NAME

AGE

START COLLECTING EVENTS

MIDDLE

FLUSH COLLECTED EVENTS

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Arrays

```
let me = Ember.Object.create({
  names: ['Mike']
});

me.addObserver('names.[]', function() {
  console.log('names changed');
});
```

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Arrays

```
let me = Ember.Object.create({
  names: ['Mike']
});

me.addObserver('names.[]', function() {
  console.log('names changed');
});

me.get('names').push('Michael');
```

OBSERVER WON'T FIRE

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Arrays

```
let me = Ember.Object.create({
  names: ['Mike']
});

me.addObserver('names.[]', function() {
  console.log('names changed');
});

me.get('names').addObject('Michael');
```

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Arrays

```
let me = Ember.Object.create({
  names: ['Mike']
});

me.addObserver('names', function() {
  console.log('names changed');
});

me.get('names').addObject('Michael');
```

OBSERVER WON'T FIRE

© Mike.Works, Inc. 2017. All rights reserved

Key-Value Observable API: Observers

⚠️ Use Observers very rarely and EXTREMELY carefully ⚠️

© Mike.Works, Inc. 2017. All rights reserved

```
Ember.Object.extend(classDefinition)
```

© Mike.Works, Inc. 2017. All rights reserved

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() {
    return 'hello world!';
  }
});
```

© Mike.Works, Inc. 2017. All rights reserved

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() {
    return 'hello world!';
  }
});

var mike = Developer.create({ // Create an instance
  name: 'Mike'
});
```

© Mike.Works, Inc. 2017. All rights reserved

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() {
    return 'hello world!';
  }
});

var mike = Developer.create({ // Create an instance
  name: 'Mike'
});

mike.writeCode(); // 'hello world!'
mike.get('name'); // 'Mike'
```

© Mike.Works, Inc. 2017. All rights reserved

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() {
    return 'hello world!';
  }
});

var mike = Developer.create({ // Create an instance
  name: 'Mike'
});

mike.writeCode(); // 'hello world!'
mike.get('name'); // 'Mike'
mike.hasOwnProperty('name'); // true
```

© Mike.Works, Inc. 2017. All rights reserved

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() {
    return 'hello world!';
  }
});

var mike = Developer.create({ // Create an instance
  name: 'Mike'
});

mike.writeCode(); // 'hello world!'
mike.get('name'); // 'Mike'
mike.hasOwnProperty('name'); // true
mike.hasOwnProperty('writeCode'); // false
```

Owned by the prototype, shared across all instances

© Mike.Works, Inc. 2017. All rights reserved

Modifying instances & prototypes

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() { return 'hello world!'; }
});
```

© Mike.Works, Inc. 2017. All rights reserved

Modifying instances & prototypes

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() { return 'hello world!'; }
});

Developer.reopen({
  printCode() {
    console.log(this.writeCode().toUpperCase());
  }
});
```

© Mike.Works, Inc. 2017. All rights reserved

Modifying instances & prototypes

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() { return 'hello world!'; }
});

Developer.reopen({
  printCode() {
    console.log(this.writeCode().toUpperCase());
  }
});

var mike = Developer.create({name: 'Mike'});
```

© Mike.Works, Inc. 2017. All rights reserved

Modifying instances & prototypes

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() { return 'hello world!'; }
});

Developer.reopen({
  printCode() {
    console.log(this.writeCode().toUpperCase());
  }
});

var mike = Developer.create({name: 'Mike'});
mike.printCode(); // 'HELLO WORLD!'
```

© Mike.Works, Inc. 2017. All rights reserved

Modifying instances & prototypes

```
var Developer = Ember.Object.extend({ // Create the subclass
  writeCode() { return 'hello world!'; }
});
```

© Mike.Works, Inc. 2017. All rights reserved

Modifying instances & prototypes

```
Developer.reopenClass({
  _jobTitles: ['engineer', 'sr. engineer'],
  isValidJobTitle(title) {
    return this._jobTitles.indexOf(title.toLowerCase()) ≥ 0;
  }
});

Developer.isValidJobTitle('Engineer'); // true
```

© Mike.Works, Inc. 2017. All rights reserved

Types of Properties

© Mike.Works, Inc. 2017. All rights reserved

Standard Properties

```
var Developer = Ember.Object.extend({
  jobs: ['developer']
});

var mike = Developer.create({
  name: 'Mike',
  jobs: ['trainer']
});

mike.get('jobs');   // ['trainer']
```

© Mike.Works, Inc. 2017. All rights reserved

Concatenated Properties

```
var Developer = Ember.Object.extend({
  jobs: ['developer'],
  concatenatedProperties: ['jobs']
});

var mike = Developer.create({
  name: 'Mike',
  jobs: ['trainer']
});
```

© Mike.Works, Inc. 2017. All rights reserved

Concatenated Properties

```
var Developer = Ember.Object.extend({
  jobs: ['developer'],
  concatenatedProperties: ['jobs']
});

var mike = Developer.create({
  name: 'Mike',
  jobs: ['trainer']
});

mike.get('jobs');   // ['developer', 'trainer']
```

© Mike.Works, Inc. 2017. All rights reserved

Concatenated Properties

```
var Developer = Ember.Object.extend({
  jobs: ['developer'],
  concatenatedProperties: ['jobs']
});

var mike = Developer.create({
  name: 'Mike',
  jobs: 'trainer'
});

mike.get('jobs');   // ['developer', 'trainer']
```

© Mike.Works, Inc. 2017. All rights reserved

Merged Properties

```
var Developer = Ember.Object.extend({
  programmingExperience: { js: 8, html: 20 },
  mergedProperties: ['programmingExperience']
});

var mike = Developer.create({
  name: 'Mike',
  programmingExperience: { elixir: 3 }
});

mike.get('programmingExperience'); // {js: 8, html: 20, elixir: 3}
```

© Mike.Works, Inc. 2017. All rights reserved

Basic Lifecycle

© Mike.Works, Inc. 2017. All rights reserved

Ember.Object lifecycle

```
var Person = Ember.Object.extend({
  init() {
    this._super( ... arguments);
    console.log("I'm being created");
  },
  willDestroy() {
    this._super( ... arguments);
    console.log('I am about to be destroyed');
  }
});
```

© Mike.Works, Inc. 2017. All rights reserved

Ember.Object lifecycle

```
var Person = Ember.Object.extend({
  init() {
    this._super( ... arguments);
    console.log("I'm being created");
  },
  setupThings: Ember.on('init', function() {
    console.log('I was just created')
  })
});
```

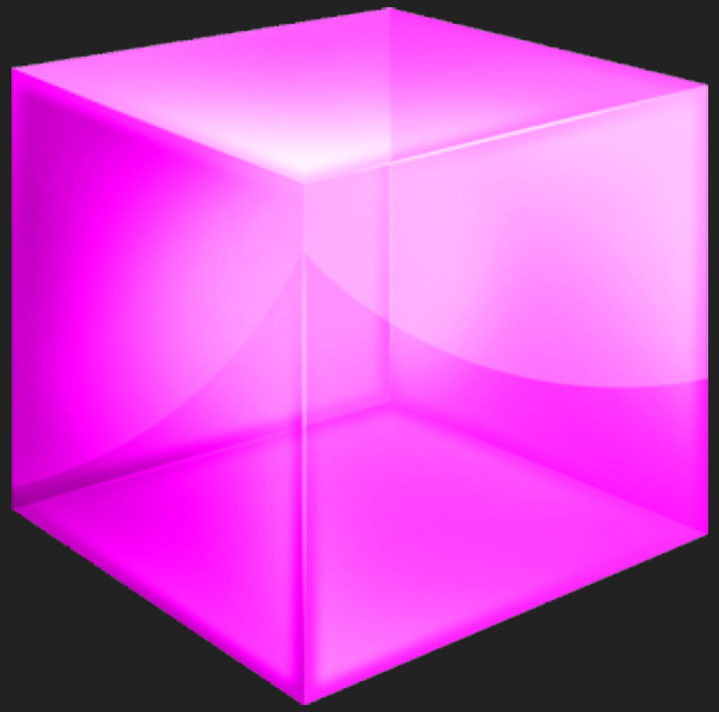
NO EVENTS FIRED

EVENTS FIRED

© Mike.Works, Inc. 2017. All rights reserved

Ember.(Core)Object

- Key-Value Observable API
  - ⚠️ Observers ⚠️
  - Concatenated & Merged properties
- Inheritance
- Some parallels to classical and prototypical inheritance
- Basic Lifecycle



© Mike.Works, Inc. 2017. All rights reserved

git clone https://github.com/mike-works/ember-fundamentals

Data Modeling

1

```
ember g util course
```

```
const Course = Ember.Object.extend({

});
export default Course;
```

- instance.title, defaults to “no title”
- instance.description, defaults to empty string
- instance.languages[]
- instance.tags[]
- Course.languageName(lang)

© Mike.Works, Inc. 2017. All rights reserved

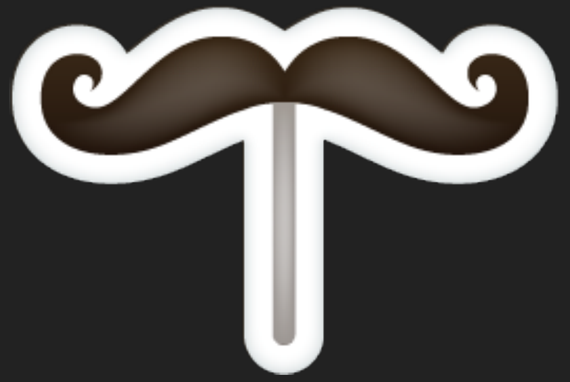
Templates

© Mike.Works, Inc. 2017. All rights reserved

Templates

- Templates are compiled at build time
- Turn into functions that are passed data, and emit _____
  - Ember 1.9 - Strings
  - Ember 1.10+ - DOM Fragments

handlebars



© Mike.Works, Inc. 2017. All rights reserved

```
Hello, <strong>{{firstName}} {{lastName}}</strong>!
```

© Mike.Works, Inc. 2017. All rights reserved

```
define('examples/templates/index', ['exports'], function (exports) {

  'use strict';

  exports['default'] = Ember.HTMLBars.template((function() {
    return {
      ...
      buildFragment: function buildFragment(dom) {
        var el0 = dom.createDocumentFragment();
        var el1 = dom.createTextNode("Hello, ");
        dom.appendChild(el0, el1);
        var el1 = dom.createElement("strong");
        var el2 = dom.createComment("");
        dom.appendChild(el1, el2);
        var el2 = dom.createTextNode(" ");
        dom.appendChild(el1, el2);
        var el2 = dom.createComment("");
        dom.appendChild(el1, el2);
        dom.appendChild(el0, el1);
        var el1 = dom.createTextNode("!");
        dom.appendChild(el0, el1);
        return el0;
      },
      buildRenderNodes: function buildRenderNodes(dom, fragment, contextualElement) {
        var element0 = dom.childAt(fragment, [1]);
        var morphs = new Array(2);
        morphs[0] = dom.createMorphAt(element0,0,0);
        morphs[1] = dom.createMorphAt(element0,2,2);
```

Lisp

- Everything is a list, with items separated by spaces
- The first thing in the list tells you what to do with the rest

```
(add 2 3) -> 5
(subtract 7 (add 2 3)) -> -2
```

© Mike.Works, Inc. 2017. All rights reserved

Handlebars:

HTML + Lisp

© Mike.Works, Inc. 2017. All rights reserved

Conditionals

```
My pet goes {{if isDog "arf" "meow"}}
```

Block syntax begins
{{#abc}}
and ends
{{/abc}}

```
My pet goes
{{#if isDog}}
  arf
{{else}}
  meow
{{/if}}
```

```
My pet goes
{{#unless isDog}}
  meow
{{else}}
  arf
{{/unless}}
```

© Mike.Works, Inc. 2017. All rights reserved

Simple helpers are just PURE FUNCTIONS

```
<h1>Hello {{user.name}}</h1>

<p>
I'm a {{if animal.isDog "Dog" "Cat"}}
</p>
```

```
function if(condition,    /* (truthy?) */
            valueIfTrue, /* value */
            valueIfFalse /* value */) {
  if (condition) return valueIfTrue;
  else return valueIfFalse;
}
```

© Mike.Works, Inc. 2017. All rights reserved

Subexpressions

```
My pet goes
{{#if (equal petType "dog")}}
  arf
{{else}}
  meow
{{/if}}
```

💡 Changes in subexpressions cause outer expressions to re-evaluate

© Mike.Works, Inc. 2017. All rights reserved

Subexpressions

```
My pet goes
{{#if (equal petType "dog")}}
  arf
{{else}}
  meow
{{/if}}
```

DATA

© Mike.Works, Inc. 2017. All rights reserved

Subexpressions

```
My pet goes
{{#if (equal petType "dog")}}
  arf
{{else}}
  meow
{{/if}}
```

HELPER

© Mike.Works, Inc. 2017. All rights reserved

Iteration

```
<ul>
{{#each myList as |myListItem|}}
  <li>{{myListItem}}</li>
{{/each}}
</ul>
```

© Mike.Works, Inc. 2017. All rights reserved

Iteration

```
<ul>
{{#each myList as |myListItem|}}
  <li>{{myListItem}}</li>
{{else}}
  <li>Sorry, list is empty!</li>
{{/each}}
</ul>
```

© Mike.Works, Inc. 2017. All rights reserved

Iteration

```
<ul>
{{#each myList as |myListItem|}}
  <li>{{myListItem}}</li>
{{else}}
  <li>Sorry, list is empty!</li>
{{/each}}
</ul>
```

```
function each(myList) {
  if (myList.length) {
    return myList.map((myListItem) =>{
      return `<li>${myListItem}</li>`
    })
  } else {
    return "<li>Sorry, list is empty!</li>";
  }
}
```

© Mike.Works, Inc. 2017. All rights reserved

Passing data to a helper

```
{{placeholder-url
    "thing1"      // params[0]
    "thing2"      // params[1]
    align="left"  // hash.align
    height=24}}   // hash.height
```

```
placeholderUrl(
  ["thing1", "thing2"],
  {align: "left", height: 24}
);
```

© Mike.Works, Inc. 2017. All rights reserved

Opt out of two-way binding

```
Hello,
<strong>
  {{unbound firstName}}
  {{unbound lastName}}
</strong>!
```

© Mike.Works, Inc. 2017. All rights reserved

Demo

MAKING A SIMPLE HANDLEBARS HELPER

© Mike.Works, Inc. 2017. All rights reserved

An image placeholder helper

2

```
ember g helper placeholder-url
```

My favorite placeholder image service uses this convention:

BASE_URL/WIDTH/HEIGHT

- http://placekitten.com/121/314

I want a handlebars helper that builds a URL for me when I use markup like:

```
<img src={{placeholder-url w=300 h=200}} />
```

make all “Exercise 2” tests pass

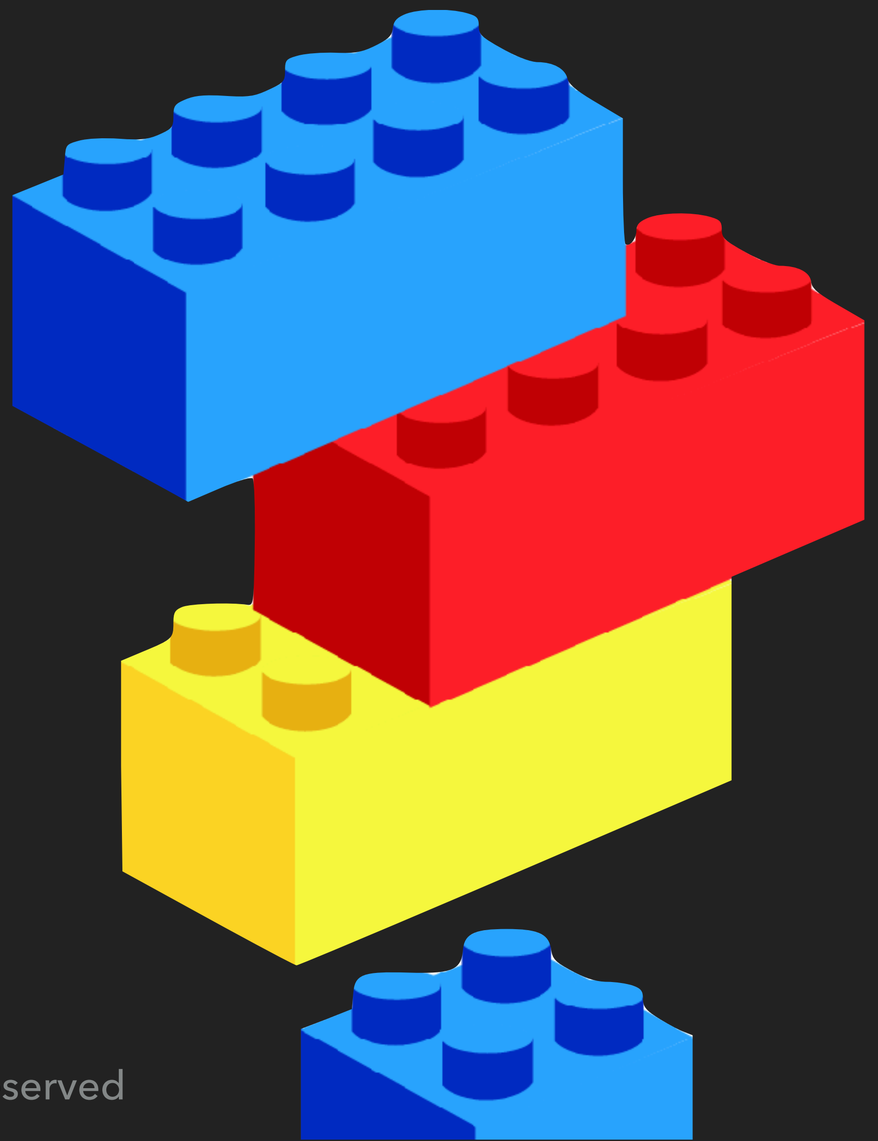
© Mike.Works, Inc. 2017. All rights reserved

Components

© Mike.Works, Inc. 2017. All rights reserved

Important Component-y things

- Well-orchestrated life-cycle
- Encapsulation
- Explicit contract with the outside world
- Reusability
- Composability



© Mike.Works, Inc. 2017. All rights reserved

Full Name

```
<input type="text" placeholder="Full Name">
▼#shadow-root (user-agent)
    <div id="inner-editor"></div>
    <div pseudo="-webkit-input-placeholder" id=
    "placeholder" style="display: block; text-
    overflow: clip;">Full Name</div>
</input>
```

© Mike.Works, Inc. 2017. All rights reserved

Where are we going with this?

```
<input
  placeholder="Password" />

<my-input
  minchars=6
  placeholder="Password" />
```

© Mike.Works, Inc. 2017. All rights reserved

Where are we going with this?

```
<input
  placeholder="Password" />

{{my-input
  minchars=6
  placeholder="Password" }}
```

© Mike.Works, Inc. 2017. All rights reserved

Where are we going with this?

	INLINE	"BLOCK"
HTML	`<img />`	`<div> … </div>`

HANDLEBARS	`{{my-thing}}`	`{{#my-thing}}`
		…
		`{{/my-thing}}`

© Mike.Works, Inc. 2017. All rights reserved

Where are we going with this?

```
<form>
…
</form>
```

```
{{#login-form}}
…
{{/login-form}}
```

© Mike.Works, Inc. 2017. All rights reserved

Where are we going with this?

```
{{#login-form}}
   {{my-input minchars=6 placeholder="Email"}}
   {{my-input minchars=8 placeholder="Password"}}

   {{#my-button}} Submit {{/my-button}}
{{/login-form}}
```

© Mike.Works, Inc. 2017. All rights reserved

Elements Network Sources Timeline Profiles Resources Audits Console

```
▶<div class="vI8oZc cS">…</div>
 <div tabindex="0"></div>
▼<div class="nH" style="width: 1411px;">
 ▼<div class="nH" style="position: relative;">
  ▶<div class="nH w-asV aiw">…</div>
  ▼<div class="nH">
   ▼<div class="no">
    ▶<div class="nH oy8Mbf nn aeN" s="width: 202px; height: 352px;">…</div>
    ▼<div class="nH nn" style="width px;">
     ▼<div class="nH">
      ▼<div class="nH">
       ▼<div class=" ar4 z">
        ▼<div id=":5" class
         ▼<div class="D E
          ▼<div class="
           ▼<div class
            ▼<div>
             ▼<div>
              ▶<div class="G-Ni J-J5-Ji">…</div>
              ▶<div class="G-Ni J-J5-Ji" style="display: none;">…</div>
              ▶<div class="G-Ni J-J5-Ji" style="display: none;">…</div>
              ▶<div class="G-Ni J-J5-Ji">…</div>
              ▶<div class="J-J5-Ji">…</div>
              ▶<div class="G-Ni J-J5-Ji">…</div>
```

Two pieces to a component

- Template module
  - The “view” part
- Ember component module
  - Template context
  - Lifecycle hooks
  - Action handlers

MY-COMPONENT.HBS

MY-COMPONENT.JS

© Mike.Works, Inc. 2017. All rights reserved

Ember Components - Basic Lifecycle Hooks

didInsertElement	after the component's element has been inserted into the DOM
willDestroyElement	before the component's element is removed from the DOM

Clean up before tear down

$(document).ready() for components

© Mike.Works, Inc. 2017. All rights reserved

Ember Components - Customizing the Boundary Element

```
Ember.Component.extend({
  tagName: 'span', // boundary element tag type
});
```

© Mike.Works, Inc. 2017. All rights reserved

Ember Components - Customizing the Boundary Element

```
Ember.Component.extend({
  tagName: 'span', // boundary element tag type

  classNames: ['my-component'], // boundary element classes
});
```

© Mike.Works, Inc. 2017. All rights reserved

Ember Components - Customizing the Boundary Element

```
Ember.Component.extend({
  tagName: 'span', // boundary element tag type

  classNames: ['my-component'], // boundary element classes

  attributeBindings: ['company:data-company'],
    // <div data-company="LinkedIn"> ... </div>

  company: 'LinkedIn'
});
```

© Mike.Works, Inc. 2017. All rights reserved

Demo

MAKING A SIMPLE COMPONENT

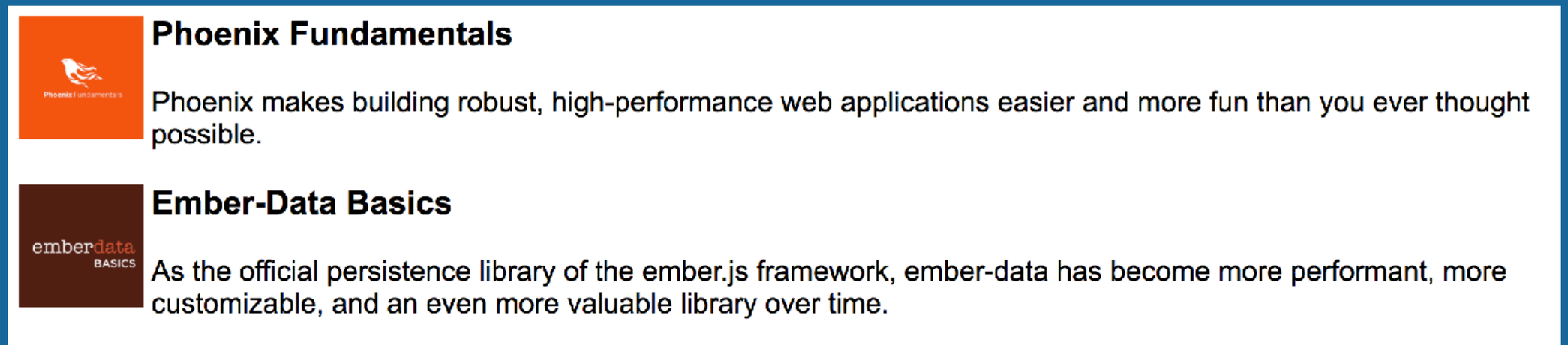
© Mike.Works, Inc. 2017. All rights reserved

Customizing Component Boundary

3

```
ember g component course-list-item
```

- There's some data in your application controller (you can iterate over it in your application.hbs file)
- Get all failing tests to pass



Https://emberlearn.herokuapp.com

Computed Properties

© Mike.Works, Inc. 2017. All rights reserved

Computed Properties

- Properties that are calculated, based on other properties
- Evaluated lazily
- Cached
- Still can get/set

© Mike.Works, Inc. 2017. All rights reserved

Computed Properties

Typical read-only syntax

```
Ember.Component.extend({
  fullName: Ember.computed('firstName', 'lastName', function() {
    return `${this.get('firstName')} ${this.get('lastName')}`;
  })
});
```

© Mike.Works, Inc. 2017. All rights reserved

Computed Properties

Alternate read-only syntax

```
Ember.Component.extend({
  fullName: Ember.computed('firstName', 'lastName', {
    get() {
      return `${this.get('firstName')} ${this.get('lastName')}`;
    }
  })
});
```

© Mike.Works, Inc. 2017. All rights reserved

Computed Properties

Alternate read-only syntax

```
Ember.Component.extend({
  fullName: Ember.computed('firstName', 'lastName', {
    get() {
      return `${this.get('firstName')} ${this.get('lastName')}`;
    }
  })
});
```

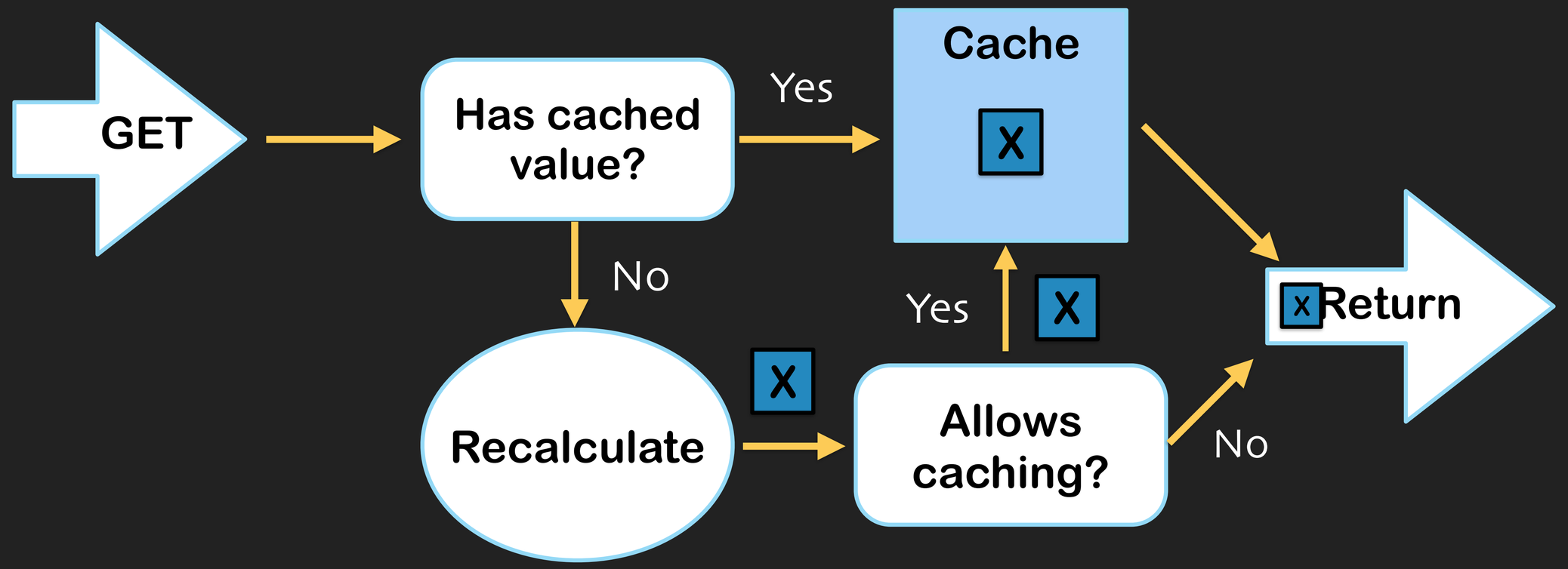
© Mike.Works, Inc. 2017. All rights reserved

Computed Properties

```
fullName: Ember.computed('firstName', 'lastName', {
  get() {
    return `${this.get('firstName')} ${this.get('lastName')}`;
  },
  set(key, newVal) {
    let nameParts = newVal.split(' ');
    this.set('firstName', nameParts[0]);
    this.set('lastName', nameParts[1]);
    return newVal; // Very important to return this
  }
})
```

© Mike.Works, Inc. 2017. All rights reserved

Computed Properties



© Mike.Works, Inc. 2017. All rights reserved

Demo

COMPUTED PROPERTIES
& COMPUTED PROPERTY MACROS

© Mike.Works, Inc. 2017. All rights reserved

First Sentence

4

- Build a computed property macro that can be used to get the first sentence of a string.
- The computed property macro must live in the utils folder
- Reasonable passing unit tests are required

© Mike.Works, Inc. 2017. All rights reserved

Two Inputs

5

Hello World! | HELLO WORLD!

```
<input type="text" class="lowercase-field" />
<input type="text" class="uppercase-field" />
```

- {{two-inputs}} component, contains two inputs
- .lowercase-field should be bound to str property
- .uppercase-field should be bound to a writable computed property
  - Typing letters in .uppercase-field should update .lowercase-field accordingly
  - .lowercase-field should end up being forced to all-lowercase

© Mike.Works, Inc. 2017. All rights reserved

Routing

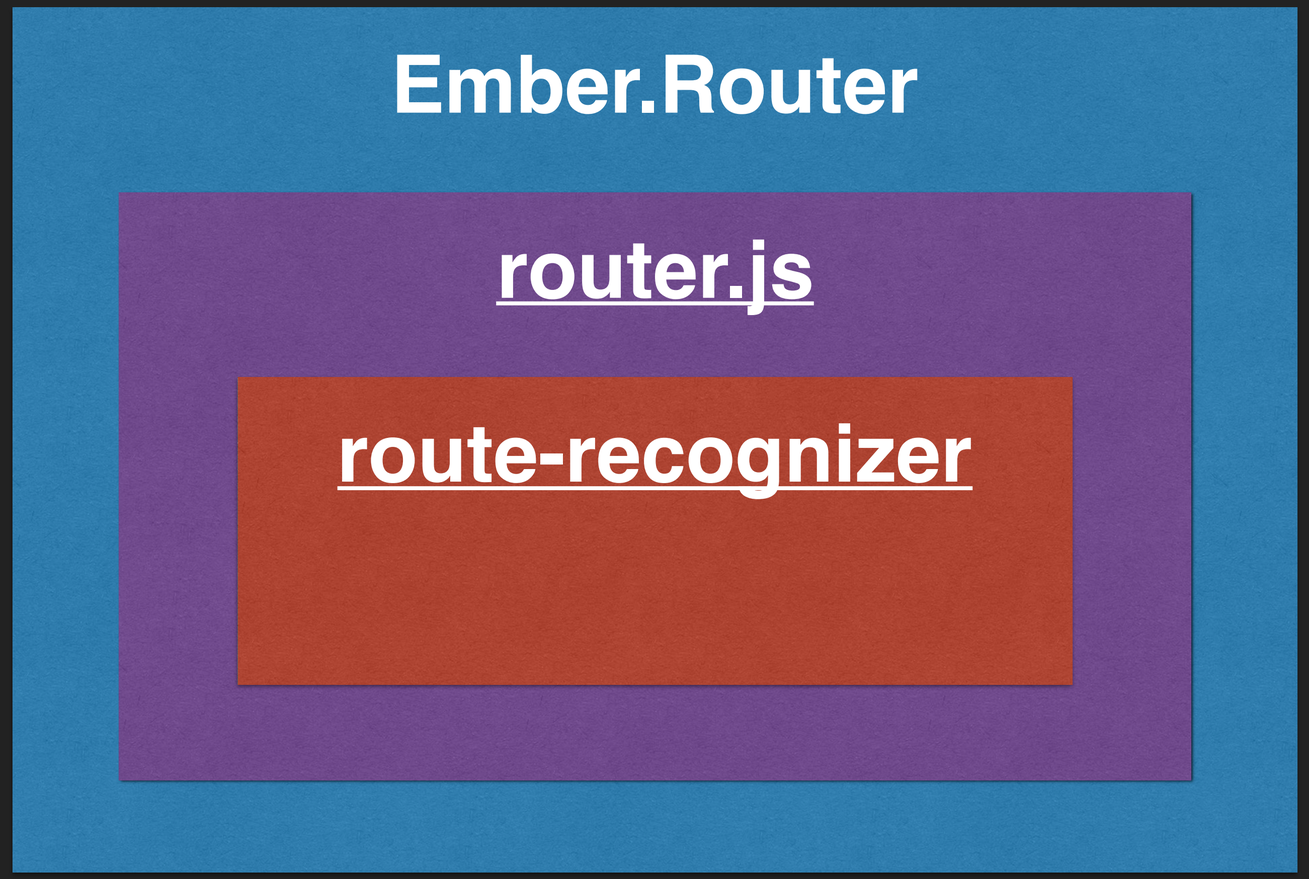
© Mike.Works, Inc. 2017. All rights reserved

Router

- One per app
- Manages transitions between URLs
- Usually leave it as-is, except for Router.map
- Core is a microlibrary: router.js

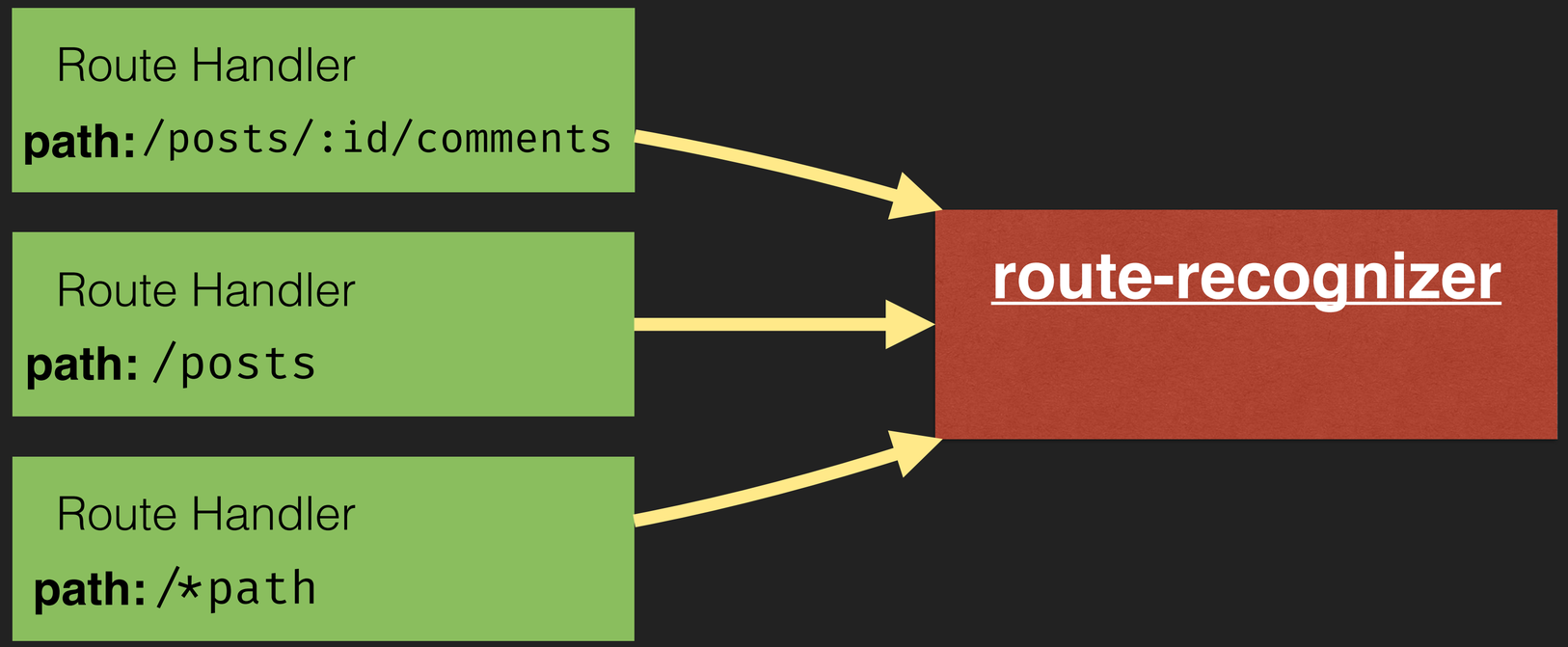
© Mike.Works, Inc. 2017. All rights reserved

Router: Microlibraries



© Mike.Works, Inc. 2017. All rights reserved

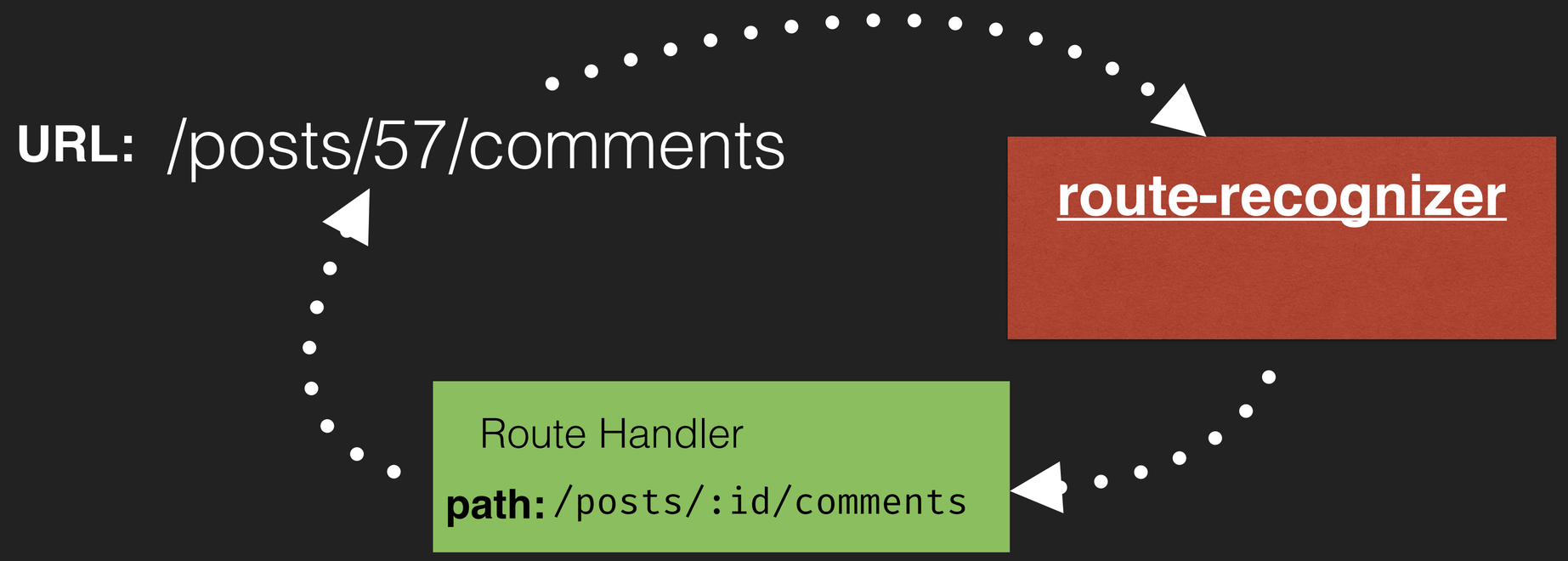
Router: Microlibraries



Route handlers are registered, each with a path

© Mike.Works, Inc. 2017. All rights reserved

Router: Microlibraries



Given a URL, the first matching handler is returned

© Mike.Works, Inc. 2017. All rights reserved

```
router.map(function(match) {
  match("/posts/:id").to("showPost");
  match("/posts").to("postIndex");
  match("/posts/new").to("newPost");
});
```

Router.js

© Mike.Works, Inc. 2017. All rights reserved

Router.js

```
router.map(function(match) {
  match("/posts/:id").to("showPost");
  match("/posts").to("postIndex");
  match("/posts/new").to("newPost");
});
urlWatcher.onUpdate((url) ⇒ router.handleURL(url));
```

© Mike.Works, Inc. 2017. All rights reserved

Router.js

```
router.map(function(match) {
  match("/posts/:id").to("showPost");
  match("/posts").to("postIndex");
  match("/posts/new").to("newPost");
});
urlWatcher.onUpdate((url) ⇒ router.handleURL(url));

var myHandlers = {
  showPost: { }
}
```

© Mike.Works, Inc. 2017. All rights reserved

Router.js

```
router.map(function(match) {
  match("/posts/:id").to("showPost");
  match("/posts").to("postIndex");
  match("/posts/new").to("newPost");
});
urlWatcher.onUpdate((url) ⇒ router.handleURL(url));

var myHandlers = {
  showPost: {
    model: (params) ⇒ $.get("post/" + params.id)
  }
}
```

© Mike.Works, Inc. 2017. All rights reserved

Router.js

```
router.map(function(match) {
  match("/posts/:id").to("showPost");
  match("/posts").to("postIndex");
  match("/posts/new").to("newPost");
});
urlWatcher.onUpdate((url) ⇒ router.handleURL(url));

var myHandlers = {
  showPost: {
    model: (params) ⇒ $.get("post/" + params.id),
    setup: (post) ⇒ {/* render something */}
  }
}
```

© Mike.Works, Inc. 2017. All rights reserved

Router.js

```
router.map(function(match) {
  match("/posts/:id").to("showPost");
  match("/posts").to("postIndex");
  match("/posts/new").to("newPost");
});
urlWatcher.onUpdate((url) => router.handleURL(url));

var myHandlers = {
  showPost: {
    model: (params) => $.get("post/" + params.id),
    setup: (post) => {/* render something */}
  }
}
router.getHandler = (name) => myHandlers[name];
```

© Mike.Works, Inc. 2017. All rights reserved

Router

```
Router.map(function() {
  // IMPLIED index              /
  this.route('posts'); //       /posts
  this.route('post', {path: 'post/:id'}, function() {
    // IMPLIED index            /post/123
    this.route('author'); //    /post/123/author
    this.route('comments');// /post/123/comments
  })
});
```

© Mike.Works, Inc. 2017. All rights reserved

Routing: link-to Component

Text Route Name

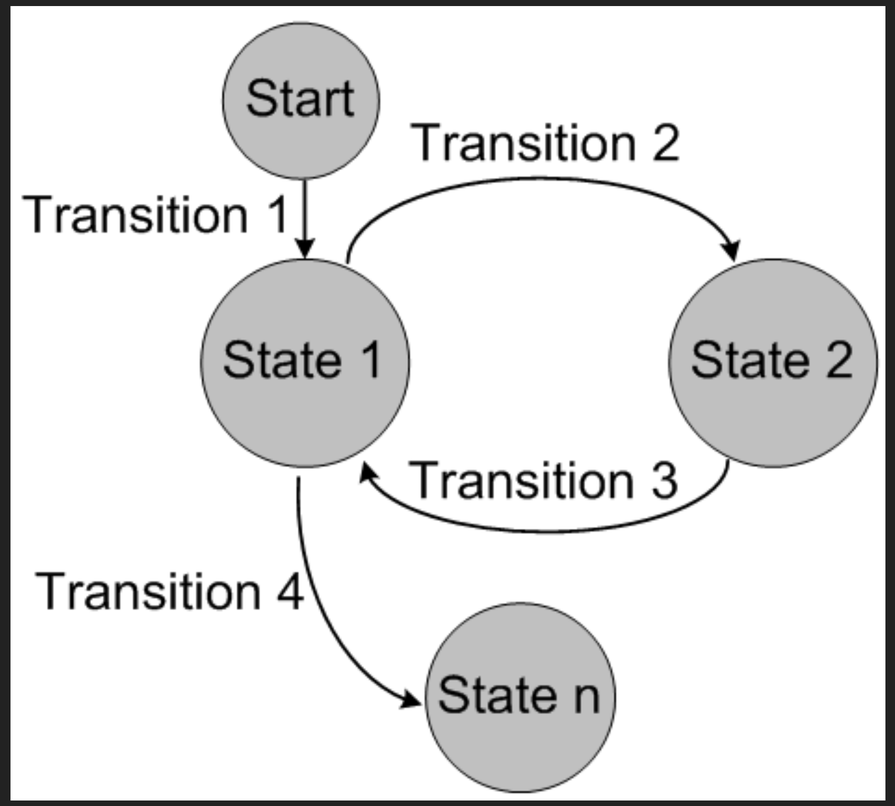
```
{{link-to "Go Home" "index"}}
{{link-to "Go to post 31 comments" "post.comments" 31}}
```

Text Route Name Model or ID

© Mike.Works, Inc. 2017. All rights reserved

Routes

- Router is a finite state machine
- Route manages transitions
- Loads templates and models
- Can handle user events



© Mike.Works, Inc. 2017. All rights reserved

Routes: Lifecycle

Promise Aware

activate	`()`	Called when router enters the route
beforeModel	`(transition)`	First entry validation hook
model	`(params, transition)`	Convert URL to a model
afterModel	`(modelObj, transition)`	Called after model is resolved
redirect	`(modelObj, transition)`	Similar to afterModel
setupController	`(controller, model)`	Setup the controller for the current
renderTemplate	`(controller, model)`	Render the template for the current
deactivate	`()`	Called when router leaves route

© Mike.Works, Inc. 2017. All rights reserved

Routes

`beforeModel(transition)`

THIS IS THE PROPER PLACE FOR…

- A redirect that doesn't require the resolved model
- Async entry validation

BY DEFAULT THIS DOES…

- Nothing

© Mike.Works, Inc. 2017. All rights reserved

Routes

`model(params, transition)`

THIS IS THE PROPER PLACE FOR…

- API call(s) to fetch data based on current URL

BY DEFAULT THIS DOES…

- If route has dynamic segment like `:post_id`, fetch a record `post` with id `post_id`

© Mike.Works, Inc. 2017. All rights reserved

Routes

`model(params, transition)`

- Whatever this returns will be available
  - in template as model
  - in routes via `this.modelFor('post')`
- If you need to make multiple API calls, use `Ember.RSVP.hash()`
- Common location for some light data massaging

© Mike.Works, Inc. 2017. All rights reserved

Routes `afterModel(modelObj, transition)`

THIS IS THE PROPER PLACE FOR…

- Entry validation that requires the resolved model

BY DEFAULT THIS DOES…

- Nothing

© Mike.Works, Inc. 2017. All rights reserved

Routes `redirect(modelObj, transition)`

SEEMS JUST LIKE AFTERMODEL. WHY HAVE THIS HOOK?

- redirect in afterModel aborts the current transition and starts a new one
- Think about the case where you're redirecting into a child route

© Mike.Works, Inc. 2017. All rights reserved

Routes `setupController(controller, model)`

THIS IS THE PROPER PLACE FOR

- Additional setup of the context for your template

BY DEFAULT, THIS DOES…

- A lot of really important stuff. Make sure to call super if you extend!

© Mike.Works, Inc. 2017. All rights reserved

Routes: Transitioning

```
this.transitionTo(routeName, models, options)
```

```
this.transitionTo("post", myPost)
this.transitionTo("post", 31)
this.transitionTo("posts")
```

© Mike.Works, Inc. 2017. All rights reserved

Routes: Transitioning

```
this.replaceWith(routeName, models, options)
```

```
this.replaceWith("post", myPost)
this.replaceWith("post", 31)
this.replaceWith("posts")
```

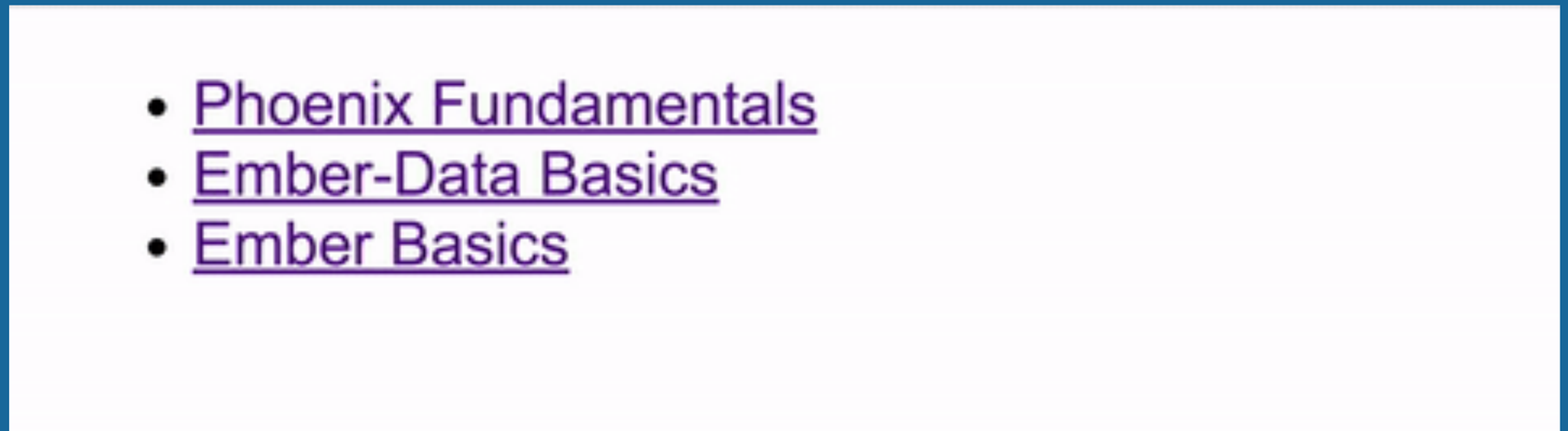
© Mike.Works, Inc. 2017. All rights reserved

```
ember g route course --path="course/:id"
```

Course Page: Fixtures

6

- The "root url" of your app should show a clickable list of courses, allowing the user to drill down into a course of their choice
- Move the data from the application controller to somewhere it can be accessed by all routes that need it
- Root app URL should show a list of courses, and “/course/<course name>” should show data for an individual course



Async Data

© Mike.Works, Inc. 2017. All rights reserved

Getting Data: Ancient Edition™

```
const xhr = new XMLHttpRequest();
xhr.open('GET', url);
xhr.responseType = 'json';

xhr.onload = () => { console.log(xhr.response); };

xhr.onerror = () => { console.error("🤡"); };

xhr.send();
```

© Mike.Works, Inc. & Steve Kinney 2017. All rights reserved

Getting Data: Classic Edition

```
$.ajax('http://example.com').then(() => {
  ...
});
```

- Not much control over request/response
- A browser-only construct
- XMLHttpRequest wasn't really made for how we typically use it

© Mike.Works, Inc. 2017. All rights reserved

Getting Data: Fetch

```
fetch('http://example.com/')
  .then((response) ⇒ response.json())
```

- Can be polyfilled today
- First-class treatment of request/response objects
- Can be fulfilled by a ⚙️Service Worker
- Node.js friendly

© Mike.Works, Inc. 2017. All rights reserved

Getting Data: Fetch

```
fetch('http://example.com/')
  .then((response) ⇒ response.json())
```

- Can be polyfilled today
- First-class treatment of request/response objects
- Can be fulfilled by a ⚙️Service Worker
- Node.js friendly

© Mike.Works, Inc. 2017. All rights reserved

Fetch Polyfill for Ember.js Apps

```
ember install ember-fetch
```

© Mike.Works, Inc. 2017. All rights reserved

Practical Promises

```
new Promise((resolve, reject) => {
  resolve('hello');
});
```

© Mike.Works, Inc. 2017. All rights reserved

Practical Promises

```
new Promise((resolve, reject) ⇒ {
  resolve('hello');
})
.then((val) ⇒  { console.log(val);   })
.catch((err) ⇒ { console.error(err); });
```

© Mike.Works, Inc. 2017. All rights reserved

Practical Promises

```
new RSVP.Promise((resolve, reject) ⇒ {
    resolve('hello');
})
.then((val) ⇒  { console.log(val);        })
.catch((err) ⇒ { console.error(err);      })
.finally(() ⇒  { console.log('all done'); });
```

© Mike.Works, Inc. 2017. All rights reserved

Practical Arrays of Promises

```
[
  'https://api.mike.works/api/v1/courses',
  'https://api.mike.works/api/v1/talks',
  'https://api.mike.works/api/v1/events'
].map((url) => {
  return fetch(url).then((response) => response.json());
});
```

© Mike.Works, Inc. 2017. All rights reserved

Practical Arrays of Promises

```
let requests = [
  'https://api.mike.works/api/v1/courses',
  'https://api.mike.works/api/v1/talks',
  'https://api.mike.works/api/v1/events'
].map((url) => fetch(url).then((response) => response.json()));
```

© Mike.Works, Inc. 2017. All rights reserved

Practical Arrays of Promises

```
let requests = [
  'https://api.mike.works/api/v1/courses',
  'https://api.mike.works/api/v1/talks',
  'https://api.mike.works/api/v1/events'
].map((url) => fetch(url).then((response) => response.json()));

Ember.RSVP.all(requests).then((bodies) => {
  let [courses, talks, events] = bodies;
   ...
});
```

Wait for all promises to resolve

© Mike.Works, Inc. 2017. All rights reserved

Practical Arrays of Promises

```
let requests = [
  'https://api.mike.works/api/v1/courses',
  'https://api.mike.works/api/v1/talks',
  'https://api.mike.works/api/v1/events'
].map((url) => fetch(url).then((response) => response.json()));

Ember.RSVP.allSettled(requests).then((bodies) => {
  let [courses, talks, events] = bodies;
   ...
});
```

Wait for all promises to **SETTLE**

© Mike.Works, Inc. 2017. All rights reserved

Practical Hashes of Promises

```
const getData =
  (url) ⇒ fetch(url).then((response) ⇒ response.json());

let requests = {
  courses: getData('https://api.mike.works/api/v1/courses'),
  talks: getData('https://api.mike.works/api/v1/talks')
}

Ember.RSVP.hash(requests).then((bodies) ⇒ {
  let { courses, talks } = bodies;
});
```

Wait for all promises to resolve

© Mike.Works, Inc. 2017. All rights reserved

Practical Hashes of Promises

```
const getData =
  (url) ⇒ fetch(url).then((response) ⇒ response.json());

let requests = {
  courses: getData('https://api.mike.works/api/v1/courses'),
  talks: getData('https://api.mike.works/api/v1/talks')
}

Ember.RSVP.hashSettled(requests).then((bodies) ⇒ {
  let { courses, talks } = bodies;
});
```

Wait for all promises to **SETTLE**

© Mike.Works, Inc. 2017. All rights reserved

Course Page: Real Data

7

```
ember install ember-fetch
```

- Hook up the course list, and individual course pages to real data using fetch
- Perform any JSON transformations necessary in the appropriate route model hook(s)
- HINT: https://api.mike.works/api/v1/courses
- Update link-to to use the 'slug' property instead of 'name'
- Don't get involved with ember-data yet

© Mike.Works, Inc. 2017. All rights reserved

Ember-Data

© Mike.Works, Inc. 2017. All rights reserved

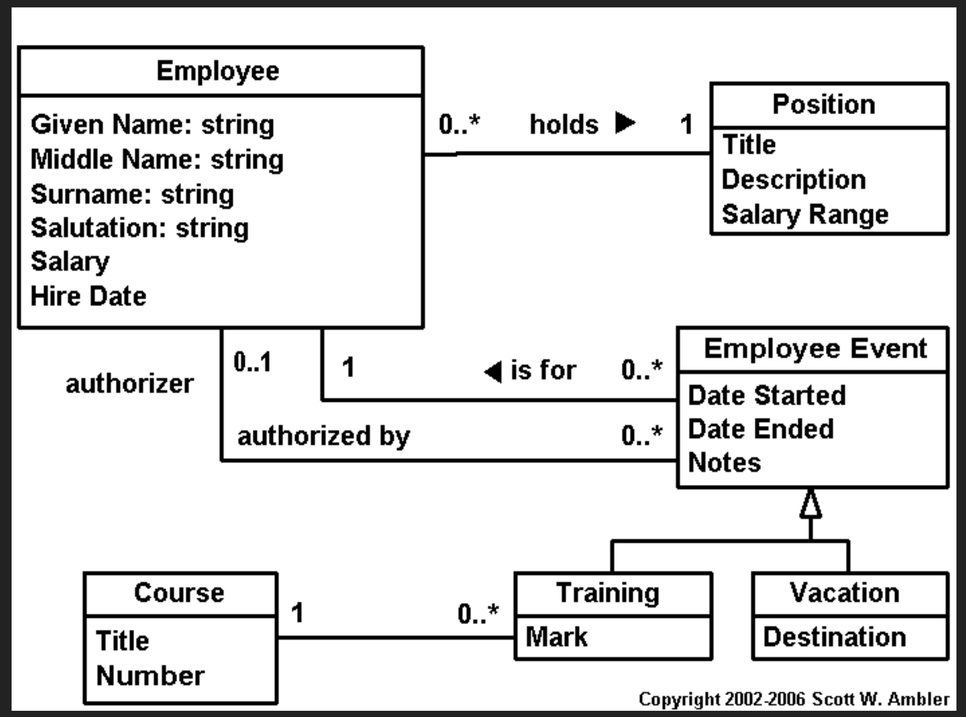
emberdata

- A persistence library
- Key objects: Store, Model, Adapter, Serializer
- Internal data structures: JSON-API (http://jsonapi.org/)
- Define how to interact with your APIs, and then no longer need to explicitly interact with them explicitly

© Mike.Works, Inc. 2017. All rights reserved

Models

- A representation of any data you may present to the user (usually persistent)
- Mostly defined by attributes, and relationships to other models
- "model" is the factory that defines the structure of "records"



© Mike.Works, Inc. 2017. All rights reserved

Models

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

export default DS.Model.extend({
  title:        DS.attr(),
  publishedAt:  DS.attr('date'),
  chapters:     DS.hasMany('chapter')
});
```

© Mike.Works, Inc. 2017. All rights reserved

Store

- Where you get/create/destroy records
- A single source of truth
- This means that all changes are kept in sync!
- Similar concept in Facebook/flux, angular-data

© Mike.Works, Inc. 2017. All rights reserved

Store

```
// API request for all records of type "author"
this.store.findAll('author');
// API request for record of type "post" with id 37
this.store.findRecord('post', 37);
```

© Mike.Works, Inc. 2017. All rights reserved

Store

```
// API request for all records of type "author"
this.store.findAll('author');
// API request for record of type "post" with id 37
this.store.findRecord('post', 37);

// look in cache for all records of type "author"
this.store.peekAll('author');
// look in cache for record of type "post" with id 37
this.store.peekRecord('post', 37);
```

© Mike.Works, Inc. 2017. All rights reserved

Adapters

- Handle the particulars of making requests
- You'll typically never touch them directly
- Which Adapter for a given type?
  - First attempt to find a type-specific adapter
  - Fall back to application adapter

© Mike.Works, Inc. 2017. All rights reserved

Adapters

```
export default DS.RESTAdapter.extend({
  host: 'https://api.github.com',

  urlForQuery (query, modelName) {
    switch (modelName) {
      case 'repo': return `${this.get('host')}/orgs/${query.orgId}/repos`;
      default:     return this._super( ... arguments);
    }
  }
});
```

© Mike.Works, Inc. 2017. All rights reserved

Serializers

- Handle the particulars of converting JSON between API and client-side representations
- normalizeResponse - lowest level normalization
- many other potentially more convenient normalization hooks

© Mike.Works, Inc. 2017. All rights reserved

Serializers

```
export default DS.RESTSerializer.extend({

  normalizeResponse (store, modelClass, payload, id, requestType){
    const newPayload = { org: payload };
    return this._super(store, modelClass, newPayload, id, requestType);
  }

});
```

© Mike.Works, Inc. 2017. All rights reserved

Asking the Store for Data

```
DS.Store#findAll
    DS.Store#_fetchAll
```

Asking the Store for Data

```
DS.Store#findAll
    DS.Store#_fetchAll(modelClass, array, options)
```

STRING ARRAY OBJECT

Asking the Store for Data

```
DS.Store#findAll
    DS.Store#_fetchAll(“course",    [],    {} )
```

STRING ARRAY OBJECT

Asking the Store for Data

```
DS.Store#findAll
    DS.Store#_fetchAll("course",    [],     {} )
```

Options

```
{
  reload: true, // New request from scratch
  backgroundReload: true // Cached data now, then refresh
}
```

Asking the Store for Data

`reload`	`backgroundReload`	
🚫	✅	Cached data now, updated live w/ new stuff
✅	✅	No point
🚫	🚫	Cached data only (peek)
✅	🚫	Fetch fresh data only

Asking the Store for Data

```
DS.Store#findAll
  DS.Store#_fetchAll(modelClass, this.peekAll(modelClass), {})
```

Asking the Store for Data

```
DS.Store#findAll
  DS.Store#_fetchAll {
    let adapter = this.adapterFor('course');
    if (options.reload || adapter.shouldReloadAll(array)) {
      return this._findAll();
    }
    if (!options.backgroundReload) {
      return array;
    }
  }
```

Asking the Store for Data

```
DS.Store#findAll
    DS.Store#_fetchAll
        DS.Store#_findAll {
            let promise = adapter.findAll();

            let serializer =
              adapter.serializer ||
              store.serializerFor('course');

            return promise.then((adapterPayload) => {
              return serializer.normalizeResponse(adapterPayload);
            });
        }
```

```
1   switch (requestType) {
2     case 'findRecord': return this.urlForFindRecord(id, model
3     case 'findAll': return this.urlForFindAll(modelName, snap
4     case 'query': return this.urlForQuery(query, modelName);
5     case 'queryRecord': return this.urlForQueryRecord(query,
6     case 'findMany': return this.urlForFindMany(id, modelName
7     case 'findHasMany': return this.urlForFindHasMany(id, mod
8     case 'findBelongsTo': return this.urlForFindBelongsTo(id,
9     case 'createRecord': return this.urlForCreateRecord(model
10    case 'updateRecord': return this.urlForUpdateRecord(id, m
11    case 'deleteRecord': return this.urlForDeleteRecord(id, m
12    default: return this._buildURL(modelName, id);
13  }
```

Asking the Store for Data

```
DS.Store#findAll
    DS.Store#_fetchAll
        DS.Store#_findAll {
            let promise = adapter.findAll();

            let serializer =
              adapter.serializer ||
              store.serializerFor('course');

            return promise.then((adapterPayload) => {
              let payload = serializer.normalizeResponse(adapterPay
              store.push(payload);
            });
        }
```

Debugging Adapters

Developer Tools - http://localhost:4200/

Elements | Console | Sources | Network | Timeline | Profiles | Application | Ember

http://localhost:4200/

Search

Container

Render Performance

Submit an Issue

route (2)

router (1)

service (5)

template (4)

store

-document

-dom-tree...

<glue@service:st... > $E

Own Properties

typeMaps: { ember2...

recordArrayManager: > $E

glue@service:store:

(subclass of Ember.Service)

Console | Sensors

top

Preserve log

```
Ember Inspector ($E):                                                    VM8278:87
  Class {_backburner: Backburner, _pendingSave: Array[0], _instanceCache:
▶ ContainerInstanceCache, _pendingFetch: MapWithDefault, __ember1485502700119:
  "ember312"}
```

Debugging Adapters

Developer Tools - http://localhost:4200/

Elements | Console | Sources | Network | Timeline | Profiles | Application | Ember

http://localhost:4200/

Search

Container

Render Performance

Submit an Issue

route (2)

router (1)

service (5)

template (4)

store

-document

-dom-tree...

<glue@service:st... > $E

Own Properties

typeMaps: { ember2...

recordArrayManager: > $E

glue@service:store:

(subclass of Ember.Service)

Console | Sensors

top

Preserve log

```
Ember Inspector ($E):                                                    VM8278:87
  Class {_backburner: Backburner, _pendingSave: Array[0], _instanceCache:
▶ ContainerInstanceCache, _pendingFetch: MapWithDefault, __ember1485502700119:
  "ember312"}
> let courseAdapter = $E.adapterFor('course');
<· undefined
```

Debugging Adapters

Developer Tools - http://localhost:4200/

Elements | Console | Sources | Network | Timeline | Profiles | Application | Ember

http://localhost:4200/ | Search | <glue@service:st… > $E

Container
Render Performance
Submit an Issue

route (2)
router (1)
service (5)
template (4)

store
-document
-dom-tree…

▼ Own Properties
typeMaps: { ember2…
recordArrayManager: > $E
glue@service:store:
▶ (subclass of Ember.Service)

Console | Sensors

top | Preserve log

```
Ember Inspector ($E):                                                    VM8278:87
  Class {_backburner: Backburner, _pendingSave: Array[0], _instanceCache:
▶ ContainerInstanceCache, _pendingFetch: MapWithDefault, __ember1485502700119:
  "ember312"}
> let courseAdapter = $E.adapterFor('course');
<· undefined
> courseAdapter.buildURL('course', 12, null, 'findRecord');
<· "https://localhost:4000/api/v1/courses/12"
```

Debugging Serializers

Developer Tools - http://localhost:4200/

Elements Console Sources Network Timeline Profiles Application Ember

http://localhost:4200/ Search <glue@service:stor… $E

Container
Render Performance
Submit an Issue

route (2)
router (1)
service (5)
template (4)

store
-docum…
-dom-tr…

▼ Own Properties
typeMaps: { ember377…
recordArrayManager: …
glue@service:store:
▶ (subclass of Ember.Service)

Console Sensors

top Preserve log

```
> let courseSerializer = $E.serializerFor('course');
<· undefined
> courseSerializer.normalizeResponse($E, 'course', {data: []}, null, 'findAll');
<· ▶ Object {data: Array[0]}
>
```

Course Page: Ember-Data

- Consume the mike.works API using ember-data
- You should have a single “application” adapter and serializer
- The API conforms to the JSON-API spec already
  - ember g adapter application
  - ember g serializer application
  - ember g model course

© Mike.Works, Inc. 2017. All rights reserved

Testing

© Mike.Works, Inc. 2017. All rights reserved

Flavors of Tests

- **Unit** - for testing algorithmic complexity
- **Integration** - for testing contracts between pieces of code, and how things work together
- **Acceptance** (functional) - testing user workflows in the context of your entire application

© Mike.Works, Inc. 2017. All rights reserved

Testing Strategies to Apply

Type of object	Type of test
utils	Unit
models	Unit
services	Unit
components	Integration
routes	Acceptance
ember-data stuff	Acceptance

© Mike.Works, Inc. 2017. All rights reserved

Unit Tests

- Automatically generated by ember-cli
- Testing in isolation
- Bring other objects into the test via needs
- Very fast

© Mike.Works, Inc. 2017. All rights reserved

Component Integration Tests

- Test a component in isolation
- Setup test case via
  - A scenario template
  - Defining the “outside world”
- Inline HBS!

© Mike.Works, Inc. 2017. All rights reserved

Acceptance Tests

- Testing the app as a whole
- Much Slower
- May involve attempts to make API requests
- Best bet for testing real user workflows
- Involves running in “Testing Mode”

© Mike.Works, Inc. 2017. All rights reserved

Acceptance Test Helpers

SYNCHRONOUS

- currentPath()
- currentRouteName()
- currentURL()
- find(selector, context)

ASYNCHRONOUS

- click(selector)
- fillIn(selector, value)
- keyEvent(selector, type, keyCode)
- triggerEvent(selector, type, options)
- visit(url)

© Mike.Works, Inc. 2017. All rights reserved

Acceptance Test

```
// Visit a URL
visit('/org/emberjs/repos');

// Wait for everything to settle
andThen(function() {
  // Make sure we're at the right URL
  assert.equal(currentURL(), '/org/emberjs/repos');
});

// Click a link (which starts a transition)
click('a[href*="org/emberjs/data"]');

// Wait for everything to settle
andThen(function() {
  // Make sure we're at the right URL
  assert.equal(currentURL(), '/org/emberjs/data/issues');
});
```

© Mike.Works, Inc. 2017. All rights reserved

Running in Testing Mode

Normal Mode	Testing Mode
Property change A	Property change A
Property change B	DOM update A
Property change C	Property change B
DOM update A	DOM update B
DOM update B	Property change C
DOM update C	DOM update C

© Mike.Works, Inc. 2017. All rights reserved

Writing Acceptance Tests

9

https://api.mike.works/api/v1/courses/ember-cli-basics-ec71a23/stages

- If you look at the API response for a course, you can see that there's a relationship to a "course-stage" object
- Build another layer of child routes for these "course-stage" objects, and create an acceptance test that…
  - Starts from the index route
  - Clicks on a course
  - Clicks on a course-stage

```
ember g model course-stage
ember g model course-agenda-item
ember g route course-stage \
  -- path="course/:course_id/:stage"
ember g acceptance-test test1
```

© Mike.Works, Inc. 2017. All rights reserved

http://go/ember-state

State Management

© Mike.Works, Inc. 2017. All rights reserved

IDENTIFYING STATE

ITERATIVE EXAMPLE

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8

Search

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 2

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

Where does your state belong

By: Mike North
Written At: October 10, 2016 2:28 AM
Updated At: October 10, 2016 2:42 AM

For over a decade, most of us built web UIs operating under the assumption that servers would render mostly-static HTML, and we'd boil the ocean and free all client-side memory with every page load. This was a simple world, where the server-side was aware of the user's intent and context.

Enter the Single Page Application (SPA) - there are all sorts of usability and performance, and scalability benefits that come along with building a web app this way, but there are also some serious challenges. There are some implicit assumptions that our users make about how apps should work, and we must work a bit harder in order to keep them intact. Take the 'Back' and 'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state decisions become even more consequential and complex.

In this talk, I'll outline four types of state

IDENTIFYING STATE

ITERATIVE EXAMPLE

STATE IN YOUR EMBER APP

4 FLAVORS OF STATE

UI STATE

PERSISTED STATE

ADDRESSABLE STATE

DRAFT STATE

localhost:4200/posts/A96usFSIY4G0W4kwAqsw

Search

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

Seven Tips From Ernest Hemingway on How to Write Fiction

details...

To get started, write one true sentence

Sometimes when I was starting a new story and I could not get it going, I would sit in front of the fire and squeeze the peel of the little oranges into the edge of the flame and watch the sputter of blue that they made. I would stand and look out over the roofs of Paris and think, “Do not worry. You have always written before and you will write now. All you have to do is write one true sentence. Write the truest sentence that you know.”

So finally I would write one true sentence, and then go on from there. It was easy then because there was always one true sentence that I knew or had seen or had heard someone say. If I started to write elaborately, or like someone introducing or presenting something, I found that I could cut that scrollwork or ornament out and throw it away and start with the first true simple declarative sentence I had written.

Always stop for the day while you still know what will happen next

The best way is always to stop when you are going good and

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8UG?q=Wher

Where

Where does your state belong 1

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

our users make about how apps should work, and we must work a bit harder in order to keep them intact. Take the 'Back' and 'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state decisions become even more consequential and complex.

In this talk, I'll outline four types of state

- Navigation state
- Persisted state
- UI state
- "Will be persisted" state

and provide examples for each. Along the way, we'll start to assemble a framework of questions that you can ask yourself when encountering new pieces of state, to lead you down the right path(s).

This is another comment

21 minutes ago

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1

Mike North x DRAFT October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state decisions become even more consequential and complex.

In this talk, I'll outline four types of state

- Navigation state
- Persisted state
- UI state
- "Will be persisted" state

and provide examples for each. Along the way, we'll start to assemble a framework of questions that you can ask yourself when encountering new pieces of state, to lead you down the right path(s).

This is another comment

21 minutes ago

This talk is going pretty well!

Reply Cancel

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 2

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state decisions become even more consequential and complex.

In this talk, I'll outline four types of state

- Navigation state
- Persisted state
- UI state
- "Will be persisted" state

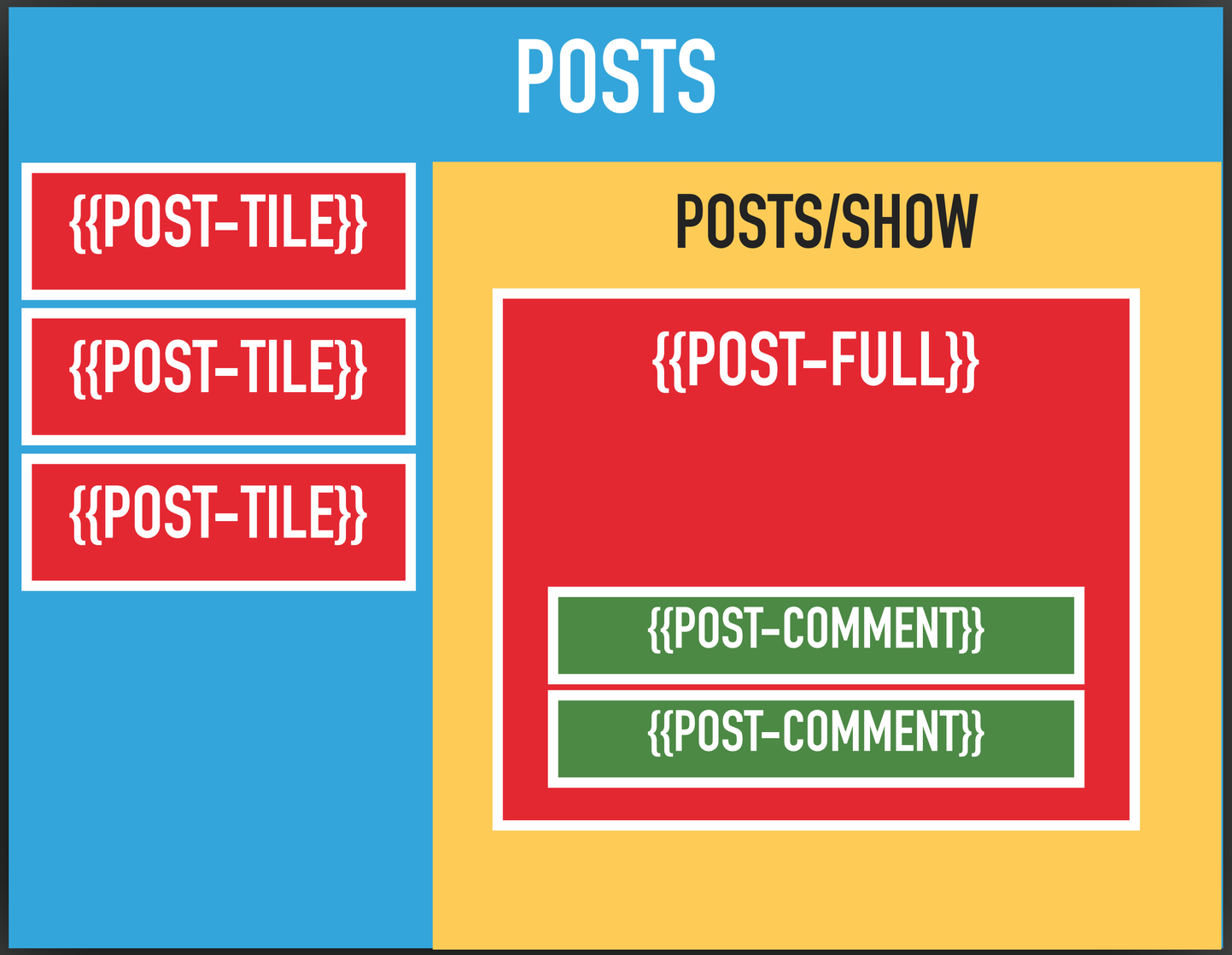
and provide examples for each. Along the way, we'll start to assemble a framework of questions that you can ask yourself when encountering new pieces of state, to lead you down the right path(s).

This is another comment

21 minutes ago

This talk is going pretty well!

a few seconds ago


POSTS
{{POST-TILE}}
{{POST-TILE}}
{{POST-TILE}}
POSTS/SHOW
{{POST-FULL}}
{{POST-COMMENT}}
{{POST-COMMENT}}

ADDRESSABLE STATE

Core Concept - Addressable State

SHARABLE, HISTORICALLY ARCHIVED AND URL-DRIVEN

EXAMPLES

- Sorted/filtered search results
- Tabs/Navigation
- A section of a Github Readme

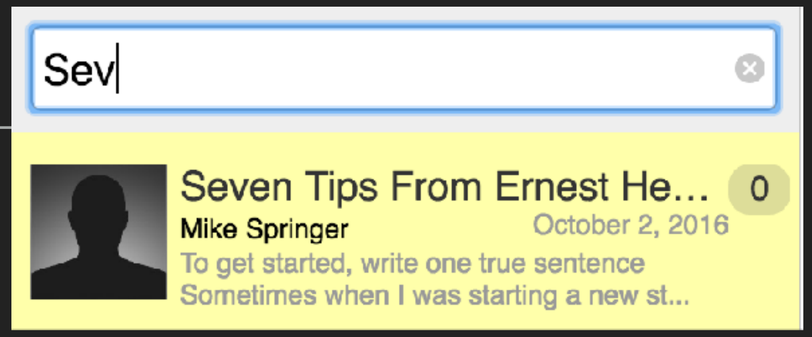
HOW TO IDENTIFY ADDRESSABLE STATE

- Affects GET API calls
  - A perspective on a record or collection of records
- Useful when shared
- Mismanaged when - **back** button breaks

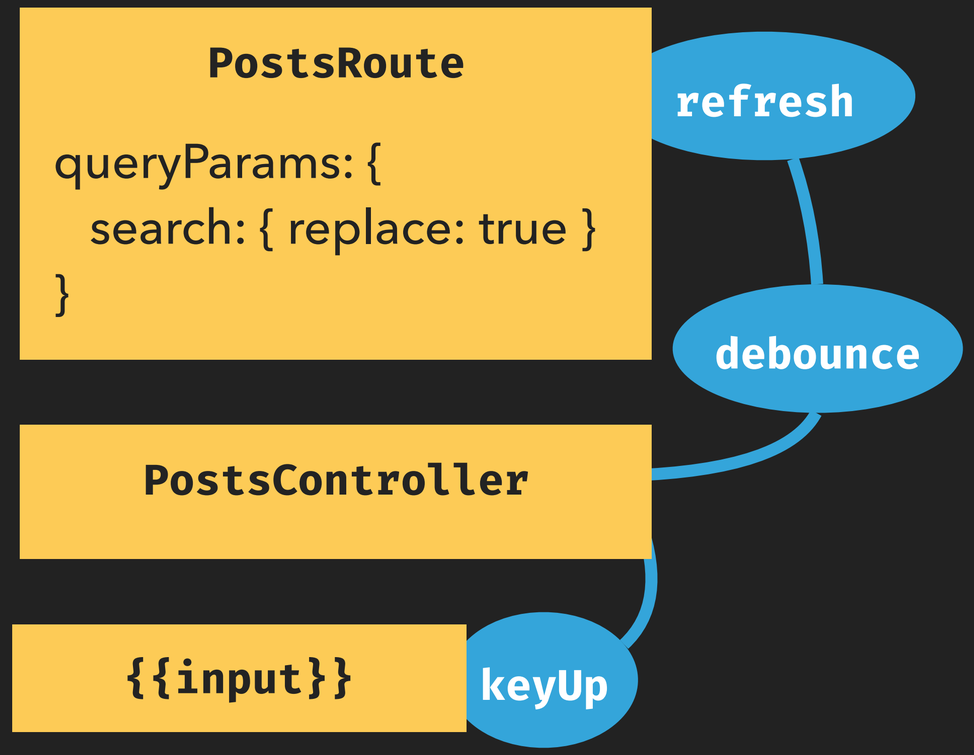
CONTRACT: USERS AND THE WEB

- **Refresh**: never destructive by surprise
- **Bookmarks**: get back to where I was
- **Back**: previous major thing
- **Back a lot**: leave the app

ENHANCING OUR BASIC APP



- Filter by a name fragment
- Bookmark & Share
- Don't abuse our API
- **D**ata **D**own, **A**ctions **U**p



posts.hbs

TEMPLATE

posts.hbs - statetalk

EXPLORER

OPEN EDITORS 1 UNSAV...

STATETALK

- posts
  - .gitkeep
  - index.js
  - posts.js
- serializers
- styles
- templates
  - components
  - posts
  - application.hbs
  - posts.hbs
- app.js
- index.html
- resolver.js
- router.js
- bower_components
- config
- dist
- node_modules
- public
- tests

posts.hbs

```
<div class="posts-index">
  <div class="posts-left">
    <div class="posts-left-search">
      {{input
        class='posts-search-field'
        type='search'
        value=search
        keyUp=(action 'queuePageRefresh')
        placeholder='🔎 Search'}}
    </div>
    <div class="posts-left-content">
      <div class="posts-left-content-pane">
      {{#each model as |post|}}
        {{post-tile model=post}}
      {{/each}}
      </div>
    </div>
  </div>
  <div class="posts-right">
    {{outlet}}
  </div>
</div>
```

master ⟳ ⊗ 0 ⚠ 0 Ln 8, Col 40 Spaces: 2 UTF-8 LF Handlebars ☺

posts.js

CONTROLLER

posts.js - statetalk

EXPLORER

OPEN EDITORS 1 UNSAV...

STATETALK

- app
  - adapters
  - components
  - controllers
    - posts
      - .gitkeep
      - posts.js
  - helpers
  - mixins
  - models
  - routes
    - posts
      - .gitkeep
      - index.js
      - posts.js
  - serializers
  - styles
  - templates
    - components
    - posts
      - application.hbs
      - posts.hbs

posts.js

```javascript
import Ember from 'ember';

const { Controller, run: { debounce } } = Ember;

export default Controller.extend({
  _triggerPageRefresh() {
      this.send('pageRefresh');
  },
  actions: {
      queuePageRefresh() {
          debounce(this, '_triggerPageRefresh', 300);
      }
  }
});
```

master* 0 0 Ln 11, Col 54 Spaces: 4 UTF-8 LF JavaScript

posts.js

ROUTE

posts.js - statetalk

EXPLORER

OPEN EDITORS 1 UNSAV...

STATETALK

- posts
  - .gitkeep
  - posts.js
- helpers
- mixins
- models
- routes
  - posts
    - .gitkeep
    - index.js
    - posts.js
- serializers
- styles
- templates
  - components
  - posts
    - application.hbs
    - posts.hbs
  - app.js
  - index.html
  - resolver.js

posts.js controllers | posts.js routes

```
import Ember from 'ember';

const { Route } = Ember;

export default Route.extend({
  queryParams: {
    search: {
      as: 's',
      replace: true
    }
  },
  model() {
    return this.store.findAll('post');
  }
});
```

master* 0 0 Ln 11, Col 5 Spaces: 2 UTF-8 LF JavaScript

posts.js - statetalk

EXPLORER

OPEN EDITORS 1 UNSAV...

STATETALK

- posts
  - .gitkeep
  - posts.js
- helpers
- mixins
- models
- routes
  - posts
    - .gitkeep
    - index.js
    - posts.js
- serializers
- styles
- templates
  - components
  - posts
    - application.hbs
    - posts.hbs
  - app.js
  - index.html
  - resolver.js

posts.js controllers | posts.js routes

```
import Ember from 'ember';

const { Route } = Ember;

export default Route.extend({
  queryParams: {
    search: {
      as: 's',
      replace: true
    }
  },
  actions: {
    pageRefresh() {
      this.refresh();
    }
  },
  model() {
    return this.store.findAll('post');
  }
});
```

master* 3 3 | Ln 16, Col 5 | Spaces: 2 | UTF-8 | LF | JavaScript

posts.js - statetalk

EXPLORER

OPEN EDITORS

STATETALK

- posts
  - .gitkeep
  - posts.js
- helpers
- mixins
- models
- routes
  - posts
    - .gitkeep
    - index.js
    - posts.js
- serializers
- styles
- templates
  - components
  - posts
    - application.hbs
    - posts.hbs
  - app.js
  - index.html
  - resolver.js

posts.js controllers | posts.js routes

```
import Ember from 'ember';

const { Route } = Ember;

export default Route.extend({
  queryParams: {
    search: {
      as: 's',
      replace: true
    }
  },
  actions: {
    pageRefresh() {
      this.refresh();
    }
  },
  model({ search }) {
    if (!search) {
      return this.store.findAll('post');
    } else {
      return this.store.query('post', { search });
    }
  }
});
```

master* 0 0 | Ln 21, Col 39 | Spaces: 2 | UTF-8 | LF | JavaScript

localhost:4200/posts

Search

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 2

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

Please select a post

🏷️ 1-navigation

GOTCHAS AND COMMON ISSUES

- Trapping users
- Deleting useful history
- Unimportant history states
- Some Route queryParam options are mutually exclusive (right now) (`replace` vs `refreshModel`)
- Tricky: Add to Home Screen

DRAFT STATE

Core Concept – Draft State

LOCALLY-STORED TEMPORARILY, WILL EVENTUALLY BE PERSISTED

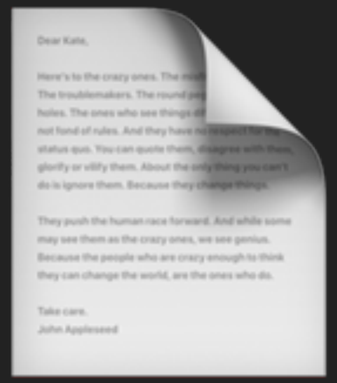
EXAMPLES

- A reply to an email message
- The body of a huge GitHub comment
- In-flight records

HOW TO IDENTIFY

- Will turn into persisted state
- High user effort
- Survives **transitions**
- Detrimental to UX if lost

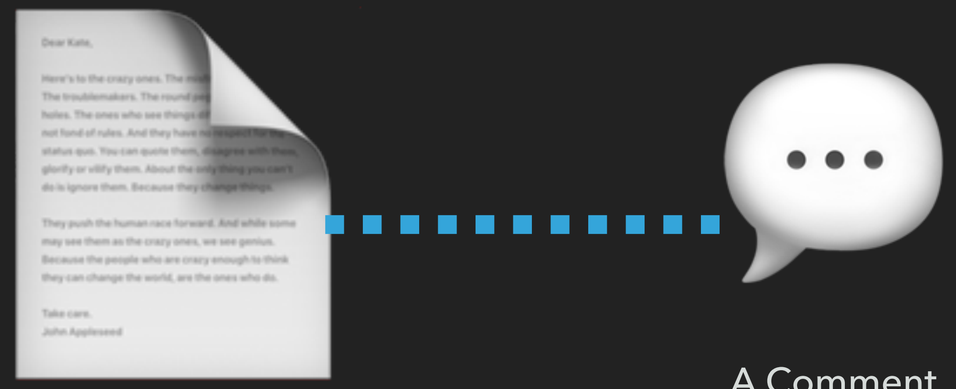
CONCEPTUAL MODEL



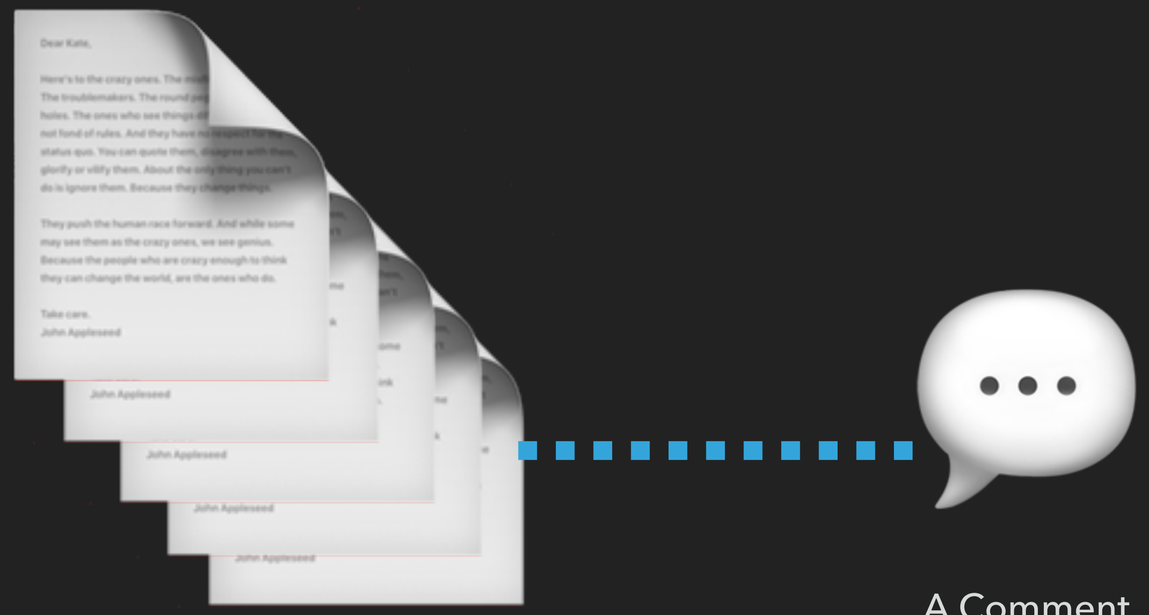
A Post

A Comment

CONCEPTUAL MODEL

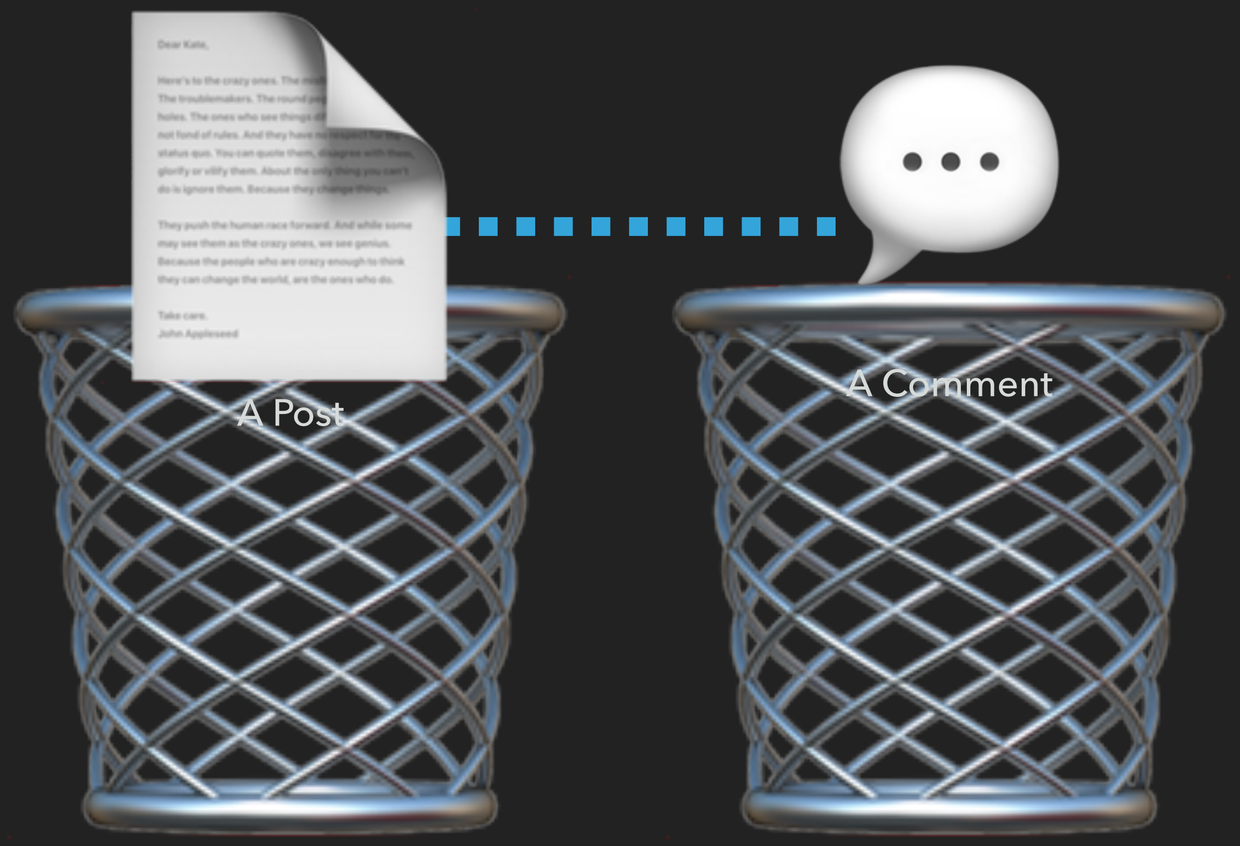


CONCEPTUAL MODEL



CONCEPTUAL MODEL

CONCEPTUAL MODEL



CONCEPTUAL MODEL - KEY CONCEPT

WEAK MAP

CONCEPTUAL MODEL - KEY CONCEPT

WEAK MAP

If an object that is being used as the key of a WeakMap key/value pair is only reachable by following a chain of references that start within that WeakMap, that key/value pair is inaccessible and is automatically removed from the WeakMap.

http://www.ecma-international.org/ecma-262/6.0/#sec-weakmap-objects

CONCEPTUAL MODEL - KEY CONCEPT

WEAK MAP

If an object that is being used as the key of a WeakMap key/value pair is only reachable by following a chain of references that start within that WeakMap, that key/value pair is inaccessible and is automatically removed from the WeakMap.

http://www.ecma-international.org/ecma-262/6.0/#sec-weakmap-objects

CONCEPTUAL MODEL – KEY CONCEPT

WEAK MAP

reachable by following a chain of references that start within that WeakMap, that key/value pair is inaccessible and is automatically removed from the WeakMap.

```
// Crea
let com
// Post
comment
```

http://www.ecma-international.org/ecma-262/6.0/#sec-weakmap-objects

CONCEPTUAL MODEL - KEY CONCEPT

WEAK MAP*

A Post

A Comment

```
// Store the value on the key
post._some_random_property_name = comment;

// Access the value, given the key
WeakMap.prototype.get = function(key) {
  return key._some_random_property_name;
};
```

THE SOLUTION FOR EMBER APPS

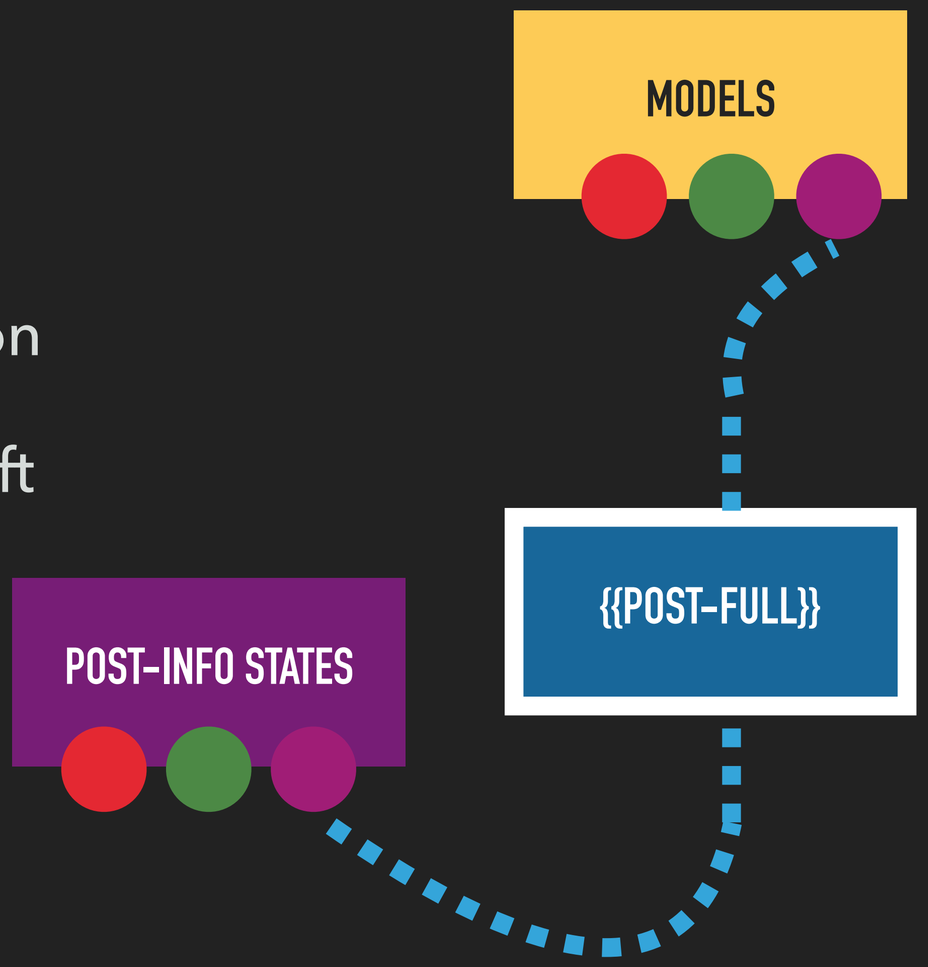
@thoov

@stefanpenner

- **ember-state-services**
- Built on top of WeakMap
- **StateFactory** - creates draft state objects
- **stateFor** - computed property that returns a particular state

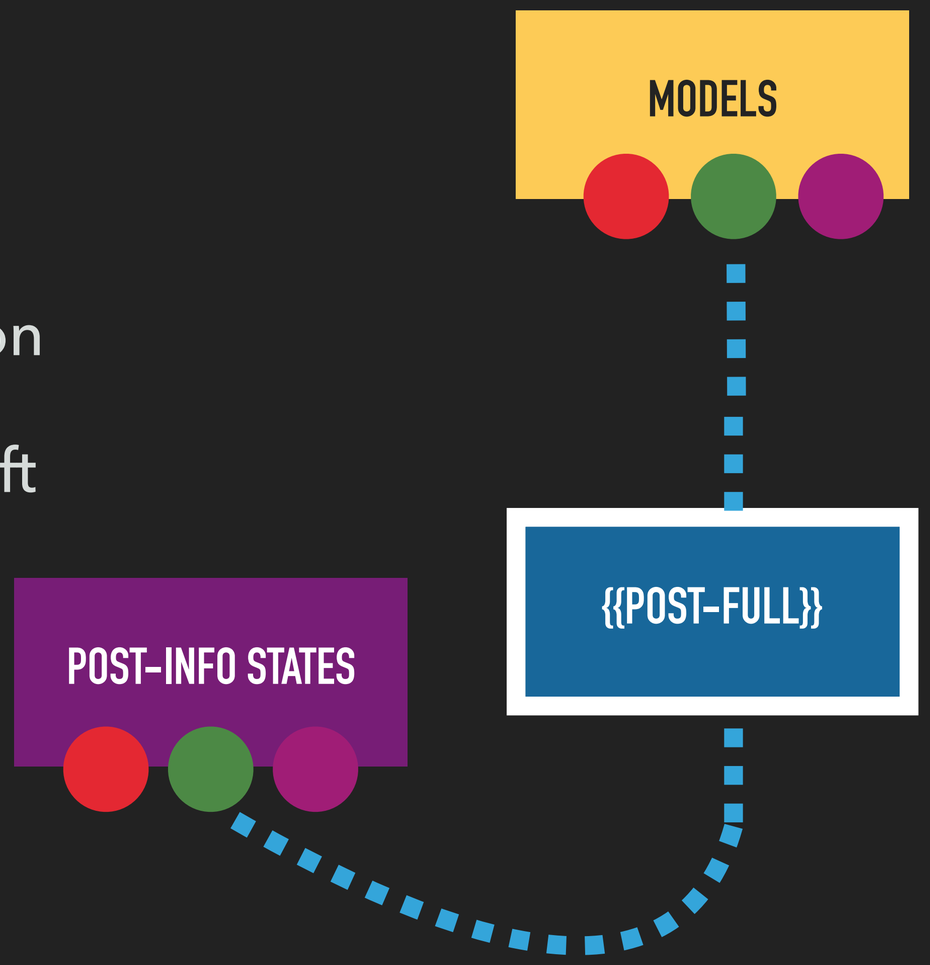
ENHANCING OUR APP

- They must “stick” with the post they’re commenting on
- If we delete a post, the draft must be deleted as well
- No leaks



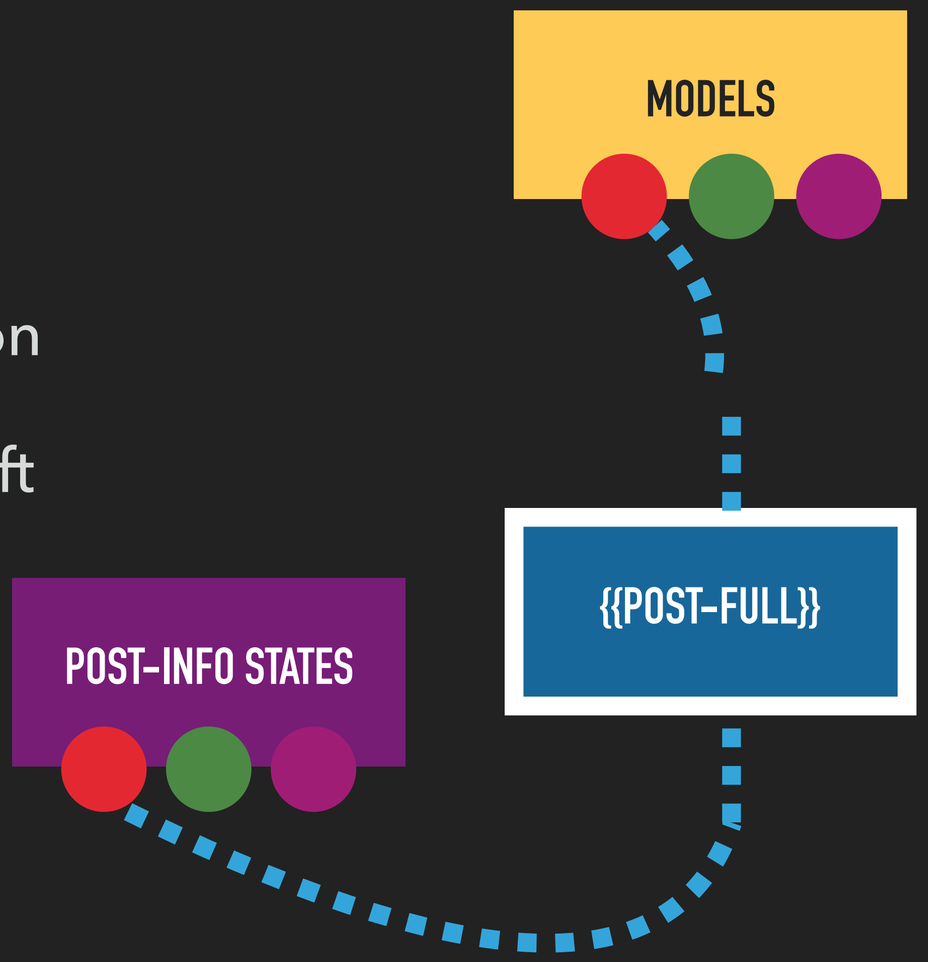
ENHANCING OUR APP

- They must “stick” with the post they’re commenting on
- If we delete a post, the draft must be deleted as well
- No leaks



ENHANCING OUR APP

- They must “stick” with the post they’re commenting on
- If we delete a post, the draft must be deleted as well
- No leaks



states/post-info.js

STATE FACTORY

post-info.js - statetalk

EXPLORER

OPEN EDITORS

STA...

- app
  - adapters
  - components
  - controllers
  - helpers
  - mixins
  - models
  - routes
  - serializers
  - states
    - post-info.js
  - styles
  - templates
  - app.js
  - index.html
  - resolver.js
  - router.js
- bower_components
- config
- dist
- node_modules
- public

post-info.js

```
import Ember from 'ember';

const PostInfoState = Ember.Object.extend({});

PostInfoState.reopenClass({
    initializeState(/*instance*/) {
        return {
            body: ''
        }
    }
});

export default PostInfoState;
```

master* 0↓ 1↑ 0 1 Ln 13, Col 30 Spaces: 4 UTF-8 LF JavaScript

Down the Rabbit Hole

By: Lewis Carroll
Written At: October 2, 2016 11:07 PM

components/post-full.js

COMPONENT

post-full.hbs - statetalk

post-full.js

```
1  import Ember from 'ember';
2  import stateFor from 'ember-state-services/stat
3
4  const { Component } = Ember;
5
6  export default Component.extend({
7    classNames: ['post-full'],
8    postInfo: stateFor('post-info', 'model')
9  });
10
```

post-full.hbs

```
1  <div class="title-row">…
26 </div>
27
28
29 {{#if model.featuredImageUrl}}
30 <div class="feature-image-row">…
32 </div>
33 {{/if}}
34
35 <p class="body">…
37 </p>
38
39 <hr>
40
41 {{#each model.comments as |comment|}}…
43 {{else}}…
47 {{/each}}
48 {{textarea value=postInfo.body class='repl
49
50 <div class="post-actions">…
52 </div>
53
```

master* 0↓ 1↑ 0 0 Ln 48, Col 32 Spaces: 2 UTF-8 LF Handlebars

Seven Tips From Ernest He... 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

components/post-tile.js

COMPONENT

Where does your state belong 2

Mark Nottingham October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

post-tile.js - statetalk

post-tile.js | post-tile.hbs

```js
import Ember from 'ember';
import stateFor from 'ember-state-services/state-for';

const { Component } = Ember;

export default Component.extend({
  classNames: ['post-tile'],
  postInfo: stateFor('post-info', 'model')
});
```

master* 0↓ 1↑ 0 0 Ln 8, Col 41 Spaces: 2 UTF-8 LF JavaScript

post-tile.hbs - statetalk

post-tile.js | post-tile.hbs

```
{{#link-to "posts.show" model.id}}
  <div class="author-image">
    <img src={{model.authorPhotoUrl}} alt="">
  </div>
  <div class="post-tile-content">
      <span class="comment-count">
        {{model.comments.length}}
      </span>
      <h3>{{model.title}}</h3>
    <span class="author-name">
      {{model.authorName}}
    </span>
    <span class="created-at pull-right">
      {{moment-format model.createdAt "LL"}}
    </span>
    <div class="post-tease">
      {{tease model.body chars=80}}...
    </div>
  </div>
{{/link-to}}
```

master* 0↓ 1↑ 0 0 | Ln 1, Col 1 | Spaces: 2 | UTF-8 | LF | Handlebars

Seven Tips From Ernest He… 0

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 2

Mike North October 10, 2016

For over a decade, most of us built web UIs
operating under the assumption that ...

Down the Rabbit Hole 0

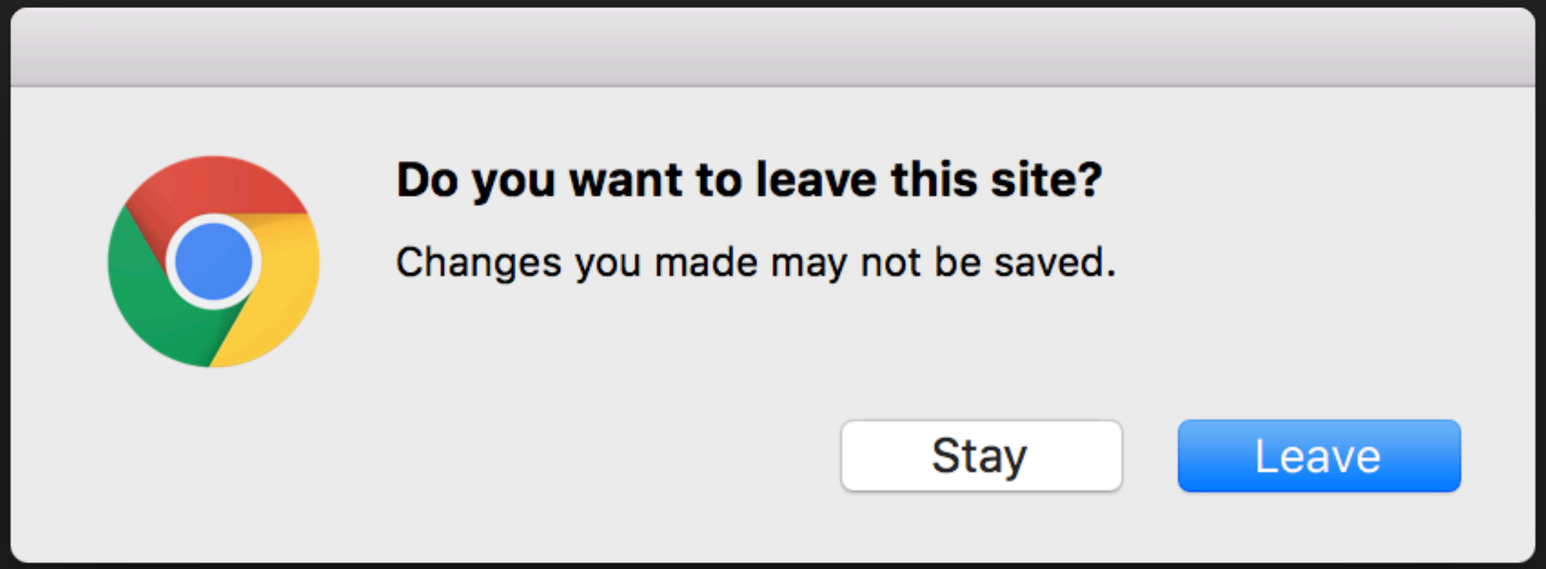
Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by
her sister on the bank, and ...

Please select a post

🏷️2-draft

MAKING DRAFTS EVEN BETTER



ember-state-services/issues/64

🏷️ 2.1-draft-confirm-unload

PERSISTED STATE

Core Concept – Persisted State

STORED IN A LONG-LIVING MEDIUM FOR A LONG TIME

EXAMPLES

- Anything stored in/on a…
  - Database/Disk
  - Local storage
  - Third party API

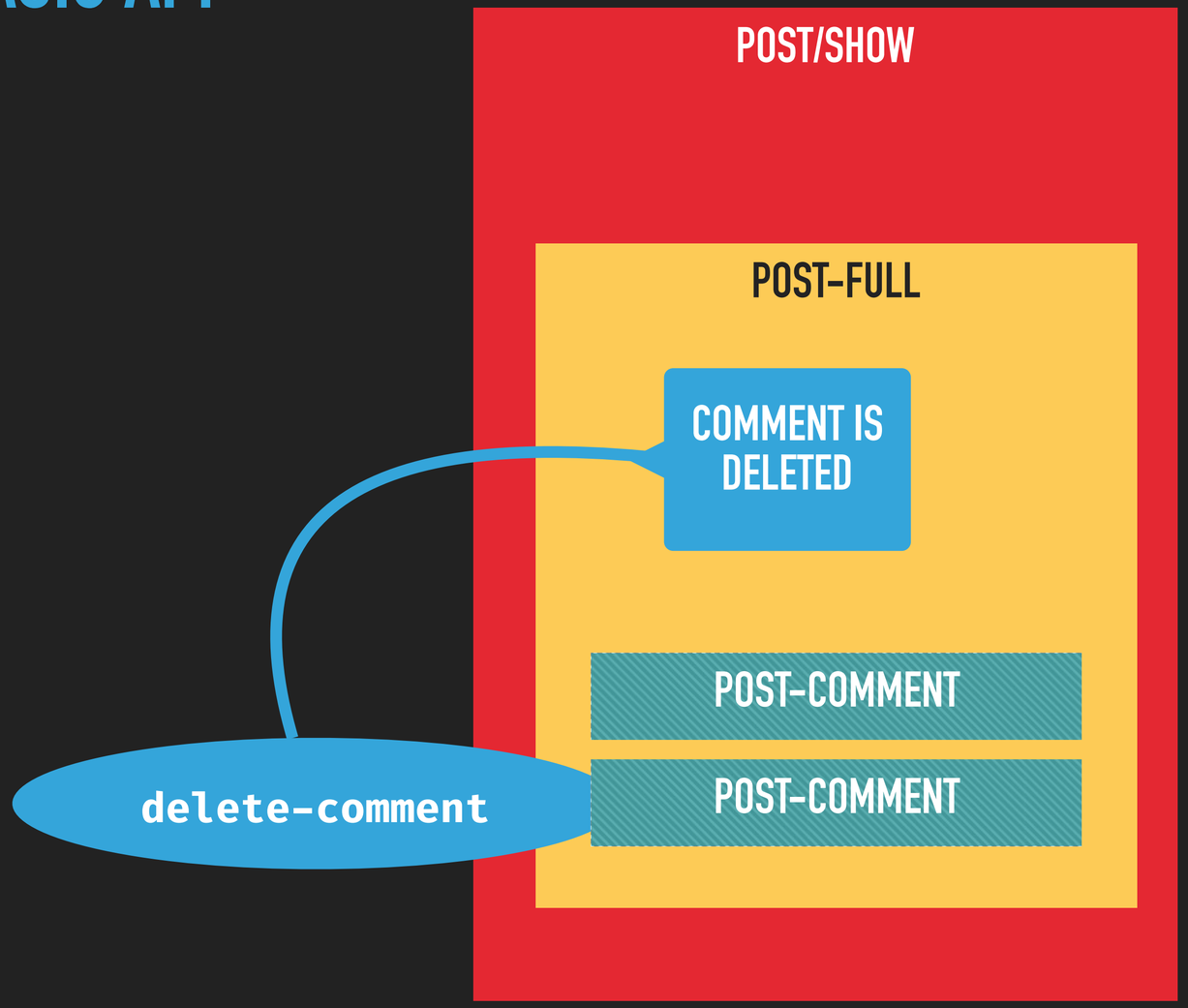
HOW TO IDENTIFY

- Lives longer than a session
- Tracked over time
- Seen by many users

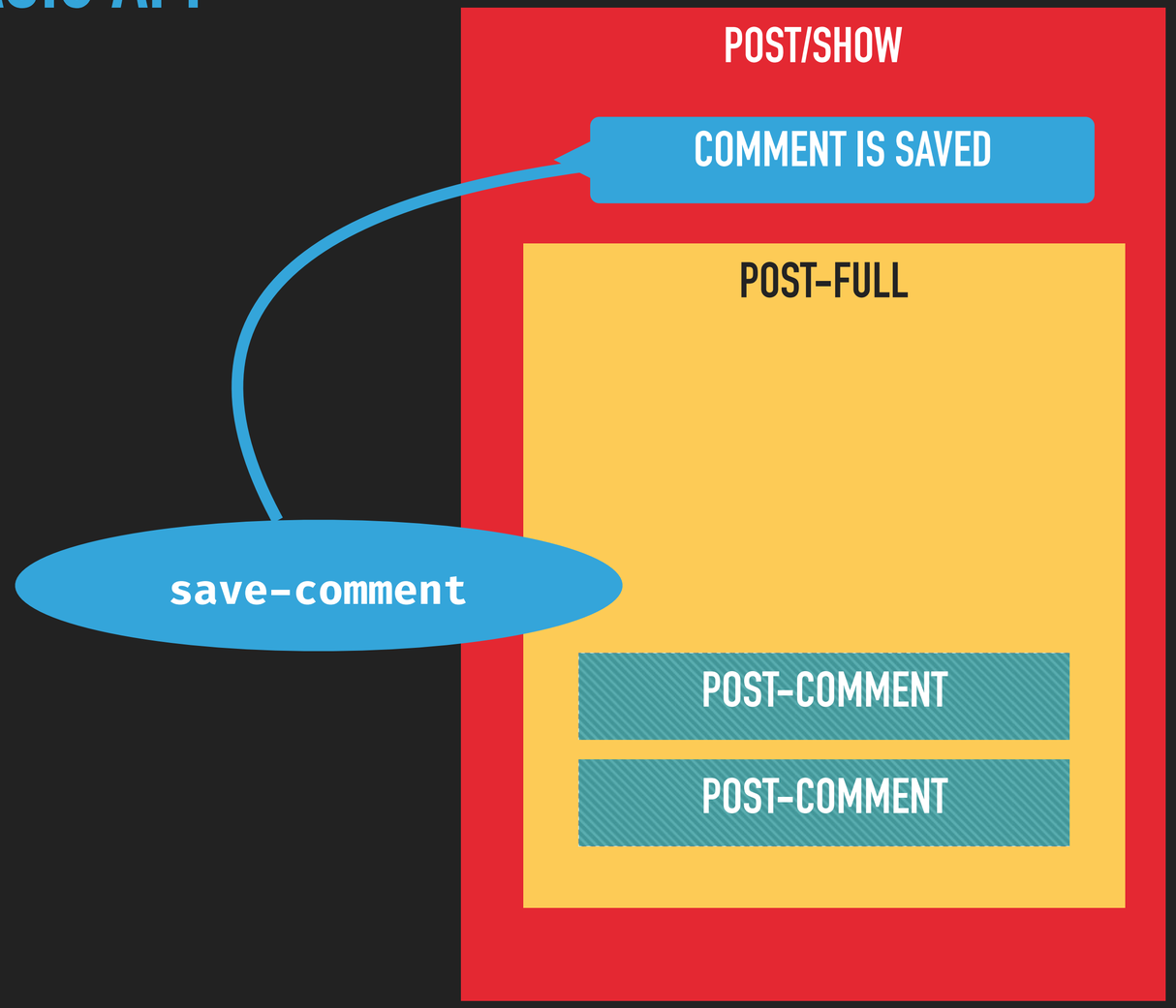
ENHANCING OUR BASIC APP

- POST to API for create
  - Draft cleared if successful
- DELETE to API for delete
- Must work in high-latency

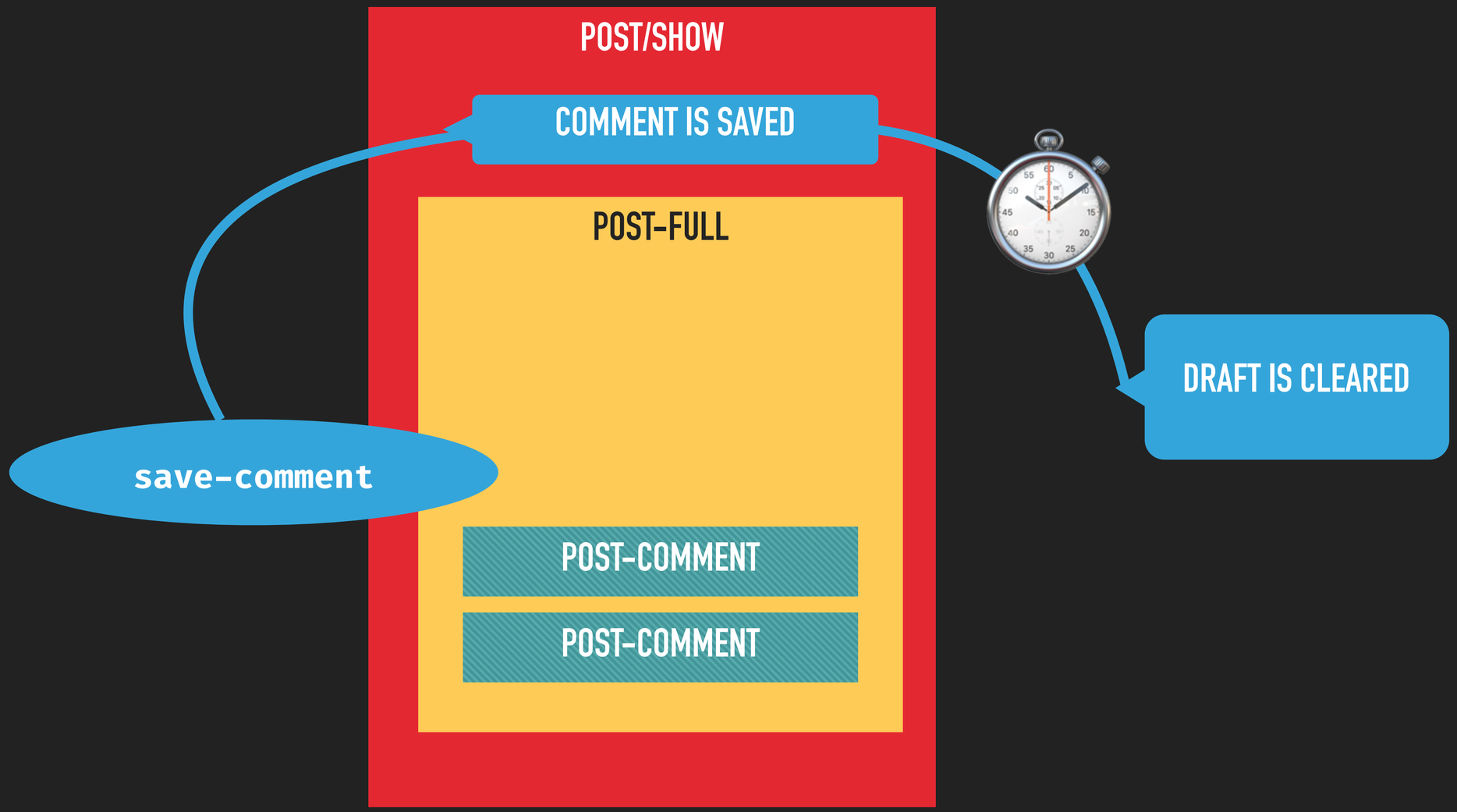
ENHANCING OUR BASIC APP



ENHANCING OUR BASIC APP



ENHANCING OUR BASIC APP



post-full.hbs - statetalk

post-full.hbs

```
35  <p class="body">…
37  </p>
38
39  <hr>
40
41  {{#each model.comments as |comment|}}
42    {{post-comment
43      model=comment
44      delete-comment=(action 'deleteComment' comment)}}
45  {{else}}
49  {{/each}}
50  {{textarea value=postInfo.body class='reply-field'}}
51
52  <div class="post-actions">
53    <button>Reply</button>
54  </div>
55
```

comment
comments

master 0 0 Ln 44, Col 51 Spaces: 2 UTF-8 LF Handlebars

post-comment.hbs - statetalk

post-comment.hbs

```
<p class='comment-body'>
  {{model.body}}
</p>

<span class="delete-indicator"
      onClick={{this.attrs.delete-comment}}>
</span>

{{#if model.updatedAt}}
  <span class="posted-at">
    {{moment-from-now model.updatedAt}}
  </span>
{{/if}}
```

Closure Action

master* 0 0 Ln 6, Col 1 Spaces: 2 UTF-8 LF Handlebars

post-full.hbs - statetalk

post-full.hbs

```
35  <p class="body">…
37  </p>
38
39  <hr>
40
41  {{#each model.comments as |comment|}}
42    {{post-comment
43      model=comment
44      delete-comment=(action 'deleteComment' comment)}}
45  {{else}}…
49  {{/each}}
50  {{textarea value=postInfo.body class='reply-field'}}
51
52  <div class="post-actions">
53    <button
54      onClick={{action 'persistCommentDraft'}}>Reply</button>
55  </div>
56
```

master 0 0 Ln 54, Col 45 Spaces: 2 UTF-8 LF Handlebars

post-full.js - statetalk

post-full.js

```javascript
import Ember from 'ember';
import stateFor from 'ember-state-services/state-for';

const { Component } = Ember;

export default Component.extend({
  classNames: ['post-full'],
  postInfo: stateFor('post-info', 'model'),
  actions: {
    deleteComment(commentRecord) {
      commentRecord.destroyRecord();
    }
  }
});
```

master* 0 1 Ln 11, Col 37 Spaces: 2 UTF-8 LF JavaScript

post-full.js - statetalk

post-full.js

```javascript
import Ember from 'ember';
import stateFor from 'ember-state-services/state-for';

const { Component } = Ember;

export default Component.extend({
  classNames: ['post-full'],
  postInfo: stateFor('post-info', 'model'),
  actions: {
    deleteComment(commentRecord) {
      commentRecord.destroyRecord();
    },
    persistCommentDraft() {
      let commentBody = this.get('postInfo.body');
      let post = this.get('model');
      this.attrs['save-comment'](post, commentBody);
    }
  }
});
```

master* 0 0 Ln 16, Col 53 Spaces: 2 UTF-8 LF JavaScript

show.hbs - statetalk

show.hbs

```
1  {{post-full model=model save-comment=(action 'saveComment')}}
```

master* 0 0 Ln 1, Col 58 Spaces: 4 UTF-8 LF Handlebars

show.js - statetalk

show.js

```js
import Ember from 'ember';

const { Controller } = Ember;

export default Controller.extend({
  actions: {
    saveComment(post, commentBody) {
      this.store.createRecord('comment', {
        post,
        body: commentBody
      }).save().then(() => {
        //delete the draft so we're clean again
      });
    }
  }
});
```

master* 0 0 Ln 13, Col 10 Spaces: 2 UTF-8 LF JavaScript

show.js - statetalk

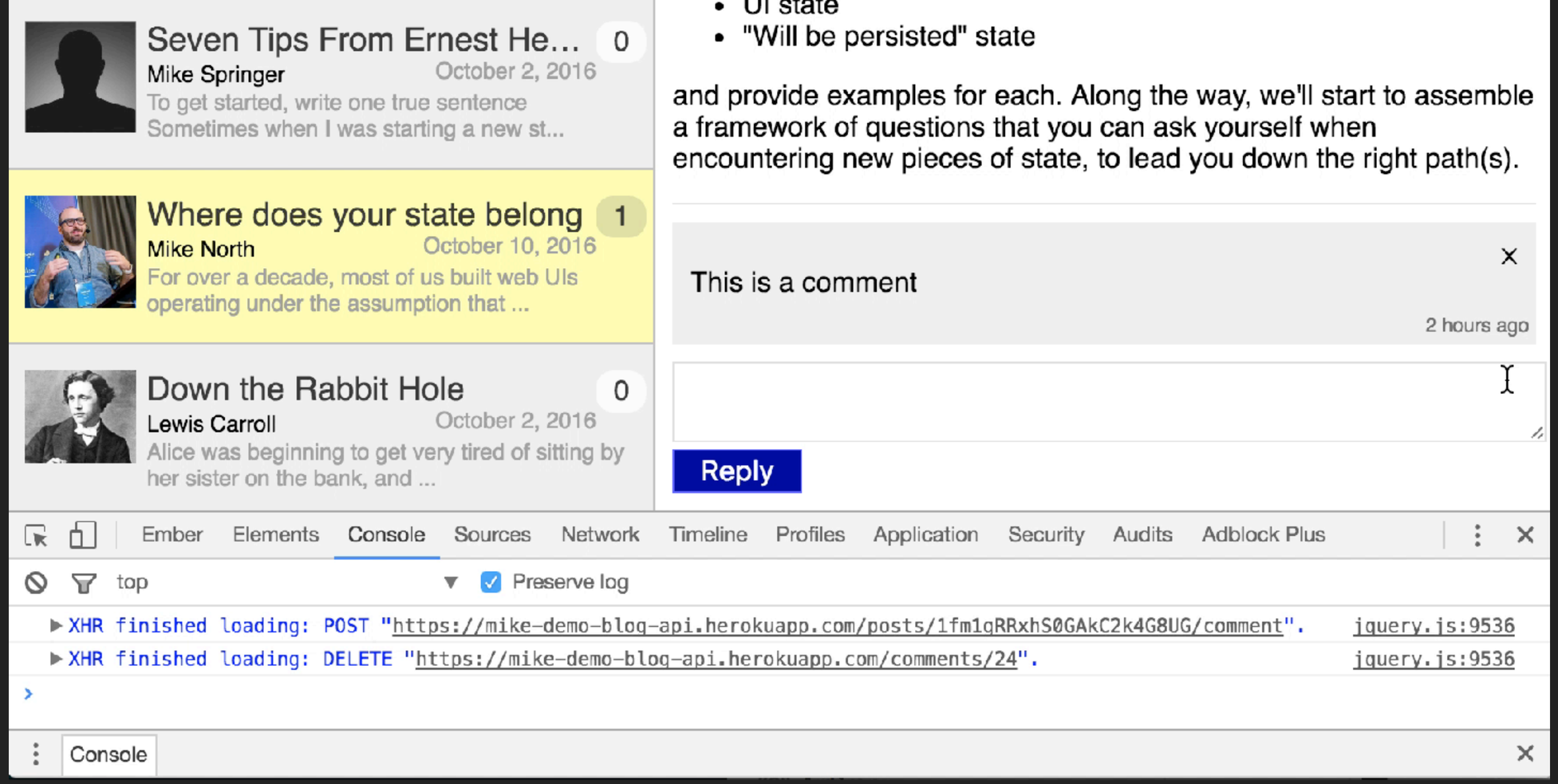
show.js

```javascript
import Ember from 'ember';
import stateFor from 'ember-state-services/state-for';

const { Controller } = Ember;

export default Controller.extend({
  actions: {
    saveComment(post, commentBody) {
      this.store.createRecord('comment', {
        post,
        body: commentBody
      }).save().then(() => {
        //delete the draft so we're clean again
        stateFor('post-info') // WeakMap for all post-infos
          .get(post) // post-info for the post we commented on
          .set('body', ''); // set the draft comment back to ''
      });
    }
  }
});
```

master* 0 0 Ln 16, Col 64 Spaces: 2 UTF-8 LF JavaScript


Seven Tips From Ernest He... 0
Mike Springer October 2, 2016
To get started, write one true sentence
Sometimes when I was starting a new st...
Where does your state belong 1
Mike North October 10, 2016
For over a decade, most of us built web UIs operating under the assumption that ...
Down the Rabbit Hole 0
Lewis Carroll October 2, 2016
Alice was beginning to get very tired of sitting by her sister on the bank, and ...
UI state
"Will be persisted" state
and provide examples for each. Along the way, we'll start to assemble a framework of questions that you can ask yourself when encountering new pieces of state, to lead you down the right path(s).
This is a comment
2 hours ago
Reply
Ember Elements Console Sources Network Timeline Profiles Application Security Audits Adblock Plus
top
Preserve log
XHR finished loading: POST "https://mike-demo-blog-api.herokuapp.com/posts/1fm1qRRxhS0GAkC2k4G8UG/comment". jquery.js:9536
XHR finished loading: DELETE "https://mike-demo-blog-api.herokuapp.com/comments/24". jquery.js:9536
Console

🏷️3-persisted-state

A COMMON MISTAKE IN "AFTER SAVE" LOGIC

show.js - statetalk

show.js

```
1   import Ember from 'ember';
2   import stateFor from 'ember-state-services/state-for';
3
4   const { Controller } = Ember;
5
6   export default Controller.extend({
7     actions: {
8       saveComment(post, commentBody) {
9         this.store.createRecord('comment', {
10          post,
11          body: commentBody
12        }).save().then(() => {
13          // delete the draft so we're clean again
14          stateFor('post-info') // WeakMap for all post-infos
15            .get(/*post*/ this.get('model')) // post-info for the post we
16            .set('body', ''); // set the draft comment back to ''
17        });
18      }
19    }
20  });
21
```

master 0↓ 1↑ 0 0 Ln 18, Col 6 Spaces: 2 UTF-8 LF JavaScript

DON'T DO THIS

DON'T DO THIS

Statetalk

localhost:4200/posts/A96usFSlY4G0W4kwAqswk

Search

Seven Tips From Ernest He… 0
Mike Springer DRAFT October 2, 2016
To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1
Mike North DRAFT October 10, 2016
For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0
Lewis Carroll October 2, 2016
Alice was beginning to get very tired of sitting by her sister on the bank, and ...

and you enjoy it that much more. After you learn to write
object is to convey everything, every sensation, sight, feeli
emotion to the reader. To do this you have to work over what

If you write with a pencil you get three different sights at it to see i
reader is getting what you want him to. First when you read it over; t
when it is typed you get another chance to improve it, and again in the
proof. Writing it first in pencil gives you one-third more chance to improve it. That is .333 which is a damned good average for a hitter. It also keeps it fluid longer so you can better it easier.

Be brief

It wasn't by accident that the Gettysburg address was so short. The laws of prose writing are as immutable as those of flight, of mathematics, of physics.

The original, unabbreviated version of the blog post can be found on the Open Culture website.

No comments yet

Draft for first post

Reply

Ember Elements Console Sources Network Timeline Profiles Application Security Audits Adblock Plus

View: Preserve log Disable cache Offline Really high latency (30

Filter Regex Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

DON'T DO THIS

Statetalk

localhost:4200/posts/A96usFSlY4G0W4kwAqswk

Search

Seven Tips From Ernest He… 0
Mike Springer DRAFT October 2, 2016
To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1
Mike North DRAFT October 10, 2016
For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0
Lewis Carroll October 2, 2016
Alice was beginning to get very tired of sitting by her sister on the bank, and ...

and you enjoy it that much more. After you learn to write
object is to convey everything, every sensation, sight, feeli
emotion to the reader. To do this you have to work over what

If you write with a pencil you get three different sights at it to see i
reader is getting what you want him to. First when you read it over; t
when it is typed you get another chance to improve it, and again in the
proof. Writing it first in pencil gives you one-third more chance to improve it. That is .333 which is a damned good average for a hitter. It also keeps it fluid longer so you can better it easier.

Be brief

It wasn't by accident that the Gettysburg address was so short. The laws of prose writing are as immutable as those of flight, of mathematics, of physics.

The original, unabbreviated version of the blog post can be found on the Open Culture website.

No comments yet

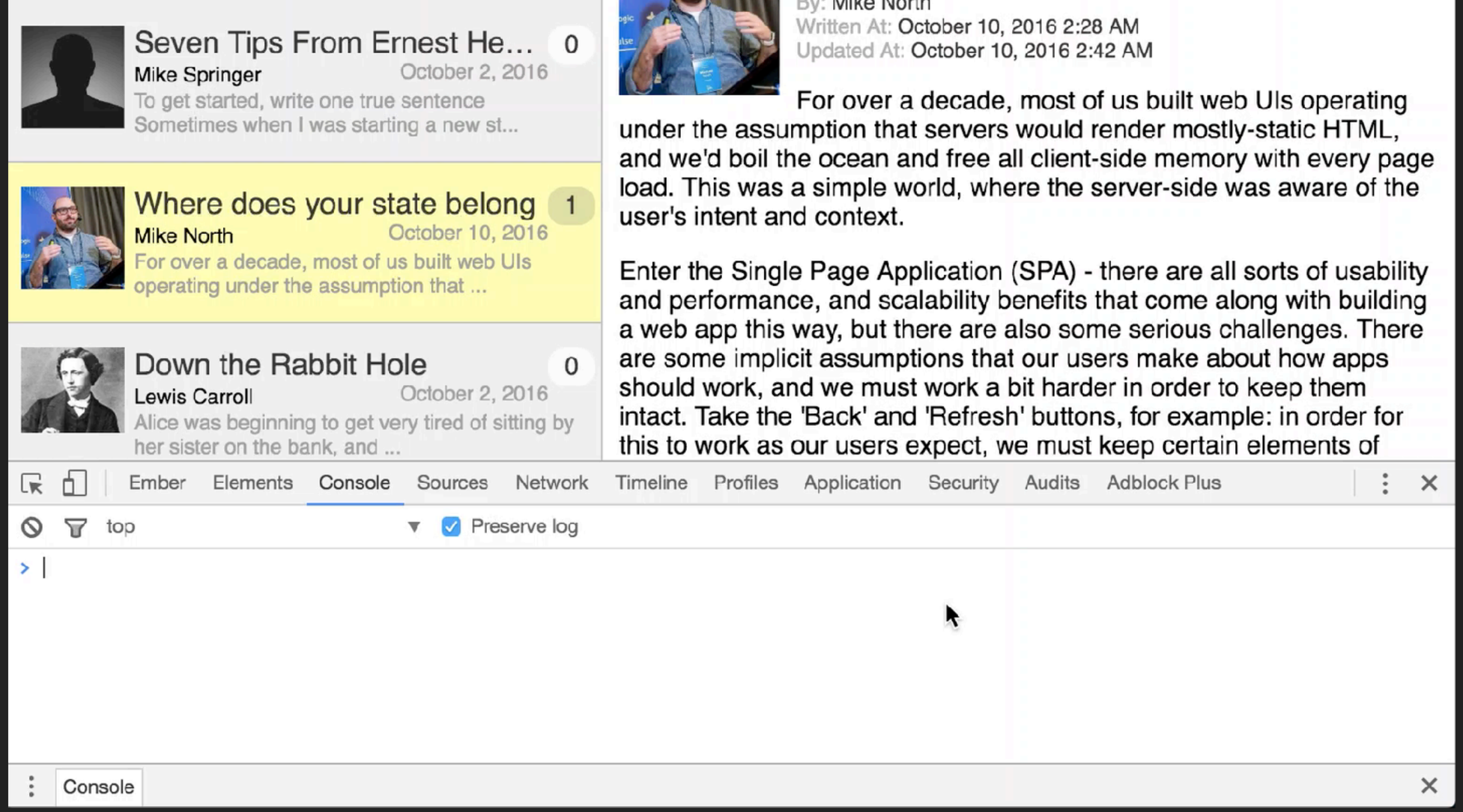
Draft for first post

Reply

Ember Elements Console Sources Network Timeline Profiles Application Security Audits Adblock Plus

View: Preserve log Disable cache Offline Really high latency (30

Filter Regex Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other


Seven Tips From Ernest He… 0
Mike Springer October 2, 2016
To get started, write one true sentence
Sometimes when I was starting a new st...
Where does your state belong 1
Mike North October 10, 2016
For over a decade, most of us built web UIs operating under the assumption that ...
Down the Rabbit Hole 0
Lewis Carroll October 2, 2016
Alice was beginning to get very tired of sitting by her sister on the bank, and ...
By: Mike North
Written At: October 10, 2016 2:28 AM
Updated At: October 10, 2016 2:42 AM
For over a decade, most of us built web UIs operating under the assumption that servers would render mostly-static HTML, and we'd boil the ocean and free all client-side memory with every page load. This was a simple world, where the server-side was aware of the user's intent and context.
Enter the Single Page Application (SPA) - there are all sorts of usability and performance, and scalability benefits that come along with building a web app this way, but there are also some serious challenges. There are some implicit assumptions that our users make about how apps should work, and we must work a bit harder in order to keep them intact. Take the 'Back' and 'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of
Ember
Elements
Console
Sources
Network
Timeline
Profiles
Application
Security
Audits
Adblock Plus
top
Preserve log
Console

PERSISTED STATE – BEST PRACTICES

- Consider that things may change during async
- Build simple objects and **then** createRecord
- Consider in-flight records

PERS

- Do
- Do

Be brief

It wasn't by accident that the Gettysburg address was so short. The laws of prose writing are as immutable as those of flight, of mathematics, of physics.

The original, unabbreviated version of the blog post can be found on the Open Culture website.

No comments yet

Reply

del.js:502

ttempted
39:35>

UI STATE

Core Concept – UI State

SHORT-LIVED, USEFUL ONLY "IN THE MOMENT", AND RELEVANT ONLY TO THE UI

EXAMPLES

- Accordion expansion
- Scroll position
- Text selection & Focus
- Matthew Beale is typing…

HOW TO IDENTIFY

- Short lived, only meaningful in the moment
- Easiest to re-create
- Potentially harmful if persisted/restored

post-full.hbs - statetalk

post-full.js | post-full.hbs

```
1   <div class="title-row">
2     <img src={{model.authorPhotoUrl}} alt={{authorName}} class="author-po
3     <h1 class='post-title'>{{titlecase model.title}}</h1>
4
5     {{#if detailsOpen}}
6     <div onClick={{action (mut detailsOpen) false}} class="metadata-row">
27    </div>
28    {{else}}
29      <span onClick={{action (mut detailsOpen) true}}>
30        details...
31      </span>
32    {{/if}}
33
34  </div>
35
36
37  {{#if model.featuredImageUrl}}
38  <div class="feature-image-row">
39    <img src={{model.featuredImageUrl}} alt="" class="feature-image">
40  </div>
41  {{/if}}
```

master 0↓ 1↑ 0 0 Ln 6, Col 48 Spaces: 2 UTF-8 LF Handlebars

post-full.js - statetalk

post-full.js | post-full.hbs

```js
import Ember from 'ember';
import stateFor from 'ember-state-services/state-for';

const { Component } = Ember;

export default Component.extend({
  classNames: ['post-full'],
  postInfo: stateFor('post-info', 'model'),
  detailsOpen: false,
  actions: {
    deleteComment(commentRecord) {
      commentRecord.destroyRecord();
    },
    persistCommentDraft() {
      let commentBody = this.get('postInfo.body');
      let post = this.get('model');
      this.attrs['save-comment'](post, commentBody);
    }
  }
});

```

master* 0↓ 1↑ 0 0 Ln 9, Col 22 Spaces: 2 UTF-8 LF JavaScript

localhost:4200/posts

Search

Seven Tips From Ernest He… 3

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1

Mike North October 10, 2016

For over a decade, most of us built web UIs
operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by
her sister on the bank, and ...

Please select a post

KEEP COMPONENT LIFE-CYCLE IN MIND

post-full.js - statetalk

post-full.js | post-full.hbs

```javascript
import Ember from 'ember';
import stateFor from 'ember-state-services/state-for';

const { Component } = Ember;

export default Component.extend({
  classNames: ['post-full'],
  postInfo: stateFor('post-info', 'model'),
  detailsOpen: false,
  actions: {
    deleteComment(commentRecord) {
      commentRecord.destroyRecord();
    },
    persistCommentDraft() {
      let commentBody = this.get('postInfo.body');
      let post = this.get('model');
      this.attrs['save-comment'](post, commentBody);
    }
  }
});
```

master* 0↓ 1↑ 0 0 Ln 9, Col 22 Spaces: 2 UTF-8 LF JavaScript

post-info.js - statetalk

post-info.js

```
1   import Ember from 'ember';
2
3   // jscs:disable disallowDirectPropertyAccess
4   const PostInfoState = Ember.Object.extend({});
5   // jscs:enable disallowDirectPropertyAccess
6
7   PostInfoState.reopenClass({
8     initializeState(/*instance*/) {
9       return {
10        body: '',
11        detailsOpen: false
12      };
13    }
14  });
15
16  export default PostInfoState;
```

master 0↓ 1↑ 0 0 Ln 11, Col 25 Spaces: 2 UTF-8 LF JavaScript

post-full.hbs - statetalk

post-full.js | post-full.hbs

```
1   <div class="title-row">
2     <img src={{model.authorPhotoUrl}} alt={{authorName}} class="author-po
3     <h1 class='post-title'>{{titlecase model.title}}</h1>
4
5     {{#if detailsOpen}}
6     <div onClick={{action (mut detailsOpen) false}} class="metadata-row">
27    </div>
28    {{else}}
29      <span onClick={{action (mut detailsOpen) true}}>
30        details...
31      </span>
32    {{/if}}
33
34  </div>
35
36
37  {{#if model.featuredImageUrl}}
38  <div class="feature-image-row">
39    <img src={{model.featuredImageUrl}} alt="" class="feature-image">
40  </div>
41  {{/if}}
```

master* 0↓ 1↑ 0 0 Ln 33, Col 1 Spaces: 2 UTF-8 LF Handlebars

localhost:4200/posts/1fm1gRRxhS0GAkC2k4G8UG

Search

Seven Tips From Ernest He... 3

Mike Springer October 2, 2016

To get started, write one true sentence
Sometimes when I was starting a new st...

Where does your state belong 1

Mike North October 10, 2016

For over a decade, most of us built web UIs operating under the assumption that ...

Down the Rabbit Hole 0

Lewis Carroll October 2, 2016

Alice was beginning to get very tired of sitting by her sister on the bank, and ...

Where does your state belong

details...

For over a decade, most of us built web UIs operating under the assumption that servers would render mostly-static HTML, and we'd boil the ocean and free all client-side memory with every page load. This was a simple world, where the server-side was aware of the user's intent and context.

Enter the Single Page Application (SPA) - there are all sorts of usability and performance, and scalability benefits that come along with building a web app this way, but there are also some serious challenges. There are some implicit assumptions that our users make about how apps should work, and we must work a bit harder in order to keep them intact. Take the 'Back' and 'Refresh' buttons, for example: in order for this to work as our users expect, we must keep certain elements of state serialized in the URL in order to avoid breaking this as we simulate a multi-page experience in a SPA.

Add in the concept of server-side rendering, where our asset serving layer sometimes needs browser details (i.e. viewport dimensions) in order to render the correct content, and state

🏷️4-ui-state

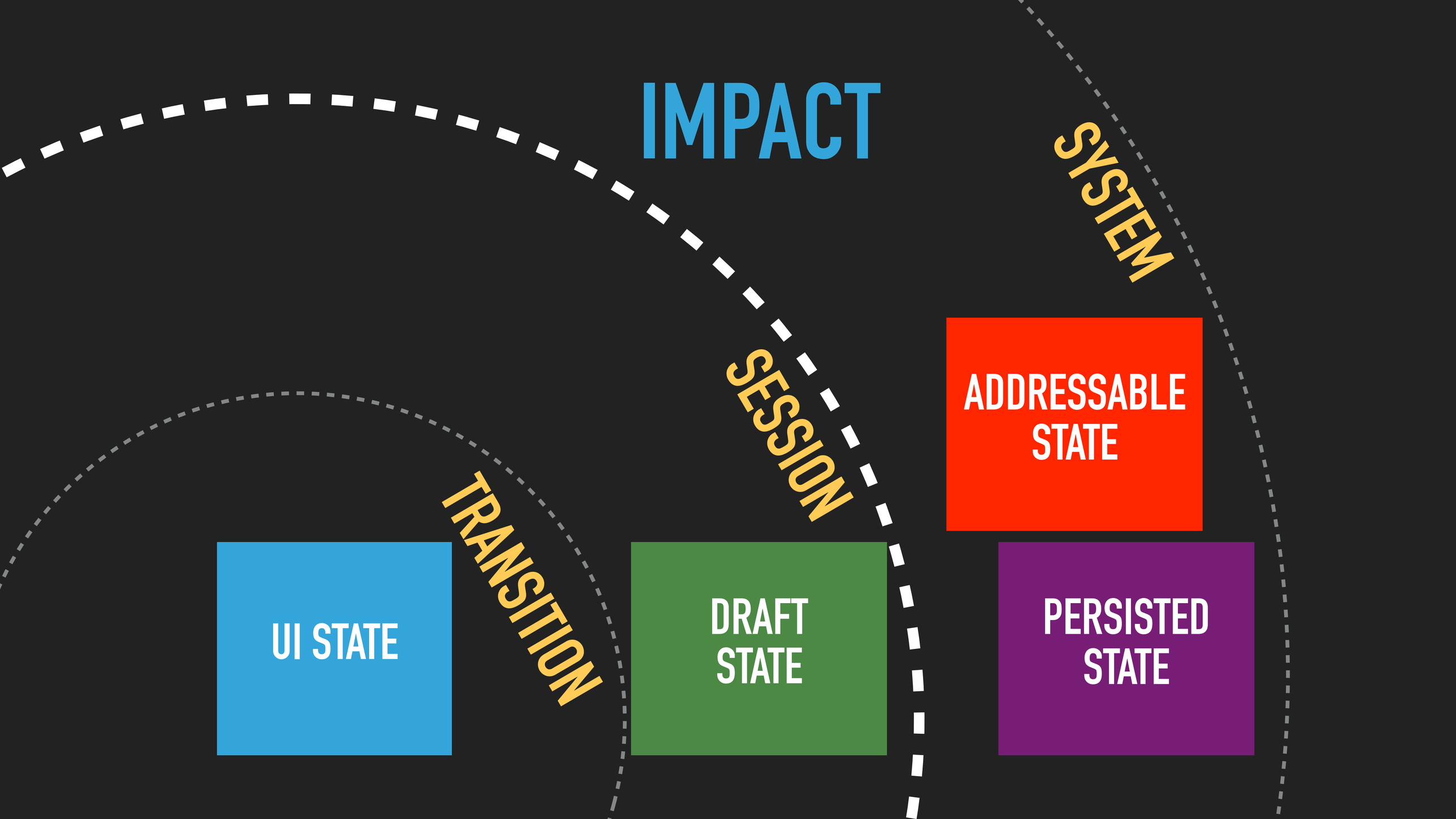
4 FLAVORS OF STATE

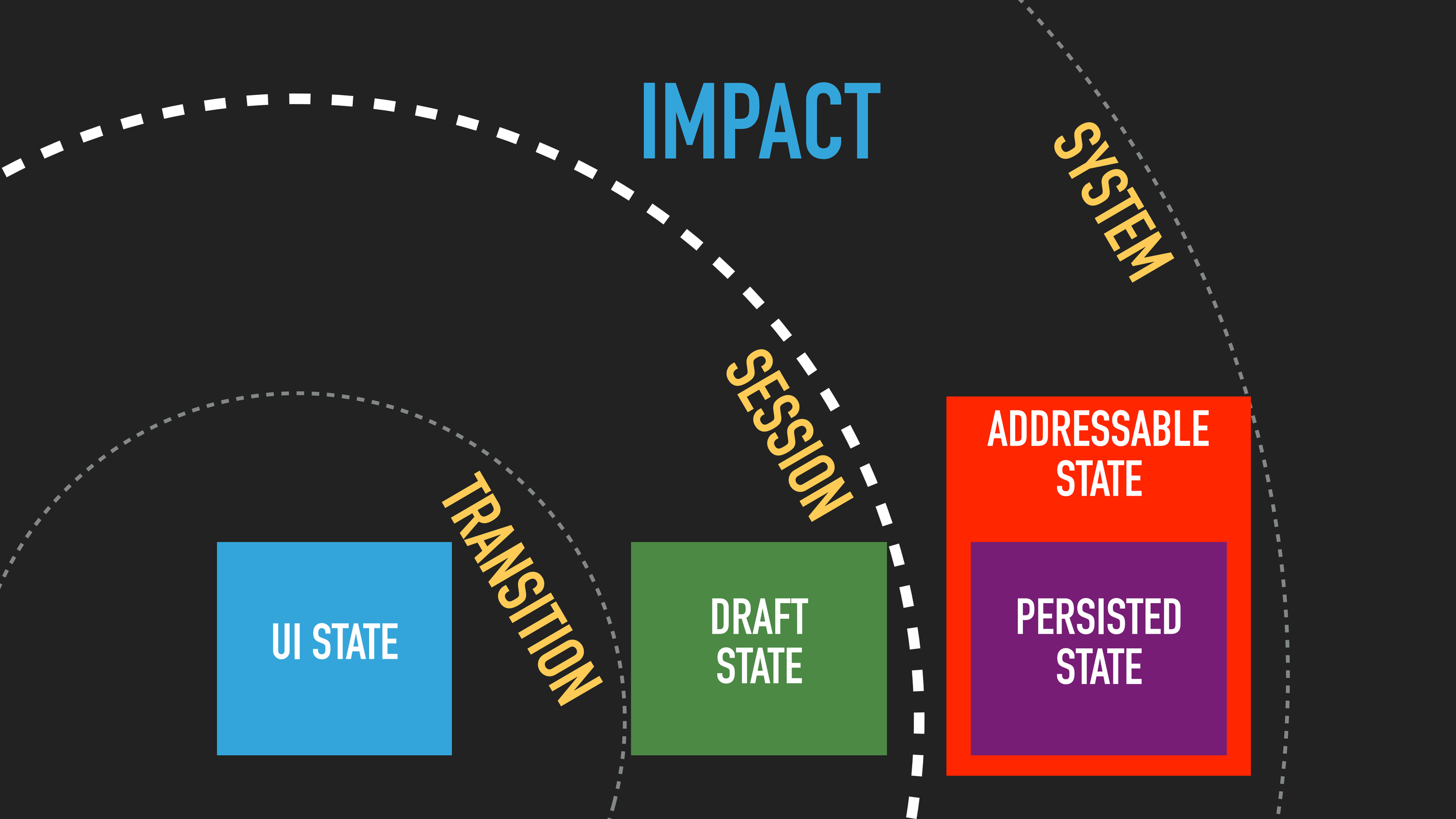
UI STATE

PERSISTED STATE

ADDRESSABLE STATE

DRAFT STATE


IMPACT
SYSTEM
SESSION
TRANSITION
ADDRESSABLE STATE
UI STATE
DRAFT STATE
PERSISTED STATE


IMPACT
SYSTEM
SESSION
TRANSITION
UI STATE
DRAFT STATE
ADDRESSABLE STATE
PERSISTED STATE

WHO CARES?

USER

WORLD

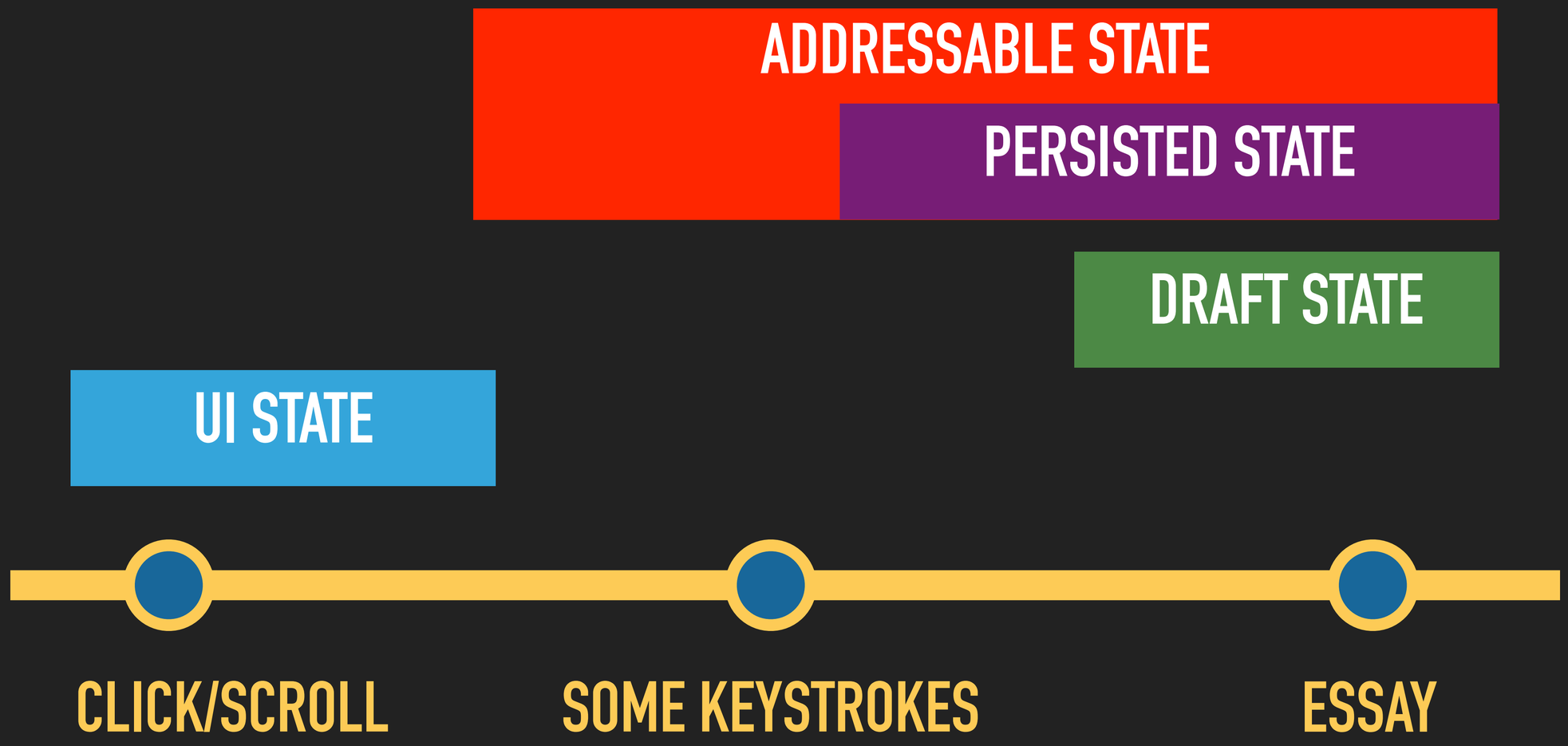
UI STATE

DRAFT STATE

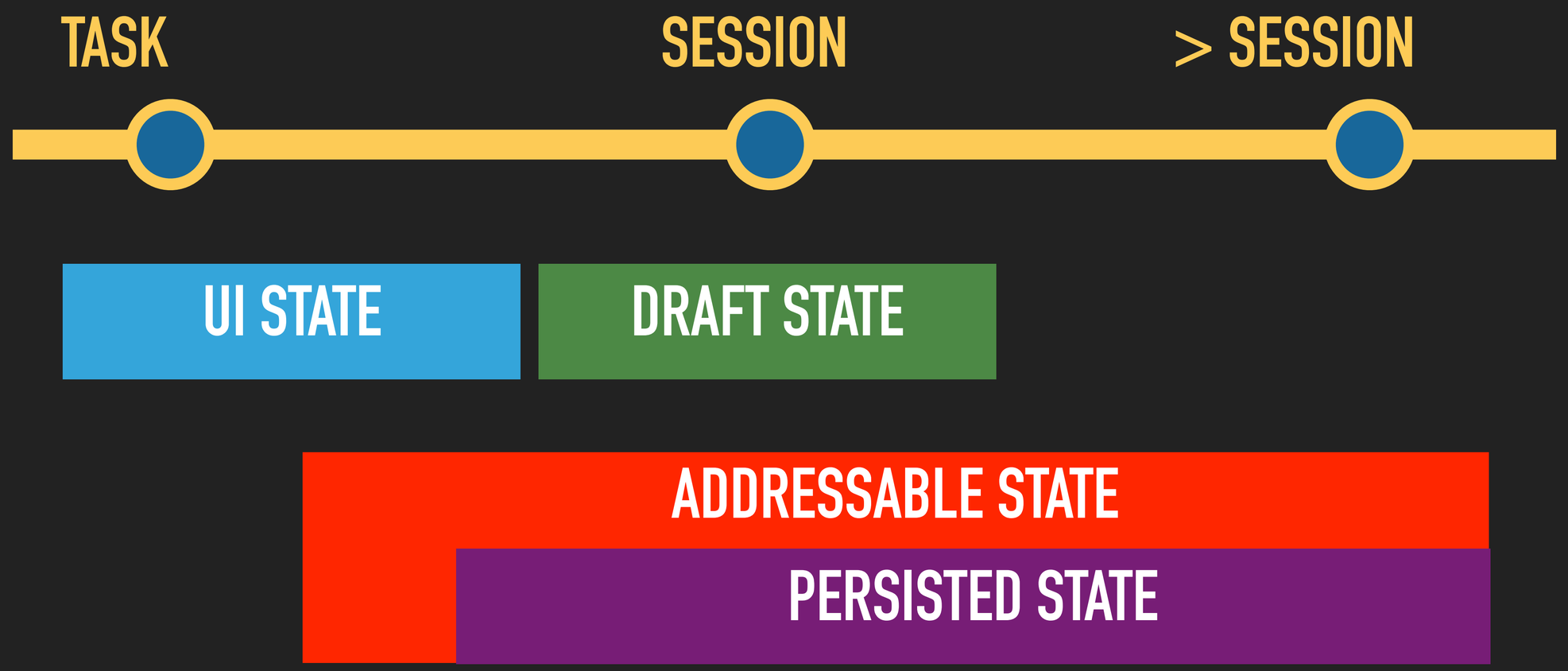
ADDRESSABLE STATE

PERSISTED STATE

USER EFFORT


ADDRESSABLE STATE
PERSISTED STATE
DRAFT STATE
UI STATE
CLICK/SCROLL
SOME KEYSTROKES
ESSAY

LIFETIME


TASK
SESSION
> SESSION
UI STATE
DRAFT STATE
ADDRESSABLE STATE
PERSISTED STATE

http://go/training-feedback

© Mike.Works, Inc. 2017. All rights reserved