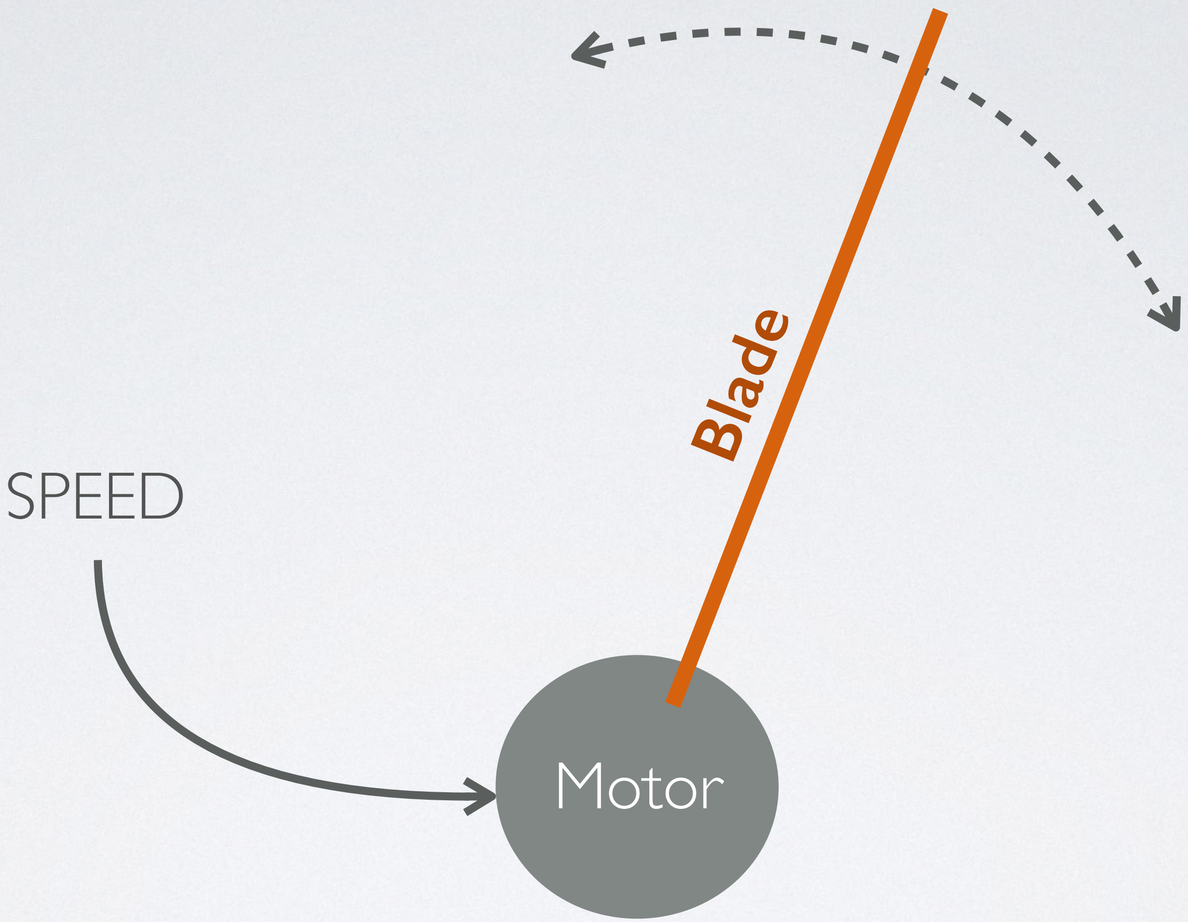
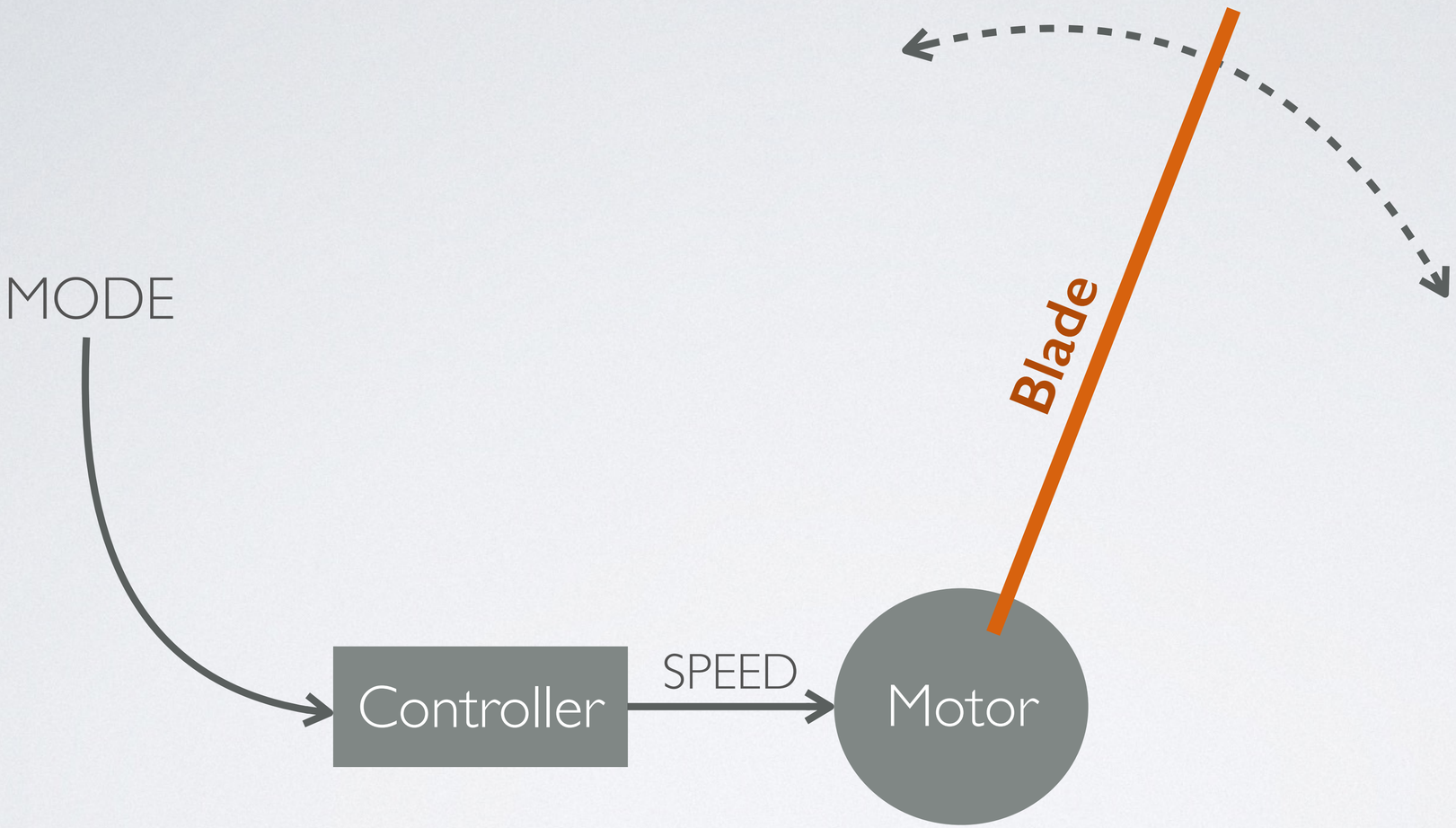
Building Stateful UI

Stefan Penner & Mike North

```
npm install -g ember-cli
git clone https://github.com/mike-north/emberconf-stateful-ui
cd emberconf-stateful-ui; npm install; bower install
```

AU
VC


Blade
SPEED
Motor


MODE
Controller
SPEED
Motor
Blade

Things don't need to be complicated for state management to matter

What to expect

- Buckle up and 🚀
- ~Equal parts instruction & pair programming
- No toy examples
- Insight into how to deal with <u>your</u> app's state

ember-cli

ember-cli-shims

Guard against "self"

In FastBoot or other node envs, self is not a thing. Closes #43

app-shims.js

```
@@ -242,8 +242,12 @@
     });
   }

+  if (self) {
+    jQuery = self.jQuery;
+  }
+
   processEmberShims();
   processTestShims();
-  generateModule('jquery', { 'default': self.jQuery });
+  generateModule('jquery', { 'default': jQuery });
```

I think if the sandbox defines the jQuery object, we don't need to safeguard against this.

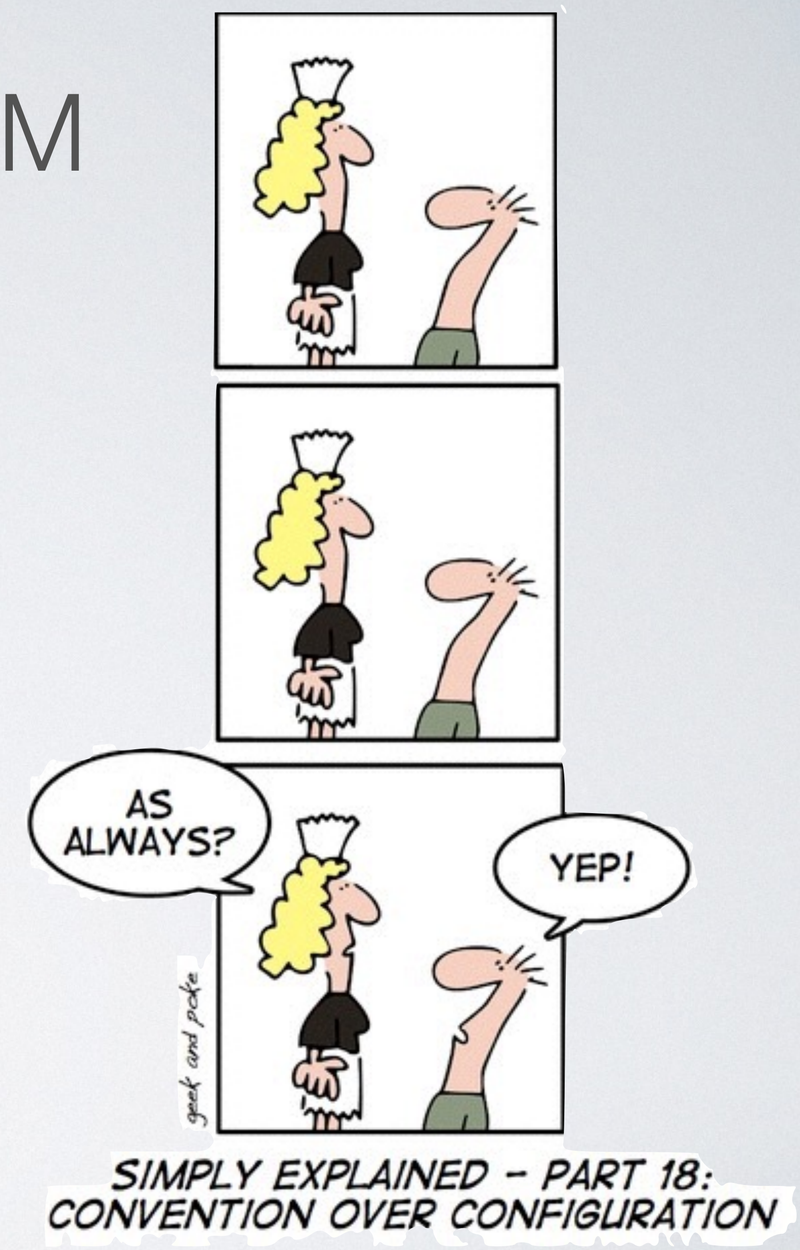
app-shims.js

What to expect

- A supersonic ember primer
- Component state
- Navigational state
- Persisted state
- Will-be-persisted state

Ember Ecosystem

- Framework, tools, add ons, conventions
- Optimizes for developer productivity
  - Reduces # of decisions you have to make



Ember Ecosystem

The Libraries

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

Ember Ecosystem

The 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

Ember (the framework)

Templates

Components

Routing

Services

Templating

- HTML++
- Template compilation

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

```
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);
        return morphs;
      },
      statements: [
        ["content","firstName",["loc",[null,[1,15],[1,28]]]],
        ["content","lastName",["loc",[null,[1,29],[1,41]]]]
      ],
    ...
    };
  }()));
});
```

Templating

```
// Handlebars
<button {{bind-attr "disabled" disabled}} >
// HTMLbars
<button disabled={{disabled}} >


// Classic actions
<button {{action 'myAction' on='click'}}>
// Closure actions!
<button onclick={{action myAction}}>
```

Templating

- Helpers increase vocabulary
- Come with Handlebars, Ember, addons, and possibly your app!

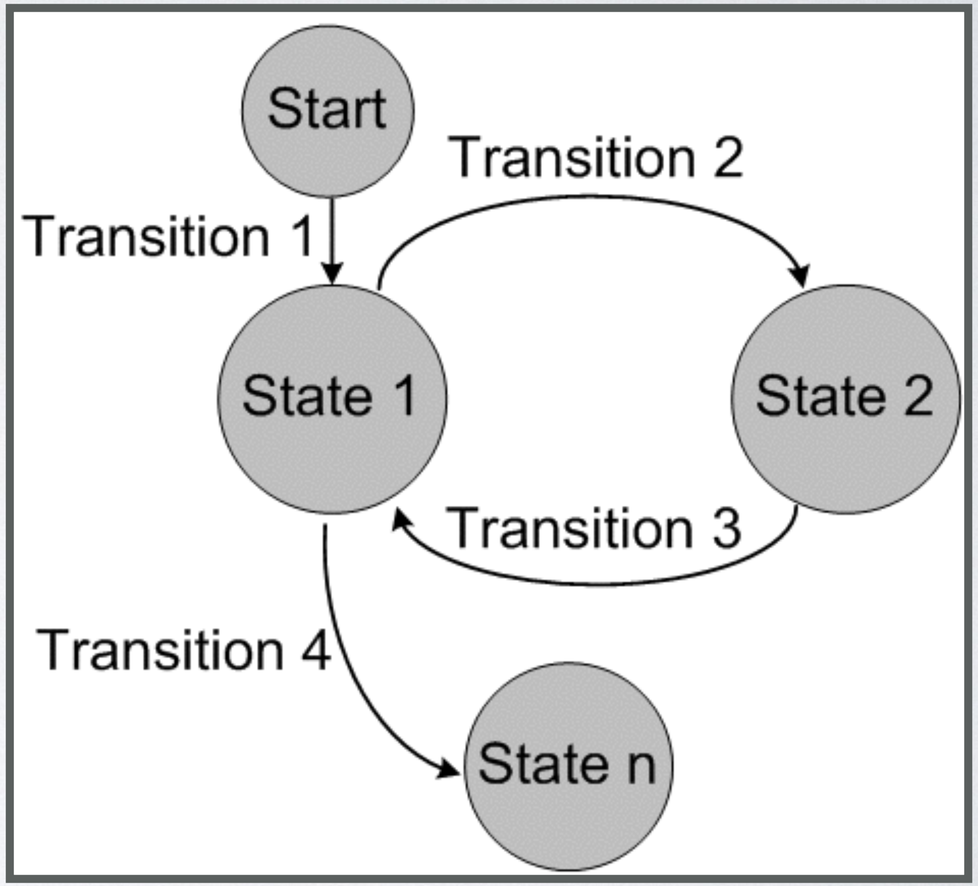
```
My pet goes
{{#if isDog}}
 arf
{{else}}
 meow
{{/if}}
```

That's great, but…

My app has multiple pages, how do I do that?

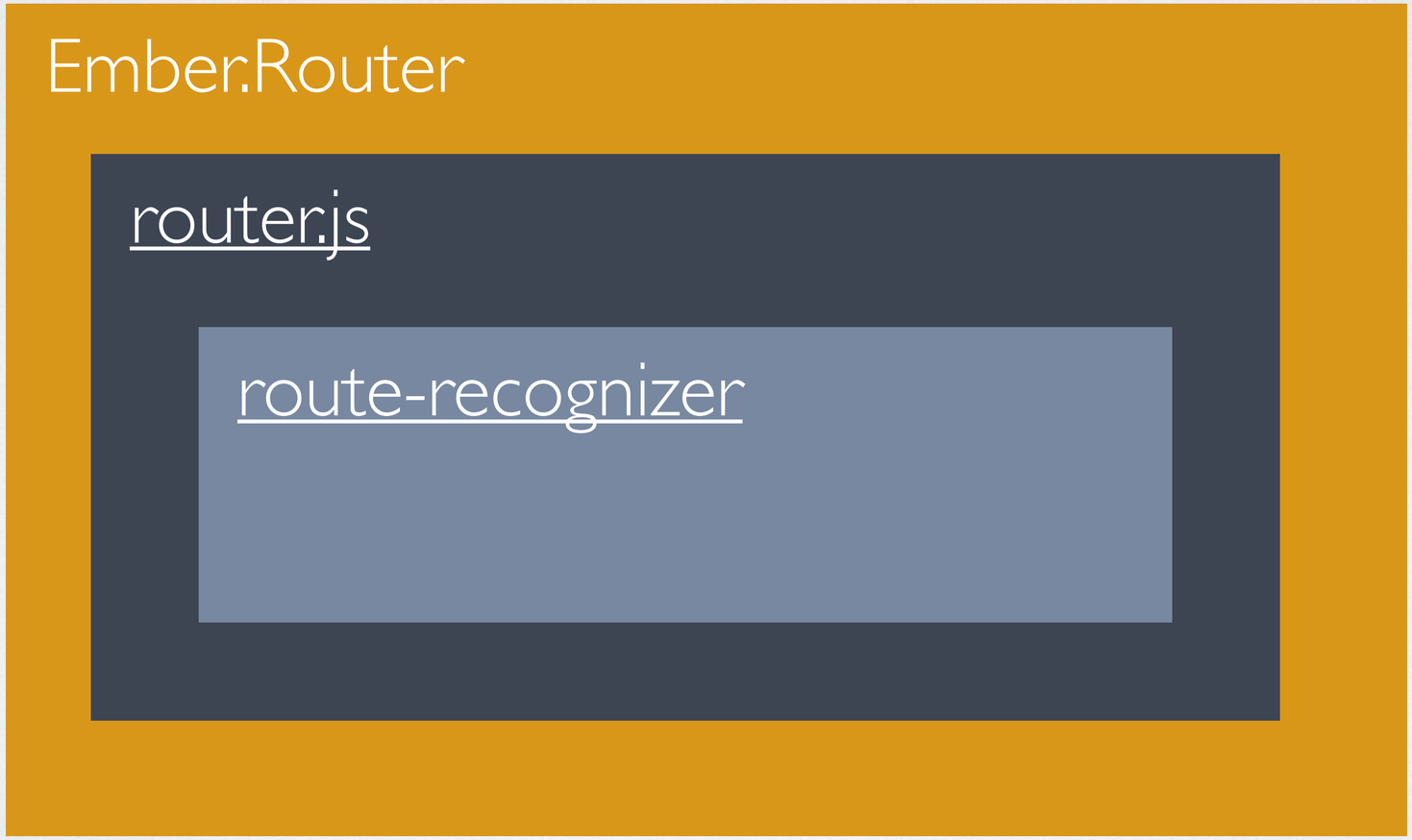
Routing

- Manages transitions between URLs
- Usually leave it as-is, except for Router.map
- A finite state machine



Routing

Foundations

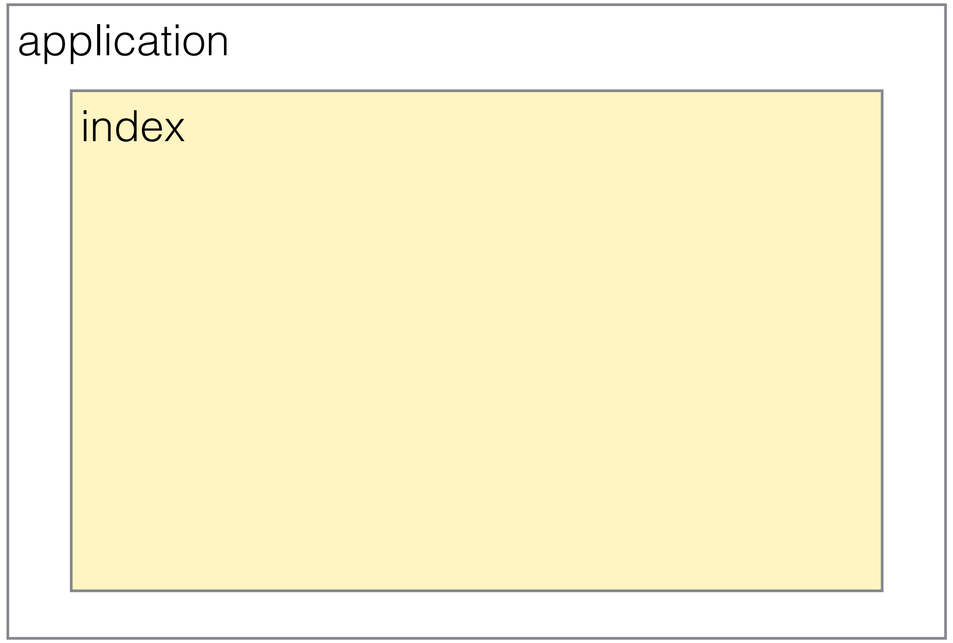


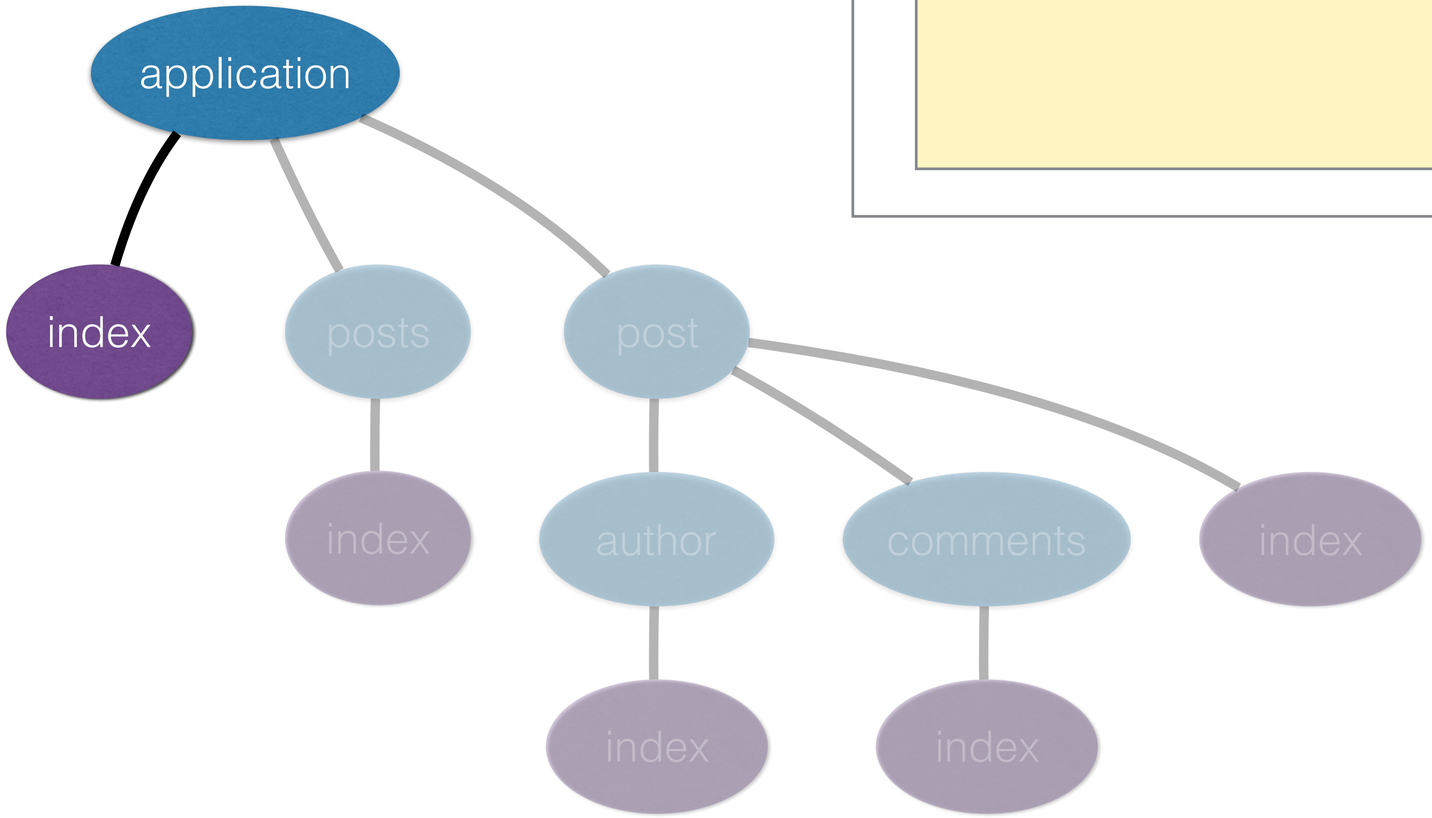
Routing

Hierarchical Organization

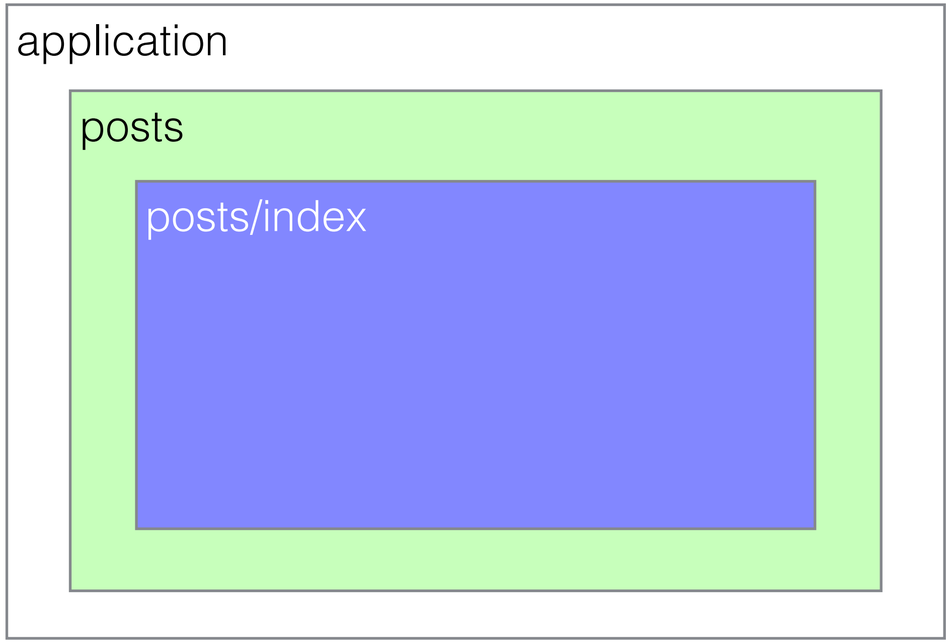
```
Router.map(function() {
  // IMPLIED index                /
 this.route(‘posts'); //          /posts
  this.route('post', {path: 'post/:id'}, function() {
    // IMPLIED index              /post/123
    this.route('author'); //  /post/123/author
    this.route('comments');// /post/123/comments
 });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

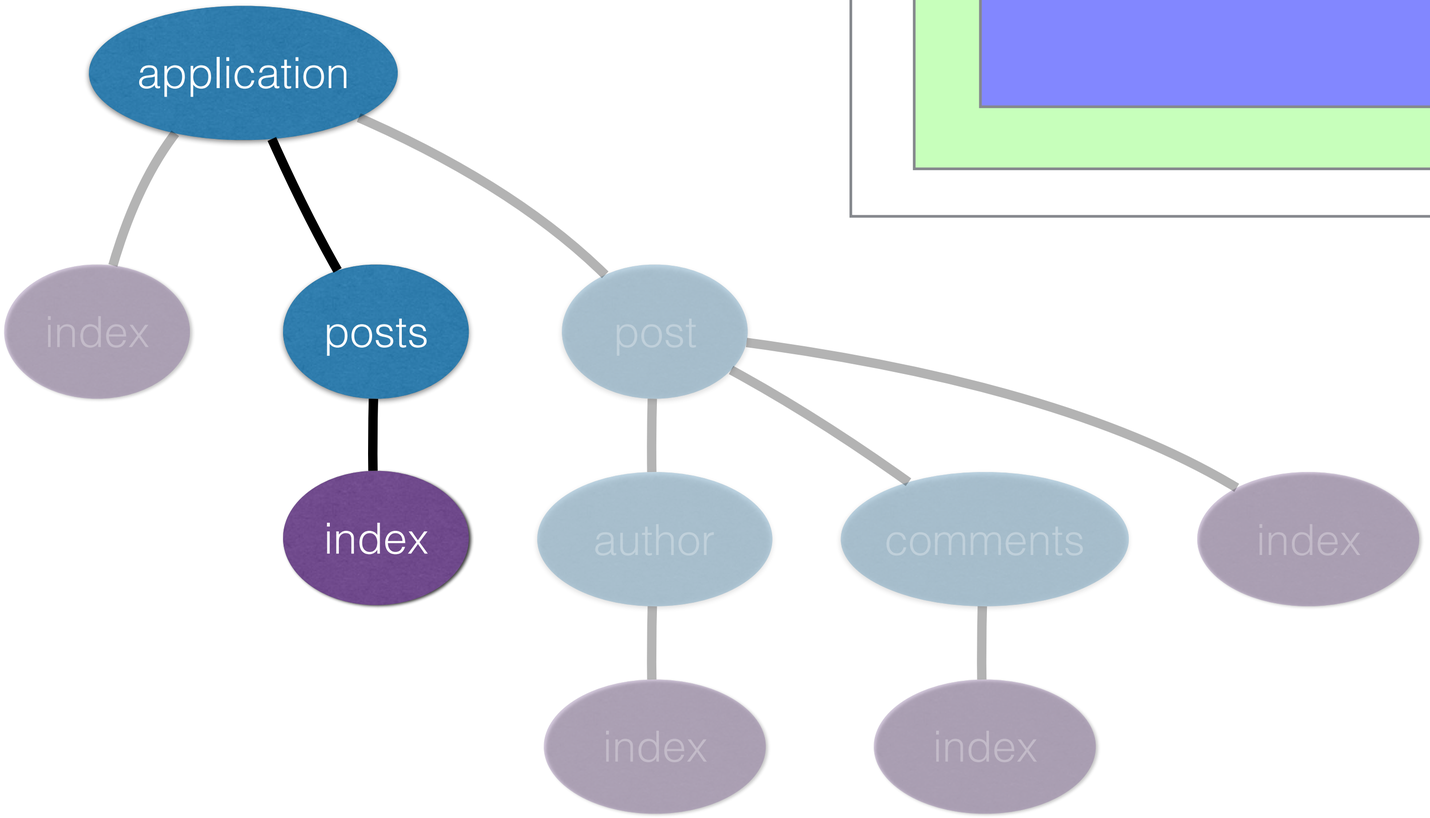
URL: /


application
index

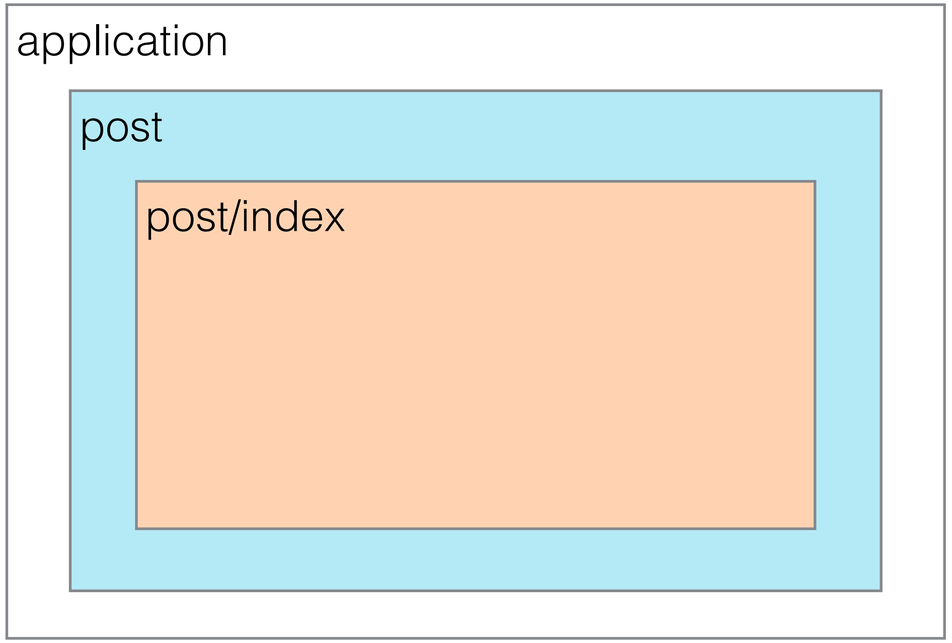

application
index
posts
post
index
author
comments
index
index
index

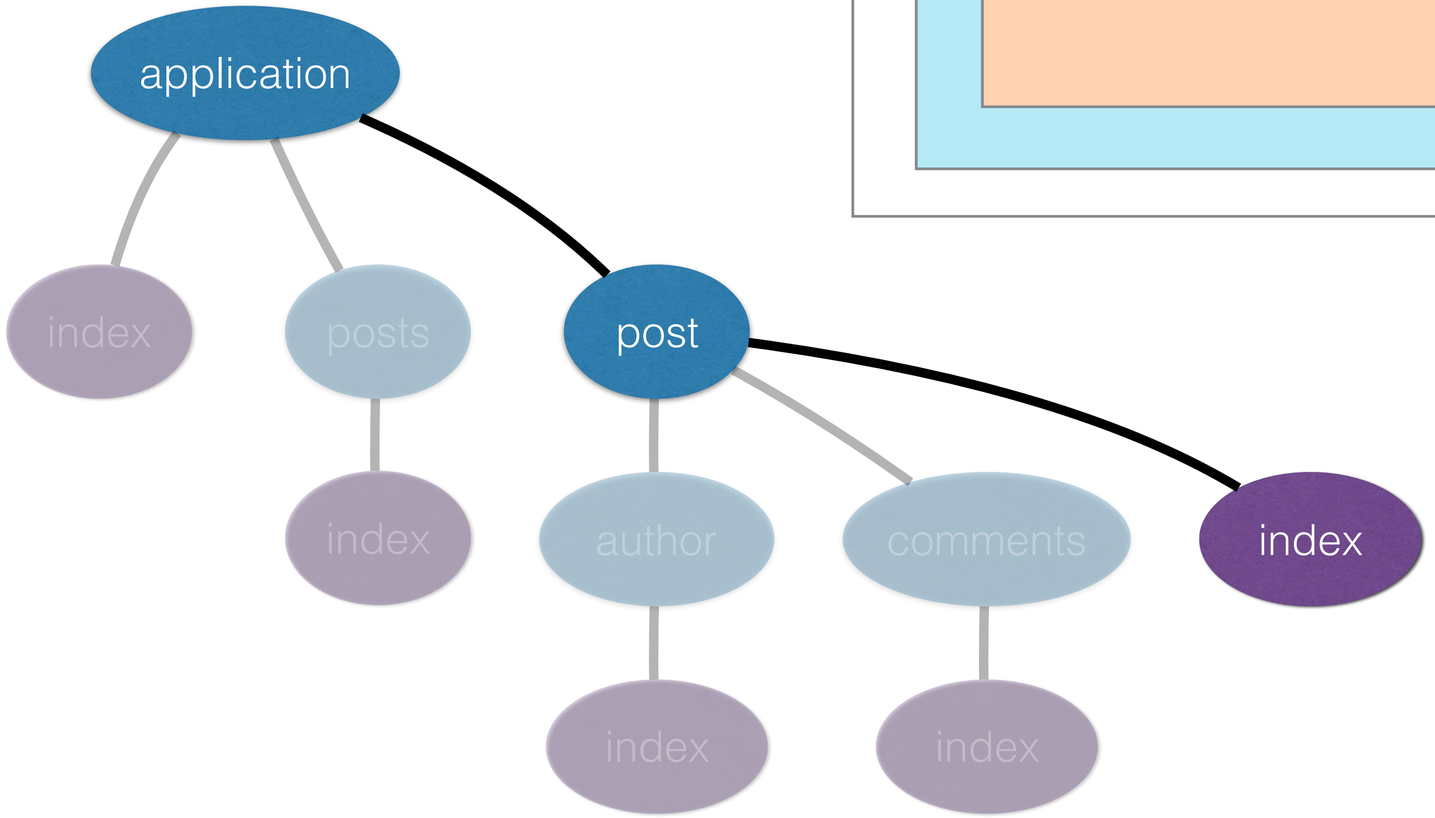
URL: `/posts/`


application
posts
posts/index

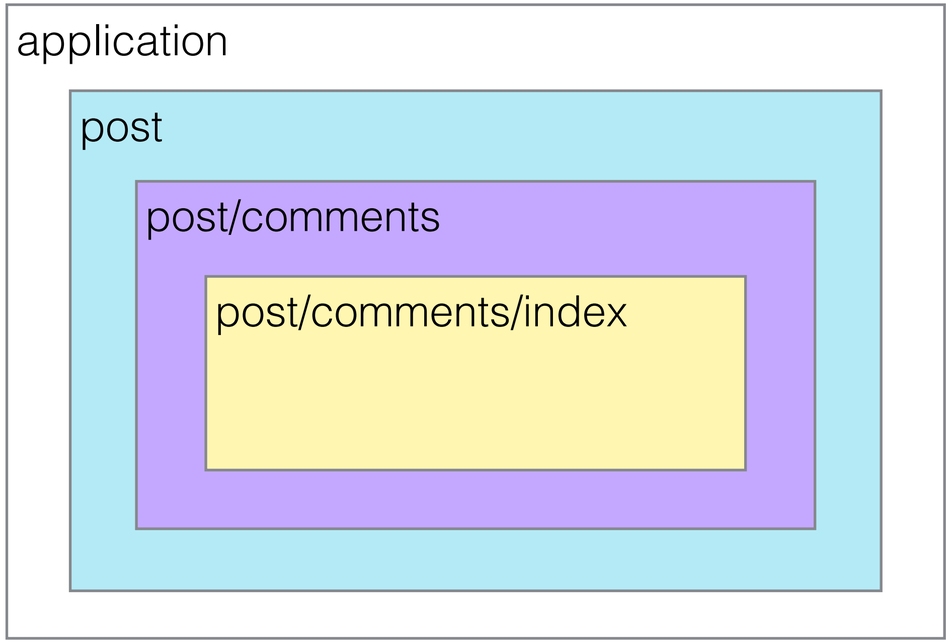

application
index
posts
post
index
author
comments
index
index
index

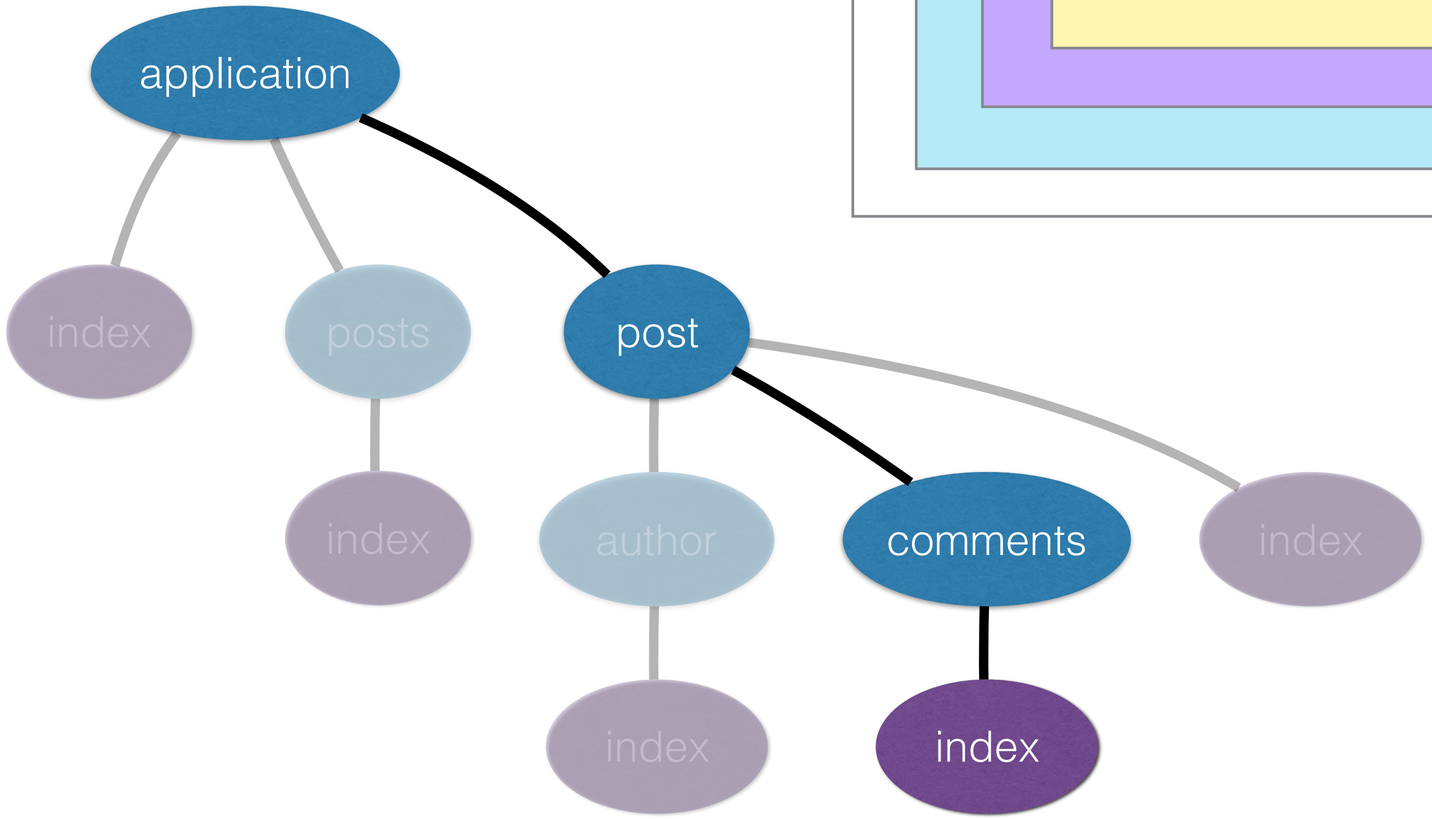
URL: `/post/31`


application
post
post/index

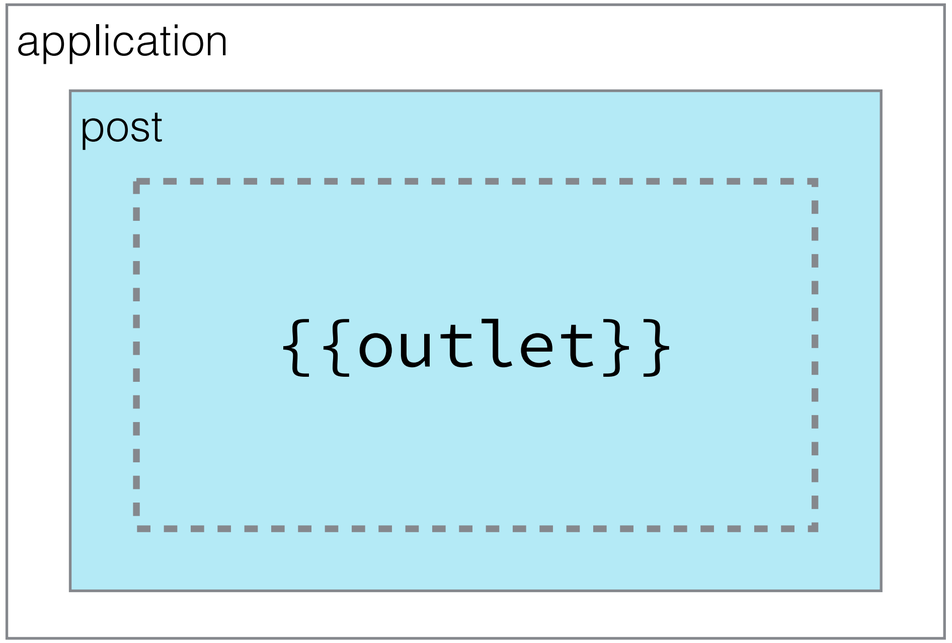

application
index
posts
post
index
author
comments
index
index
index

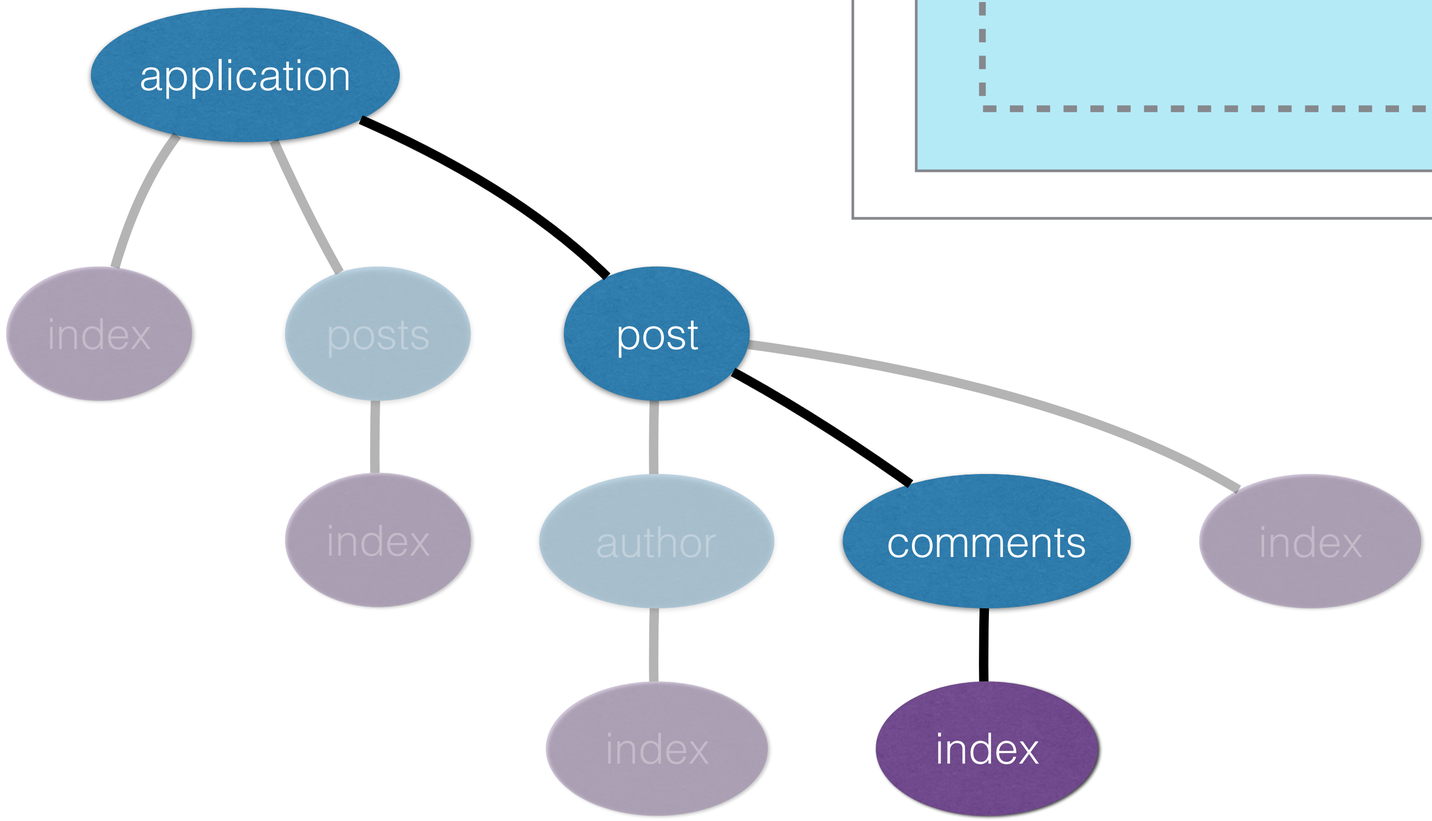
URL: `/post/31/comments/`





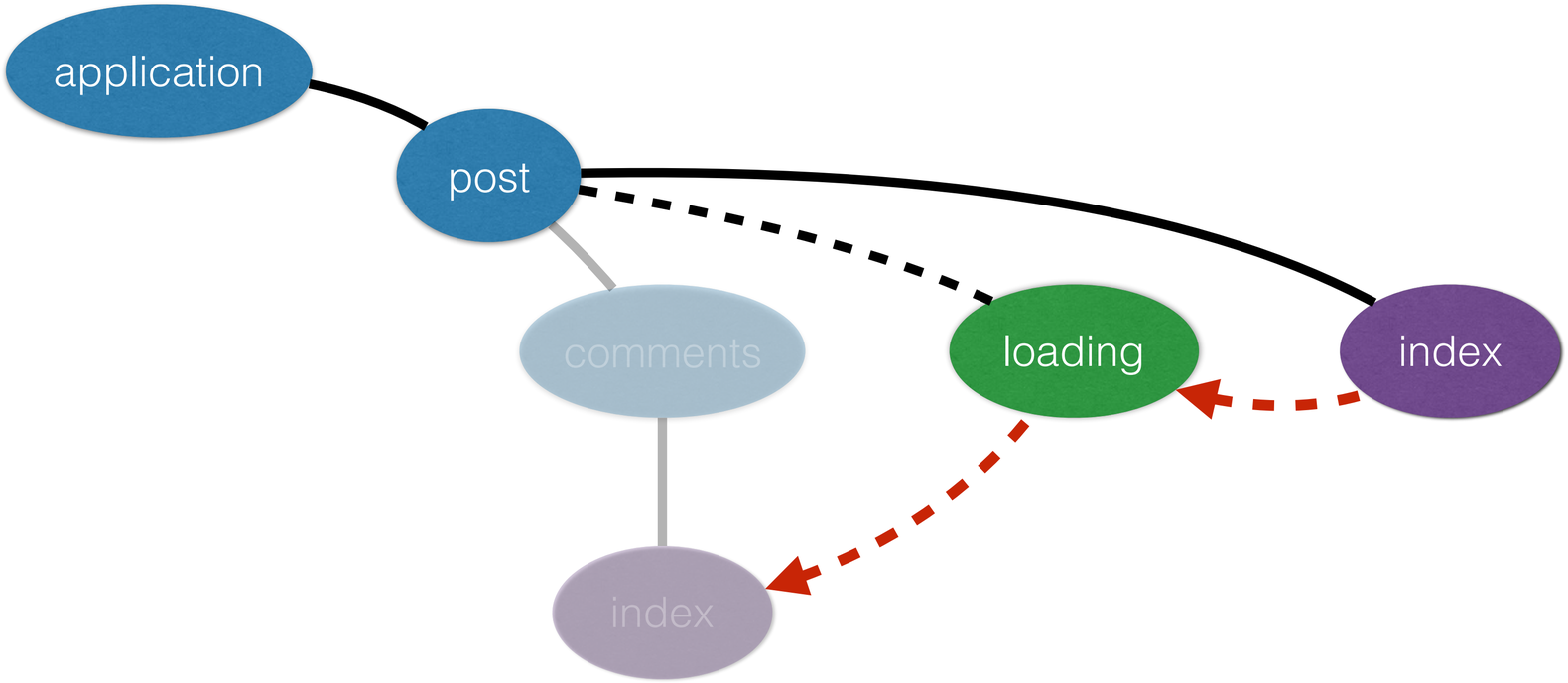
URL: `/post/31/comments/`





Loading Substate

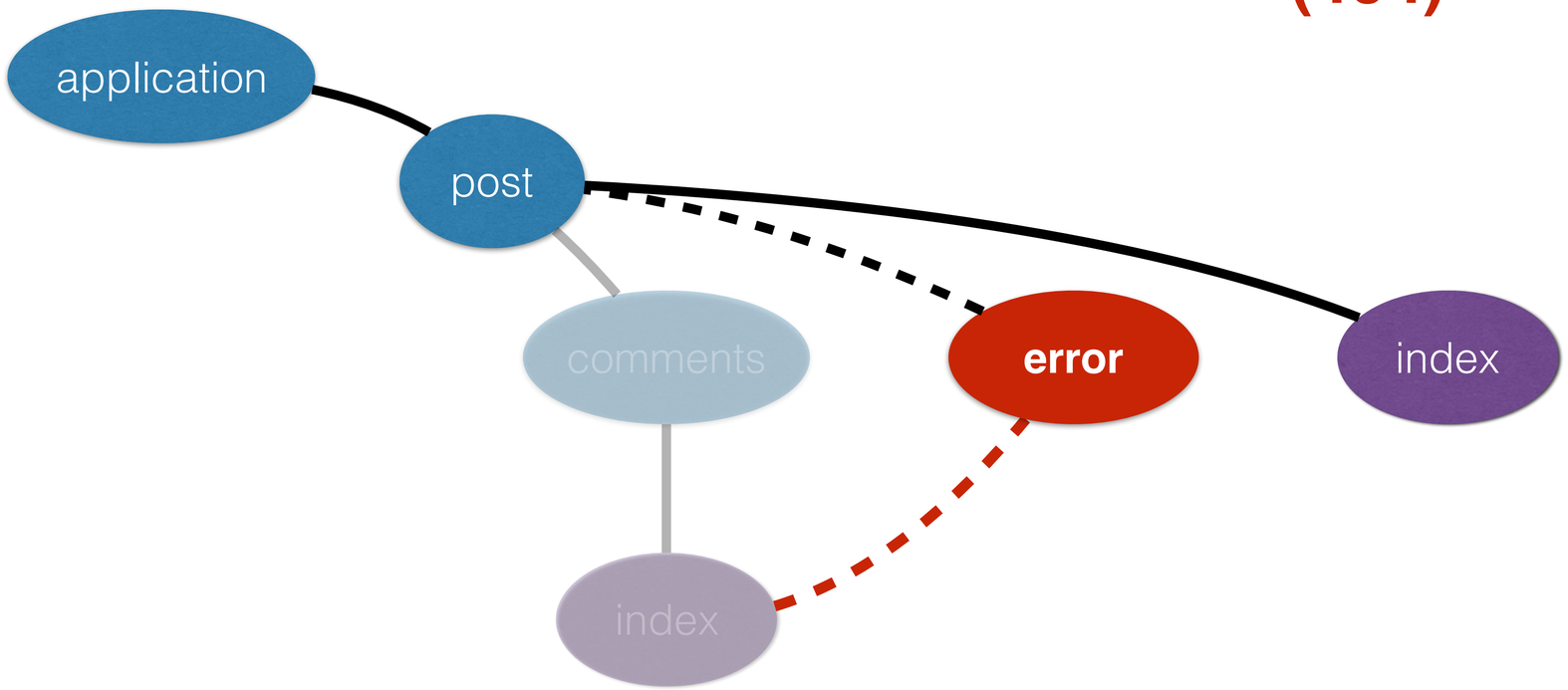
`/post/37` → `/post/37/comments`



Error Substate

`/post/37/comments` → `/post/9000001231`

(404)



Routing

Route Hooks

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 route
renderTemplate	`(controller, model)`	Render the template for the current route
deactivate	`()`	Called when router leaves route

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

Routes

```
model(params, transition)
```

This is the proper place for…

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

By default this does…

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

Routes

```
model (params, transition)
```

- Whatever this returns (or resolves to), will be available…
  - in templates as `model`
  - in other routes via `this.modelFor('post')`
- For multiple API calls, use `RSVP.hash`, `RSVP.all`
- Common location for light data massaging

Routes

```
afterModel(modelObj, transition)
```

This is the proper place for…

- Entry validation that requires the resolved model

By default this does…

- Nothing

Routing

Actions

- Actions can be placed on routes, controllers and components
- For consistency, we suggest always using closure actions

```
<span onclick={{action 'thingWasClicked' myObject}}>
  Click Here!
</span>
```

Routing

State Management Best Practices

- Routes are singletons
- Add state to routes with extreme caution.
- When using asynchrony, maintain awareness of state

Routing

Creating Links

Text Route Name

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

Text Route Name Model or ID

Routing

Creating Links

```
<!-- While transitioning into route -->
<a class='loading'>Go Home</a>

<!-- While resolved in route -->
<a class='active'>Go Home</a>
```

Components

- Encapsulated short-lived UI
- Action handling
- Well-orchestrated life-cycle
- Clear relationship w/ outside world

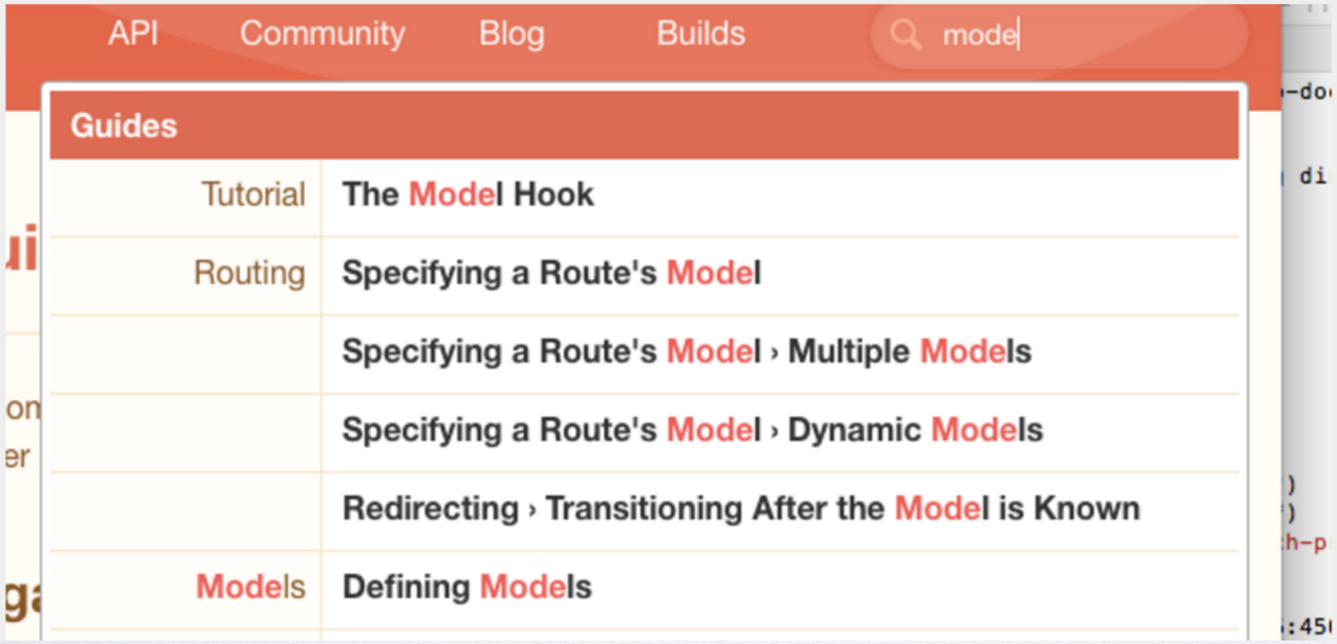
Component best practices

- Don’t confuse instance w/ prototype
- init() is cheap!
- always prefer super over on(‘init’)
- Don’t mutate observed state in didInsertElement

EXERCISE 1

http://localhost:4200/org/5/repo/66/pull/228

Mainly making the menu a bit smaller and more in keeping with the previous search menu design



EXERCISE 1

http://localhost:4200/org/6/repo/85/pull/292

```
@@ -242,8 +242,12 @@
     });
   }

+  if (self) {
+    jQuery = self.jQuery;
+  }
+
   processEmberShims();
   processTestShims();
-  generateModule('jquery', { 'default': self.jQuery });
+  generateModule('jquery', { 'default': jQuery });
```

EXERCISE 1

- using showdown and highlight.js, build me some things
  - {{markdown ‘mytext’}}
  - {{diff-text diff=‘mydiffText’}}
- https://github.com/showdownjs/showdown
- http://highlightjs.readthedocs.org/en/latest/css-classes-reference.html
- be ready to explain your approach

EXERCISE 1

- **What is the state?** text to be enhanced
- **Who does it affect?** Current User
- **How long should this state live?** Same as the UI
- **Where should it live?** - On the UI

GO!

Solution 1

- proper tear down?
- no double-rendering
- SafeString? (prefer {{{ }}} over unescaped HTML for clarity)
- was anything special needed on teardown? why? why not?

Navigational State

- Certain implicit contracts between people and web apps - violating them is bad UX
- Anything that seems like it should be “addressable” application state probably belongs in the URL
- Clear examples: list filters, “post id”, current locale, etc…

Navigational State

```
// Dynamic Segments
"/posts/:post_id/comments"

// Star Segments
"/posts/*path"

// queryparams
"/posts/:post_id/comments?order=created_at"
```

Navigational State

- star segments can include slashes
- queryParams have a nice handlebars helper, used w/ link-to

```
{{link-to "thing" “index"
   (query-params a=b c=d)}}
```

Exercise 2 - URL Based State

http://localhost:4200/org/4/repo/46/pulls?state=closed

Babel

babel.github.io

Pulls

Open Closed All

- [closed] - Replace JSON.stringify with better pretty-printer. Fixes #172
- [closed] - Inform about babel-node limitations.
- [closed] - Add documentation for AVA test framework
- [closed] - Update the docs of transform-react-inline-elements
- [closed] - Add the FAQ back, add some more questions

Exercise 2 - URL Based State

- Pull request list, filtering by open/closed
- Back button should work, passing your friend a URL should work, bookmark and hard reload should work
- Round-trip API call is needed (as opposed to live array being filtered)
- I want a specific loading substate for this experience

EXERCISE 2

- **What is the state?** status filter
- **Who does it affect?** Current User & other users w/ link
- **How long should this state live?** longer than the UI
- **Where should it live?** - URL

Exercise 2 - URL Based State

ABOUT THE API

- /api/pulls?repository_id=1&state=open
- /api/repositories/:repository_id/pulls?state=open

GO!

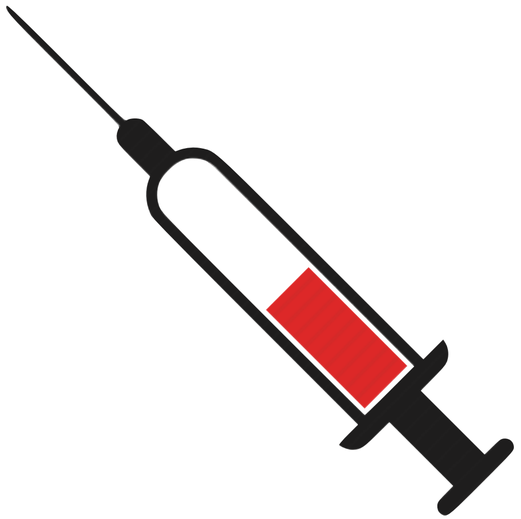
Services

- Singletons
- Very long lifecycle
- Means of managing shared app functionality/state
- live in the `/app/services` folder
- On need to know basis, `Ember.inject.service()`

Services

`Ember.inject.service(name)`

- `name` is optional
- Return value can be treated like a property
- Lazy
- No restrictions to what you can inject onto



Exercise 3 - PR View

ember-cli

ember-cli-shims

Guard against "self"

In FastBoot or other node envs, self is not a thing. Closes #43

app-shims.js

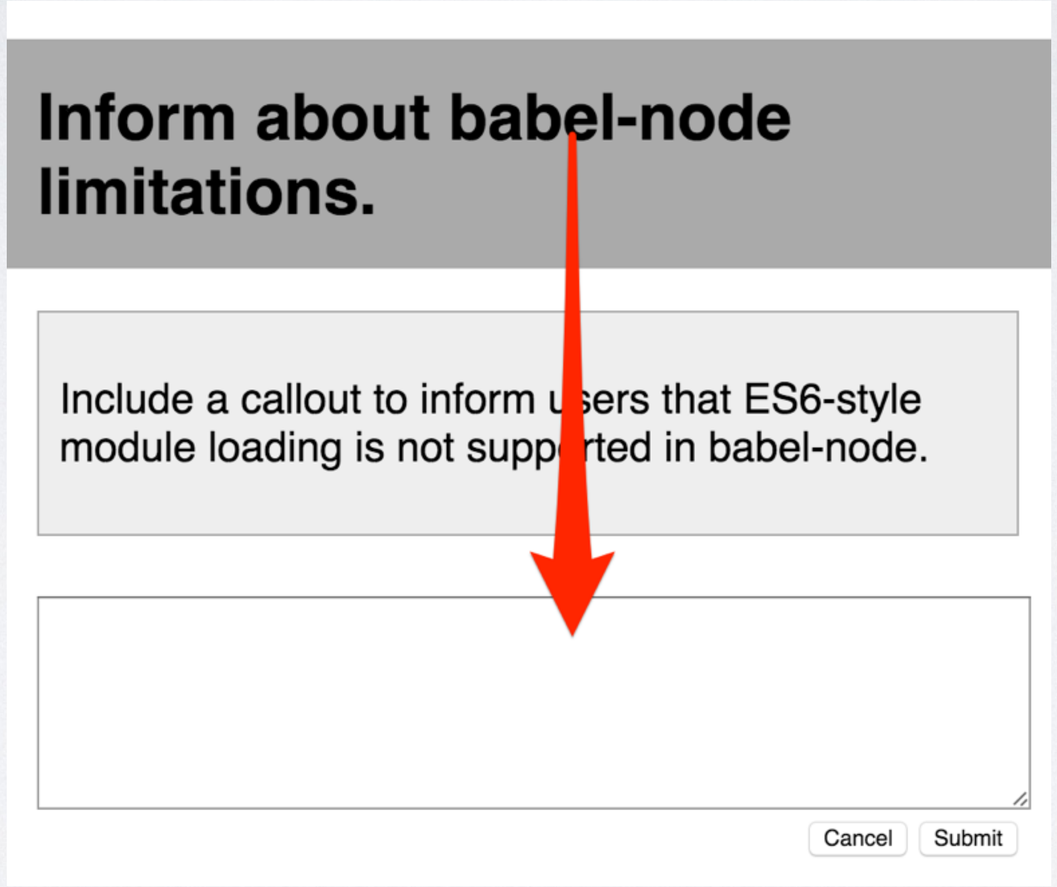
```
@@ -242,8 +242,12 @@
     });
   }

+  if (self) {
+    jQuery = self.jQuery;
+  }
+
   processEmberShims();
```

Exercise 3 - PR View

- Build me a PR view, using your fancy diff and markdown component.
- Also a list of comments which include a path (for inline comments) or no path (for the rest)

GO!

Exercise 3.1 - Save a comment



Exercise 3.1 - Save a comment

- Compose field for a new comment at the bottom of the PR view
- Should result in the appropriate POST request
- Comment-in-progress should not appear in the list of comments
- After saving, comment textarea should be empty

Using Store on a Component

```
import Ember from 'ember';

const { Component, inject } = Ember;

export default Component.extend({
  store: inject.service()
});
```

Creating a record w/ a relationship

```
this.store.createRecord('chapter', {
  book: this.get('bookModel')
});
```

EXERCISE 3.1

- **What is the state?** a new comment
- **Who does it affect?** all users of the system
- **How long should this state live?** forever
- **Where should it live?** - DB

GO!

Exercise 3.2 - Latency

- The API just got slow!
- Shouldn’t be able to double-post, or change text while save is in progress
- Shouldn’t see any errors in the console, if user navigates away from page before save is complete (ok to fail silently). They should even be able to go to another page and save another comment
- I want a loading indicator, connected to a boolean property on the component
- KEEP LIFE CYCLES IN MIND
- **BONUS:** Blank comment is rejected (422) via server-side validation

GO!

```
Error: Assertion Failed: calling set on destroyed object
    at new Error (native)
    at Error.EmberError (http://localhost:4200/assets/vendor.js:26688:21)
    at assert (http://localhost:4200/assets/vendor.js:16447:13)
    at Object.assert (http://localhost:4200/assets/vendor.js:26452:34)
    at Object.set (http://localhost:4200/assets/vendor.js:31081:22)
    at exports.default._emberMetalMixin.Mixin.create.set (http://localhost:4200/as
    at http://localhost:4200/assets/pull-request.js:79:17
    at tryCatch (http://localhost:4200/assets/vendor.js:63616:14)
    at invokeCallback (http://localhost:4200/assets/vendor.js:63631:15)
    at publish (http://localhost:4200/assets/vendor.js:63599:9)
```

Exercise 4 - Draft Comments

- Before saving a comment, the text should be considered a “draft”
- I should be able to have a draft per pull request
- When navigating to PR2 and back to PR1, PR1’s draft should remain
- Cancel button should clear the current draft
- No need to survive a reload
- **BONUS:** What might a solution look like that could handle multiple draft comments (i.e., comments at different locations on the PR)?

Exercise 4 - Draft Comments

- **What is the state?** Draft comment
- **Who does it affect?** Current User
- **How long should this state live?** Until the user discards the comment or unloads the pull request.
  - Doesn’t need to survive a hard reload
  - Should survive fwd/back and navigation to other pull requests. Each PR has up to one draft comment
- **Where should it live?** - We’ll find out :)

GO!

Comment

is UI state and it should:

- **NOT** pollute the ***pull request***
- live as long as the ***pull request***
- *often lives longer then the* ***UI***

Comment

NOT pollute the ***pull request***

imagine every piece of UI state
had to be on every model….

- *progress*
- *draft state*
- *item-controller stuff*
- *open/close*

Comment

NOT pollute the ***pull request***

Objects should remain encapsulated

Comment

live as long as the ***pull request***

So we can:

- *not leak memory*

when the primary object goes away,
the draft must also

Comment

*often lives longer then the **UI***

So we can:

- *navigating, should not cause it to be lost*

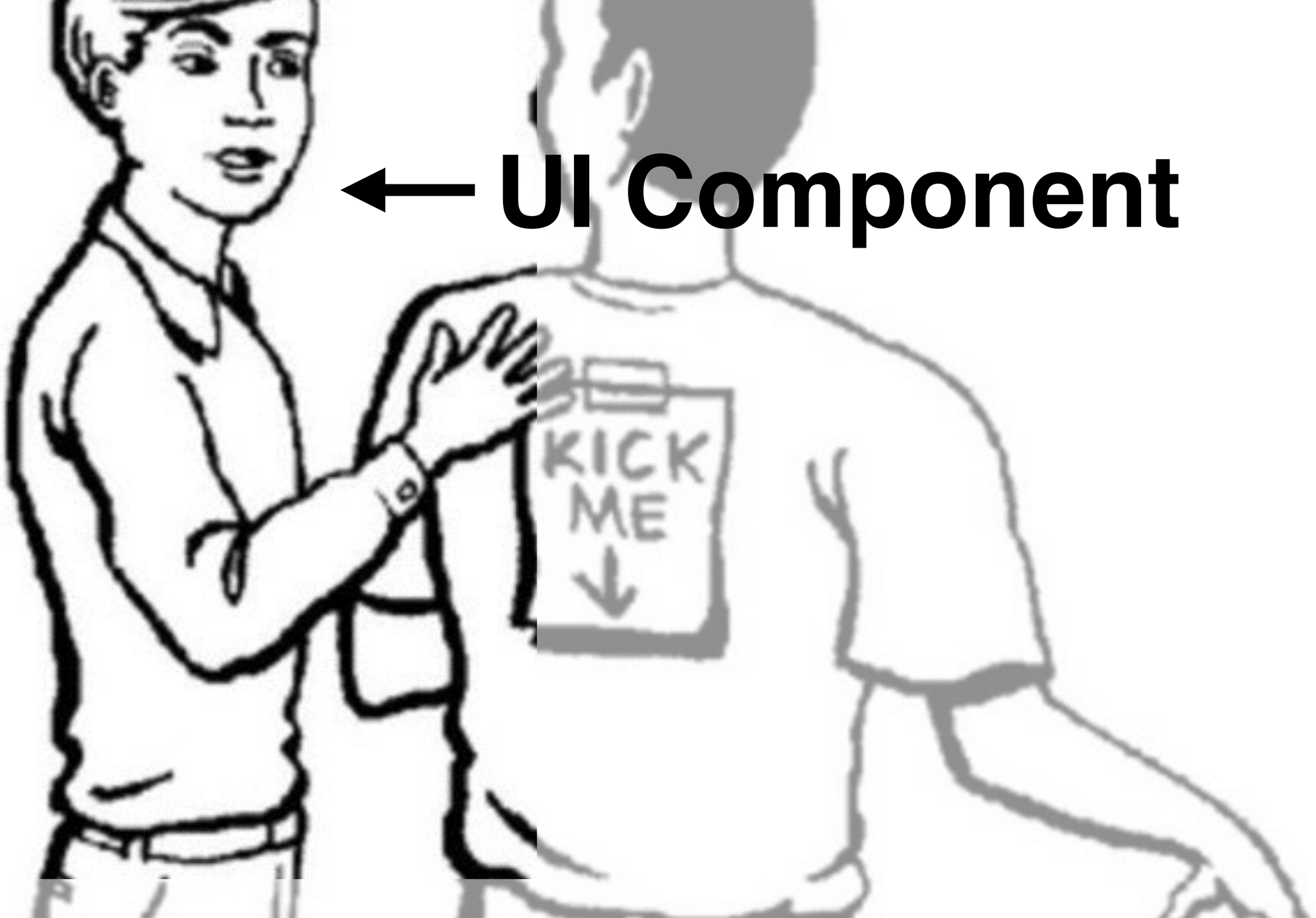
WeakMap

WeakMap?

it’s a map (key / value), but doesn’t retain ownership

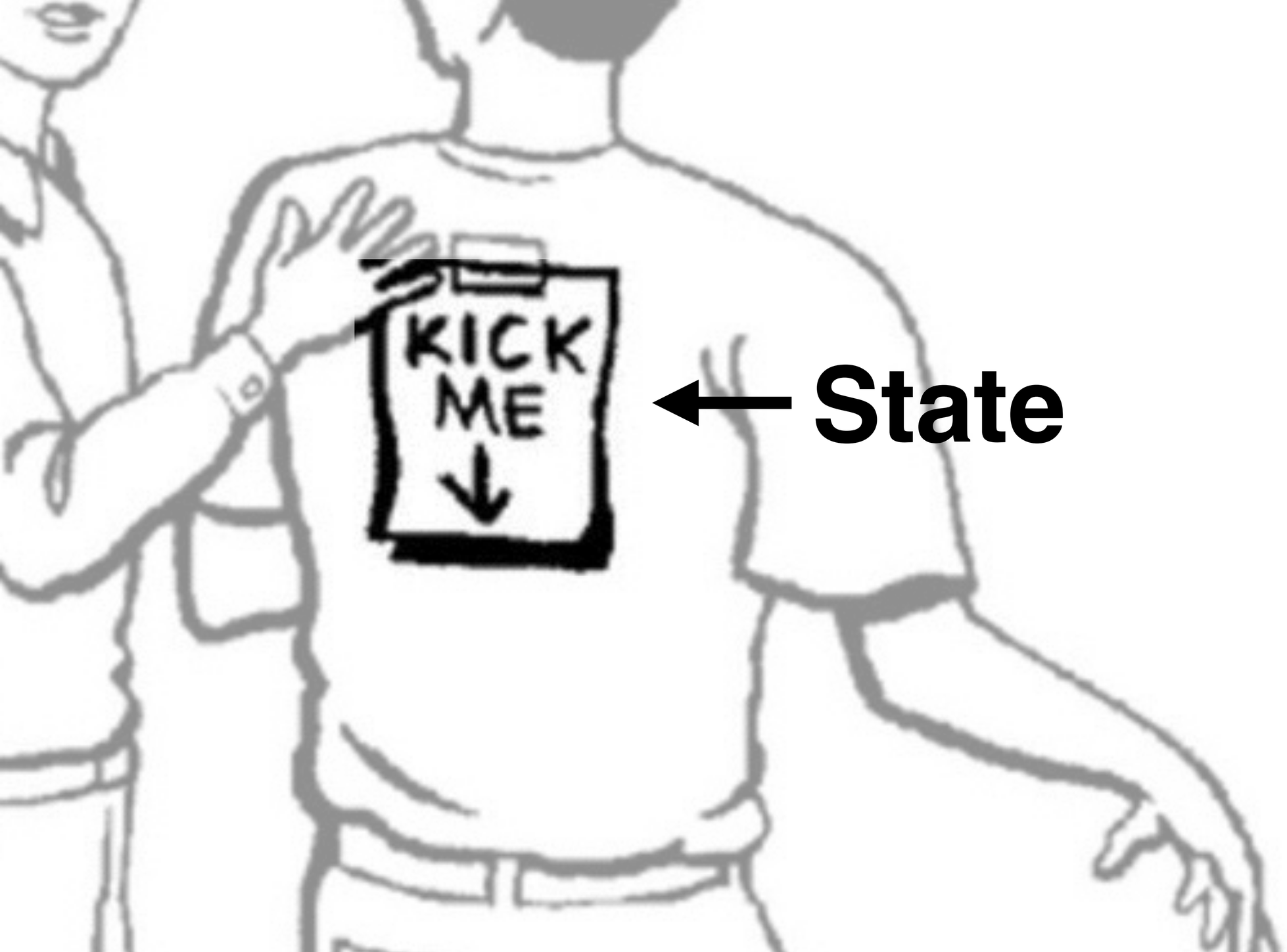
GC Magicks!!

Comment



← UI Component

← **Model**



← **State**

WeakMap

Comment

WeakMap

- **NOT** pollute the ***pull request*** ✅
- live as long as the ***pull request*** ✅
- *often lives longer then the* ***UI*** ✅

```
1 let map = new WeakMap();
2 map.set(pull, draftComment);
3 map.get(pull); // => draftComment;
```

ember install ember-weakmap

WeakMap

- important primitive ✅
- *good developer experience* ❌

Ember State Services

an ember-ish spin on weakmaps

```
1 import stateFor from 'ember-state-services/state-for'
2
3 export default Component.extend({
4   pull: null, // passed in as attr
5   draft: stateFor('comment-drafts', 'pull')
6 });
```

Ember State Services

an ember-ish spin of weakmaps

```
1 import stateFor from 'ember-state-services/state-for'
2
3 export default Component.extend({
4   pull: null, // passed in as attr
5   draft: stateFor('comment-drafts', 'pull')
6 });
```

WeakMap Name

path to weakMap key

weakMap as a Computed Property

```
1 component.set(‘pull’, pull1);
2 component.get(‘draft’) // => draft for pull1
3
4 component.set(‘pull’, pull2);
5 component.get(‘draft’) // => draft for pull2
6
7 pull1.destroy(); // draft goes away
8
```

```
previousComponent.set('pull', pull1);
previousComponent.get('draft') // => draft for pull1

// navigate away (causing the first UI to be torwn down)
// navigate back (resuming editing, powered by state-service)

nextComponent.set('pull', pull1);
nextComponent.get('draft') // => same draft as above
```

GO (IF WE HAVE TIME)!

What have we learned