# Data flow in Ember

@chadian 


kloeckner.i


ember
ember
PDX

# In the beginning

...

# How to get property of parent View in Ember?

Ask Question

17

2

This is probably beyond easy, but I'm having a hard time to figure out how to access properties of parent views:

```
App.ParentView = Ember.View.extend({
  foo: 'bar',

  child_view: Ember.View.extend({

    init: function(){
       // get the value of App.ParentView.foo
       //  ???
    }
  })

});
```

ember.js

share edit flag

asked Feb 14 '12 at 15:37

Panagiotis Panagi
7,092 ●5 ●41 ●91

asked 6 years, 3 months ago

viewed 10,278 times

active 4 years, 3 months ago



Please note that in many cases accessing the parent view is a sign of code smell. However, it's hard to say more without knowing details here. – Peter Wagenet Feb 19 '14 at 19:29

add a comment

30

To get the view: `this.get('parentView')`

To get the value of foo `this.get('parentView.foo')`

share edit flag

edited Feb 19 '14 at 19:28

Peter Wagenet
4,746 ●17 ●25

answered Feb 14 '12 at 16:22

Tom Whatmore
1,327 ●9 ●11

# Bindings

A binding creates a link between two properties such that when one changes, the other one is updated to the new value automatically. Bindings can connect properties on the same object, or across two different objects. Unlike most other frameworks that include some sort of binding implementation, bindings in Ember.js can be used with any object, not just between views and models.

The easiest way to create a two-way binding is to use a computed alias, that specifies the path to another object.

```
wife = Ember.Object.create({
  householdIncome: 80000
});

Husband = Ember.Object.extend({
  householdIncome: Ember.computed.alias('wife.householdIncome')
});

husband = Husband.create({
  wife: wife
});

husband.get('householdIncome'); // 80000

// Someone gets raise.
husband.set('householdIncome', 90000);
wife.get('householdIncome'); // 90000
```

# Handling Events

Instead of having to register event listeners on elements you'd like to respond to, simply implement the name of the event you want to respond to as a method on your view.

For example, imagine we have a template like this:

```
{{#view "clickable"}}
This is a clickable area!
{{/view}}
```

Let's implement `App.ClickableView` such that when it is clicked, an alert is displayed:

```
App.ClickableView = Ember.View.extend({
  click: function(evt) {
    alert("ClickableView was clicked!");
  }
});
```

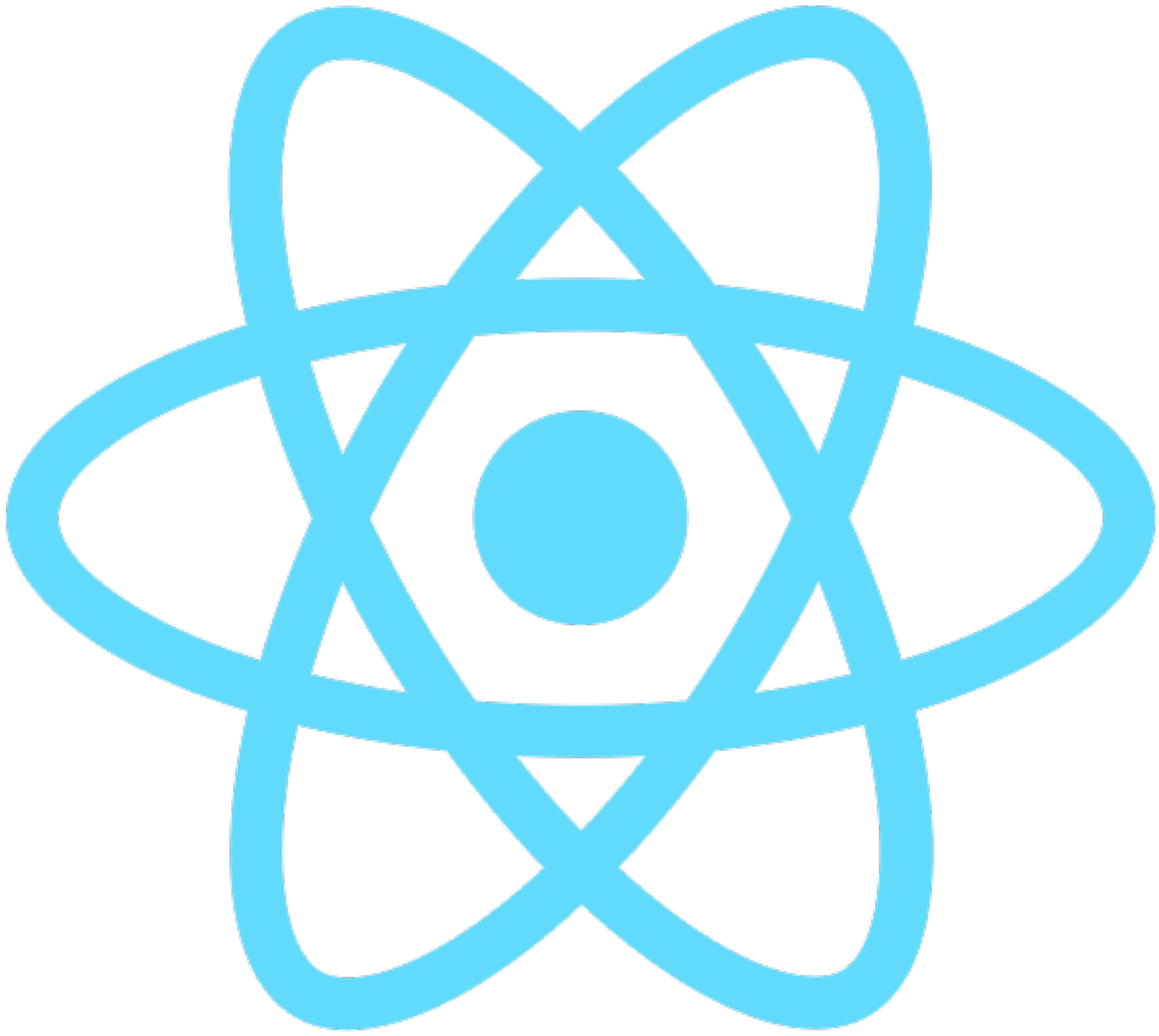
Events bubble up from the target view to each parent view in succession, until the root view. These values are read-only. If you want to manually manage views in JavaScript (instead of creating them using the `{{view}}` helper in Handlebars), see the `Ember.ContainerView` documentation below.

## Sending Events

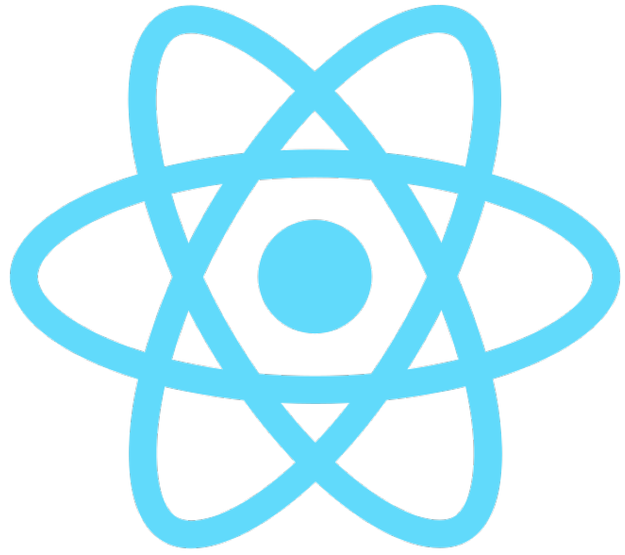
To have the click event from `App.ClickableView` affect the state of your application, simply send an event to the view's controller:

```
App.ClickableView = Ember.View.extend({
  click: function(evt) {
    this.get('controller').send('turnItUp', 11);
  }
});
```

# observers




SHUTTERSTOCK

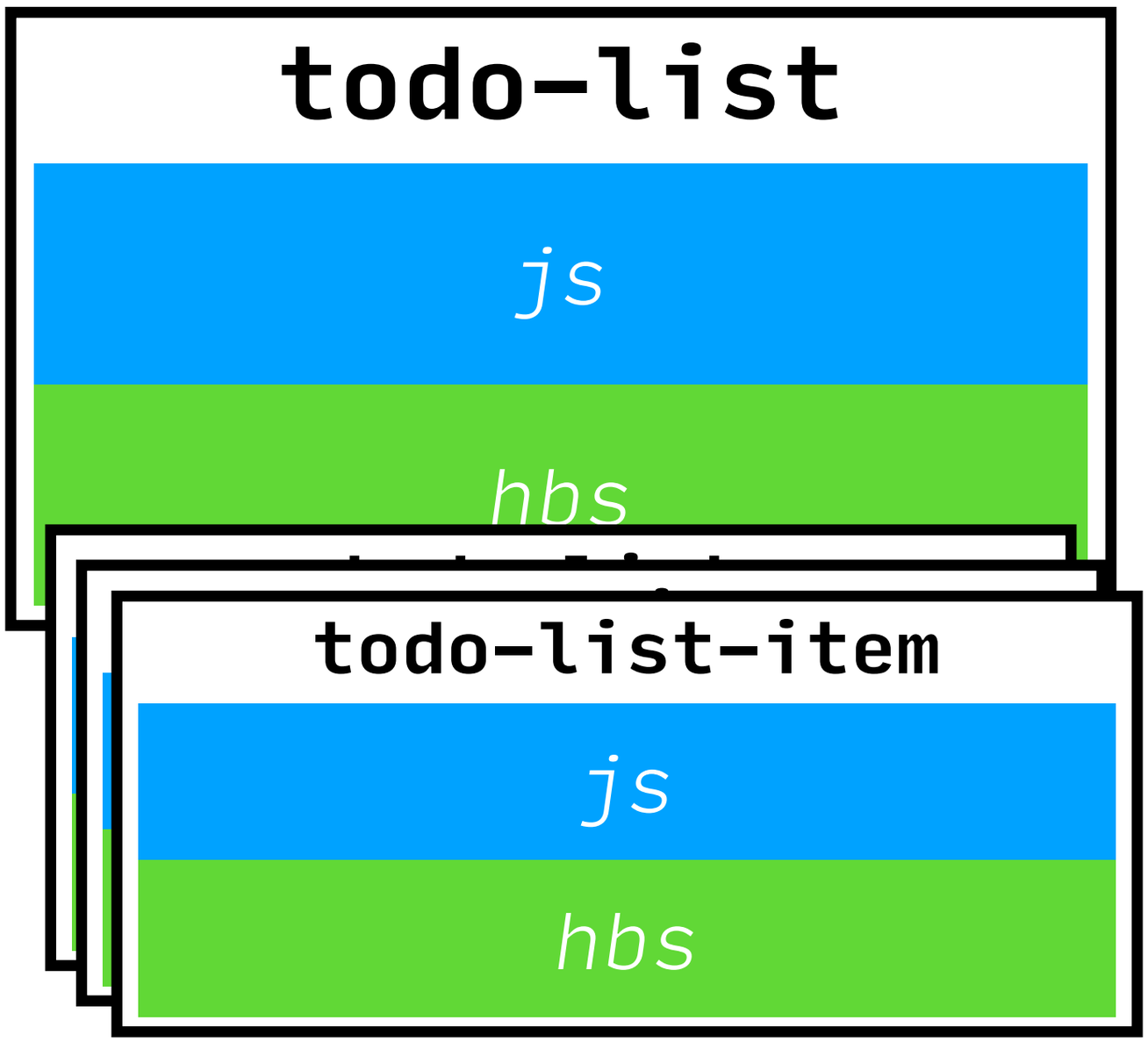


- State
- Props
- Component architecture
  - Presentational vs Container
  - High-order Components
  - Render Props ✨


Glimmer
2


in the present

- **Uni-directional data flow** and bindings by default
- Views are *gone*
- Components are first class citizens in the *template world*
- Lifecycle hooks



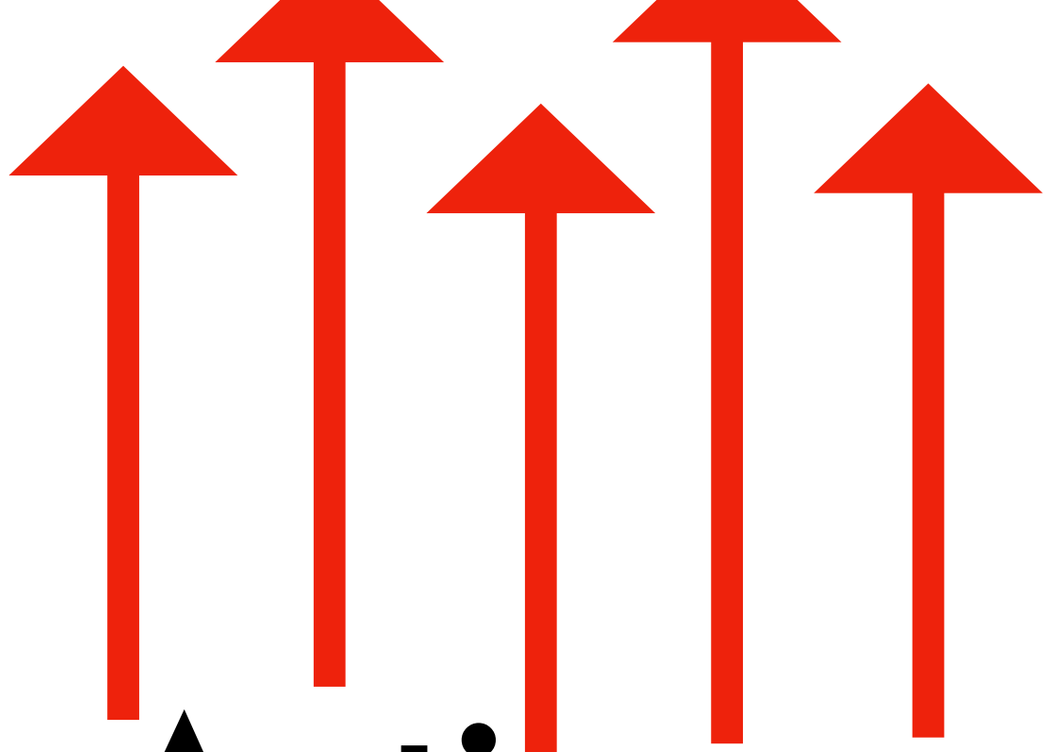

- **`.hbs`** templates
  templates are bound to the scope of their **`.js`**.

- Create new component instances from within **`.hbs`**

```hbs
{{!-- app/components/todo-list.hbs --}}
{{#each todos as |todo|}}
  {{todo-list-item todo=todo}}
{{/each}}
```

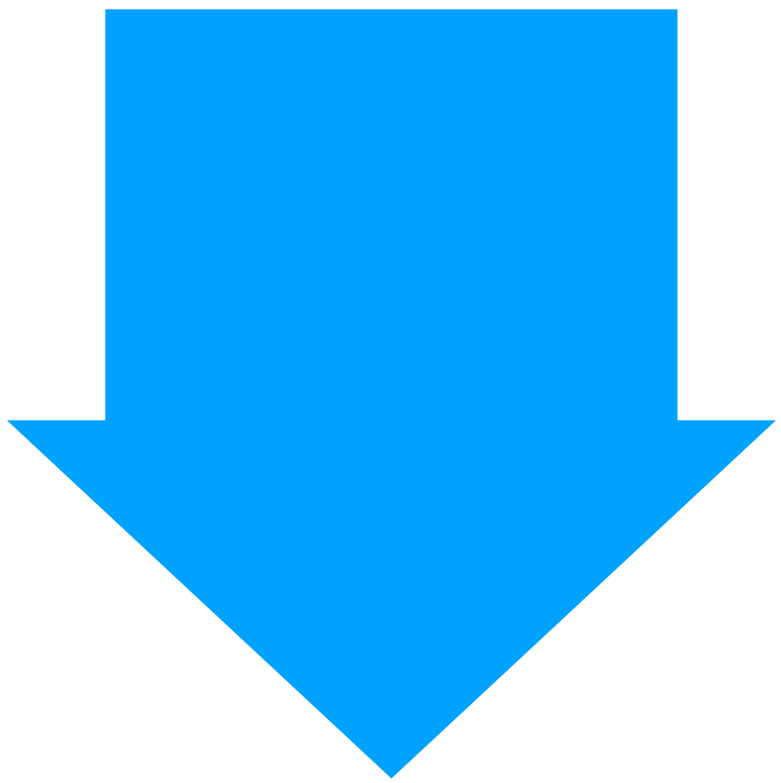
BDAU

# Data Down Actions Up



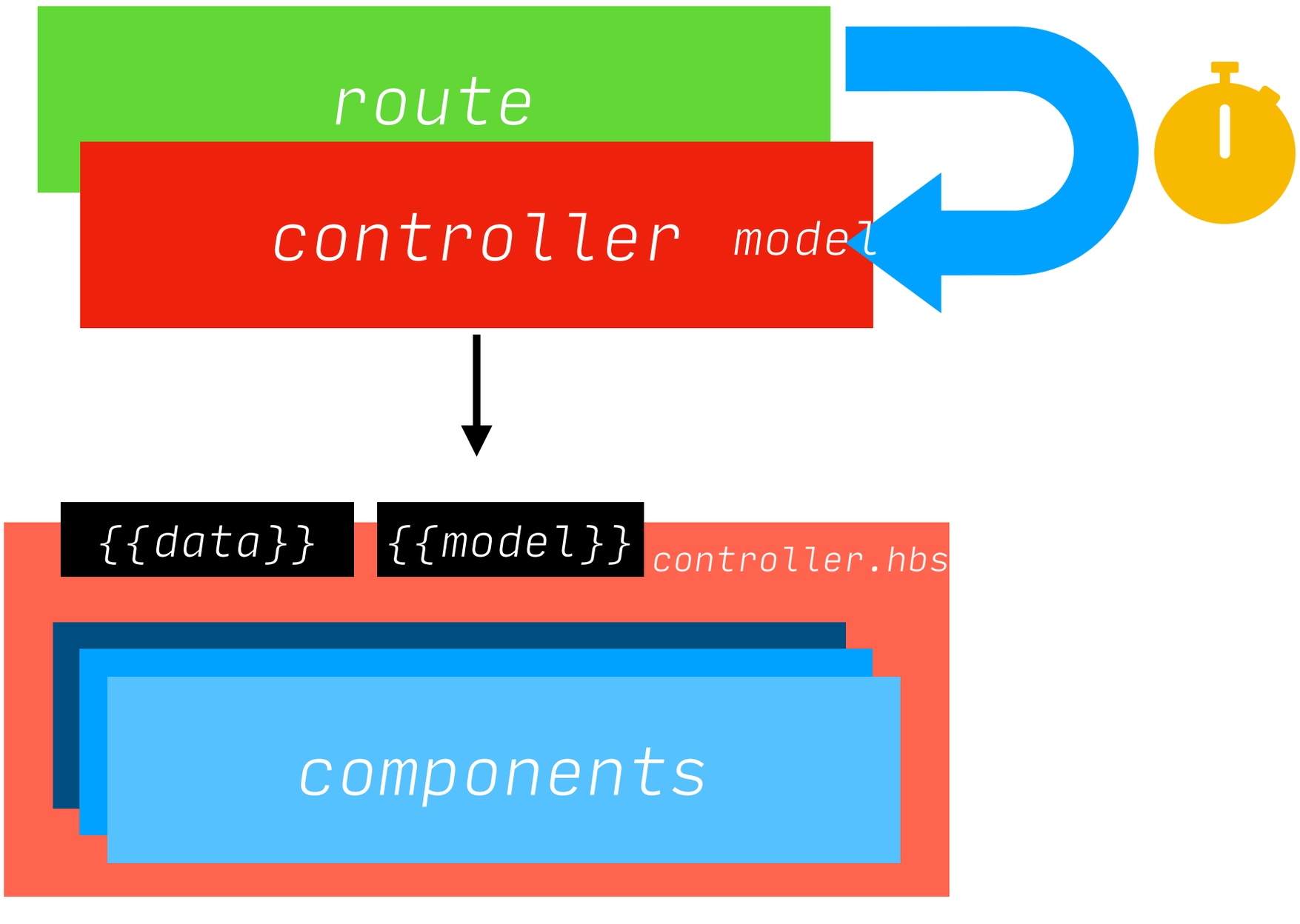
# Data Down Actions

Up



# the classical approach

# classical, data down



# classical, data down

{{model}}

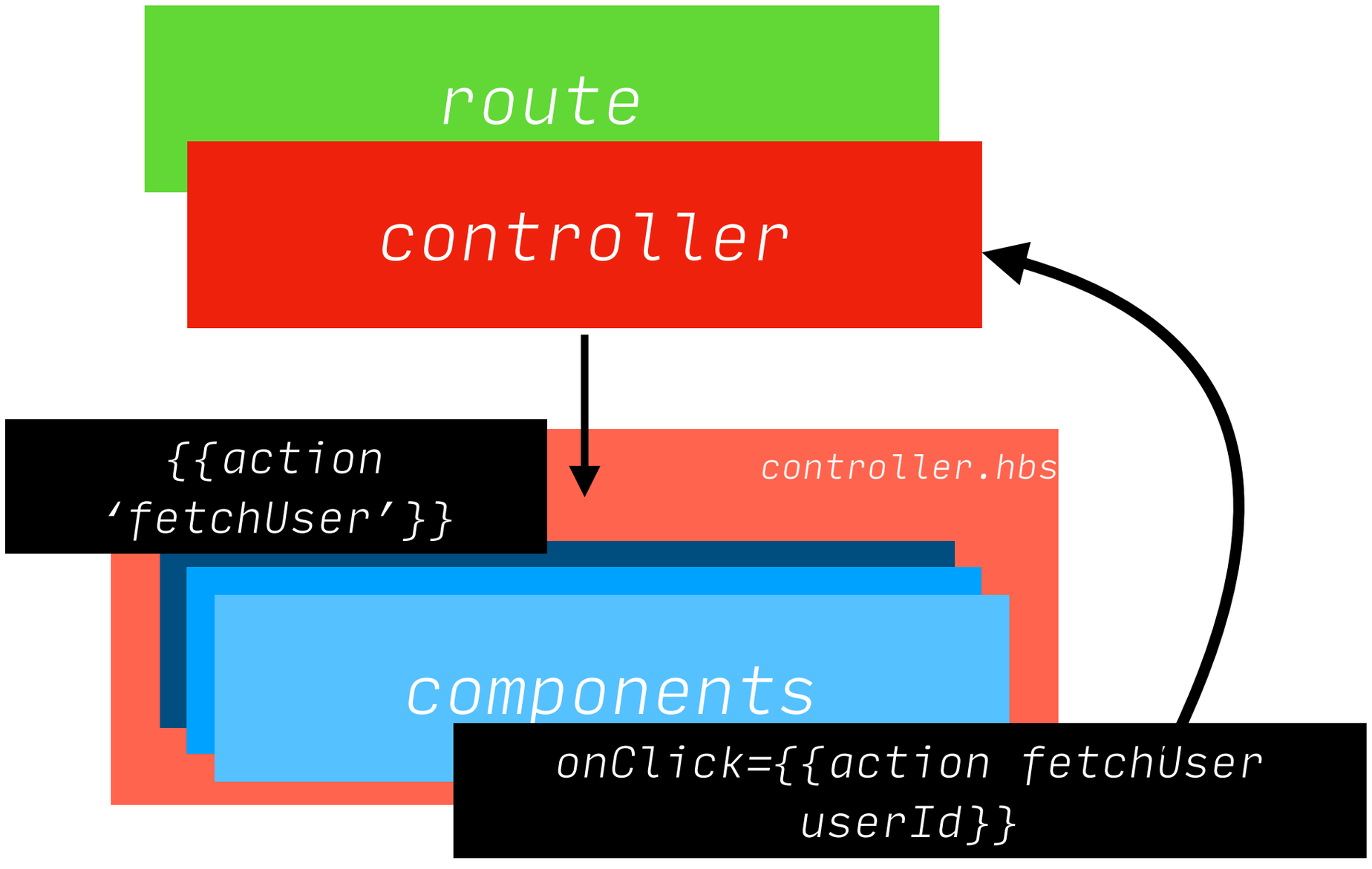
controller.hbs

component

component

component

# classical, actions up



# classical, data down

{{action 'fetchUsers'}}

controller.hbs

component

component

component

{{userId}}

onClick={{action 'fetchUers' userId}}

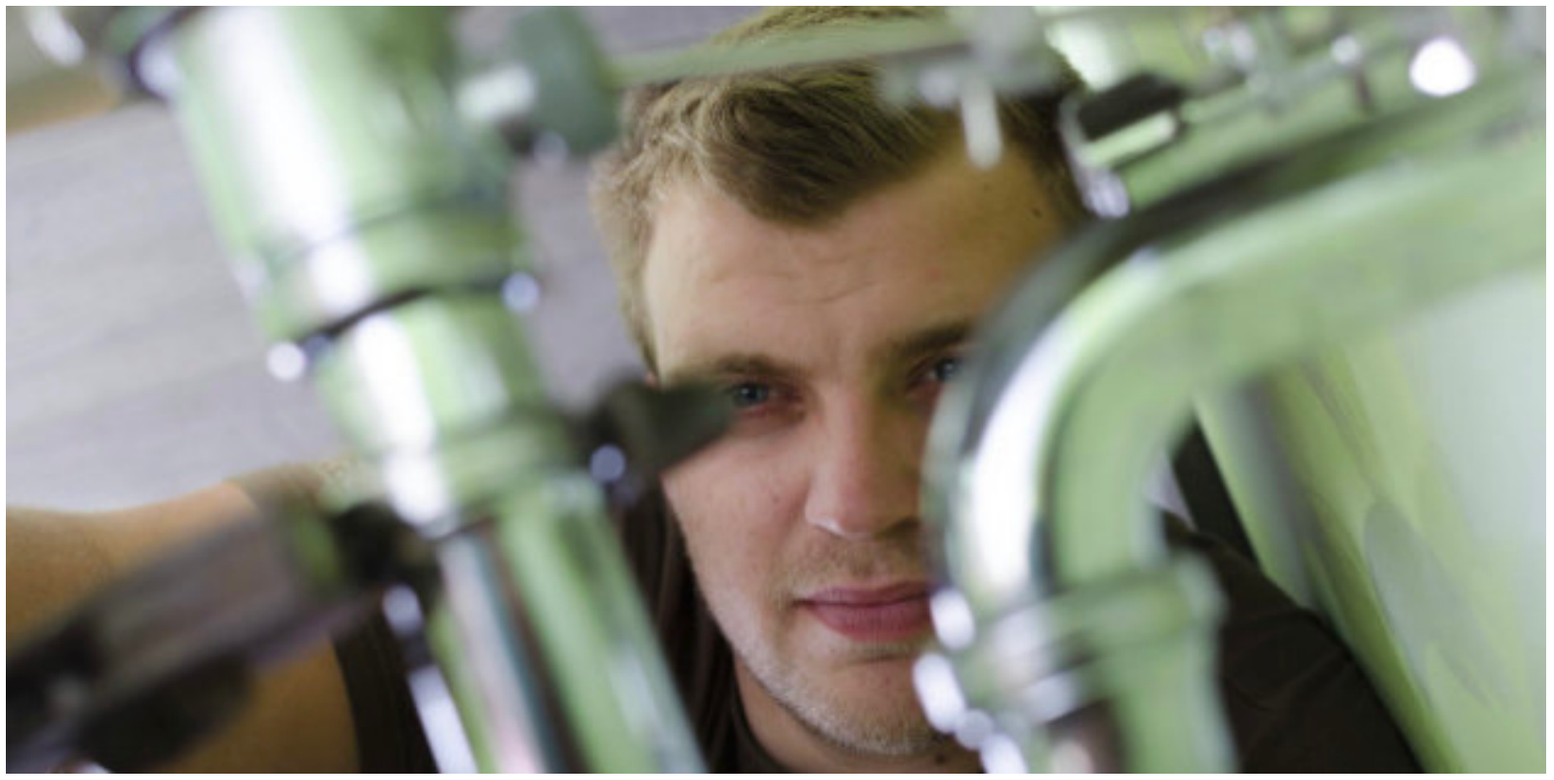
# giphy break

# okay, so data... down...

# data down...

- non-destructive
- read-only
- idempotent functions
  (zero side-effects)
- **from** → *refine* → **to**
- “pipe” (functional / unix),
  streams, observables, promises
- ***transformations***

# pipefitting our data (down)



| controllers | `model` |
| --- | --- |
| services | `session.get('isAuthenticated')` |
| computed properties | `authors: mapBy('blog', 'author')` |
| handlebar helpers | `{{t 'greetings.hello'}}` |
| get | `this.get('favouriteFood.firstChild.name')` |

# ACTION

# UP

dashboard.js

```
actions: {
  fetchUsers(userType) {
    this.get('userService').fetch(userType)
  }
}
```

dashboard.hbs

```
{{user-list
  users=users
  loadUsers=(action 'fetchUsers')
}}
```

user-list.hbs

```
{{app-button onClick=(action loadUsers
                            type)}}
```

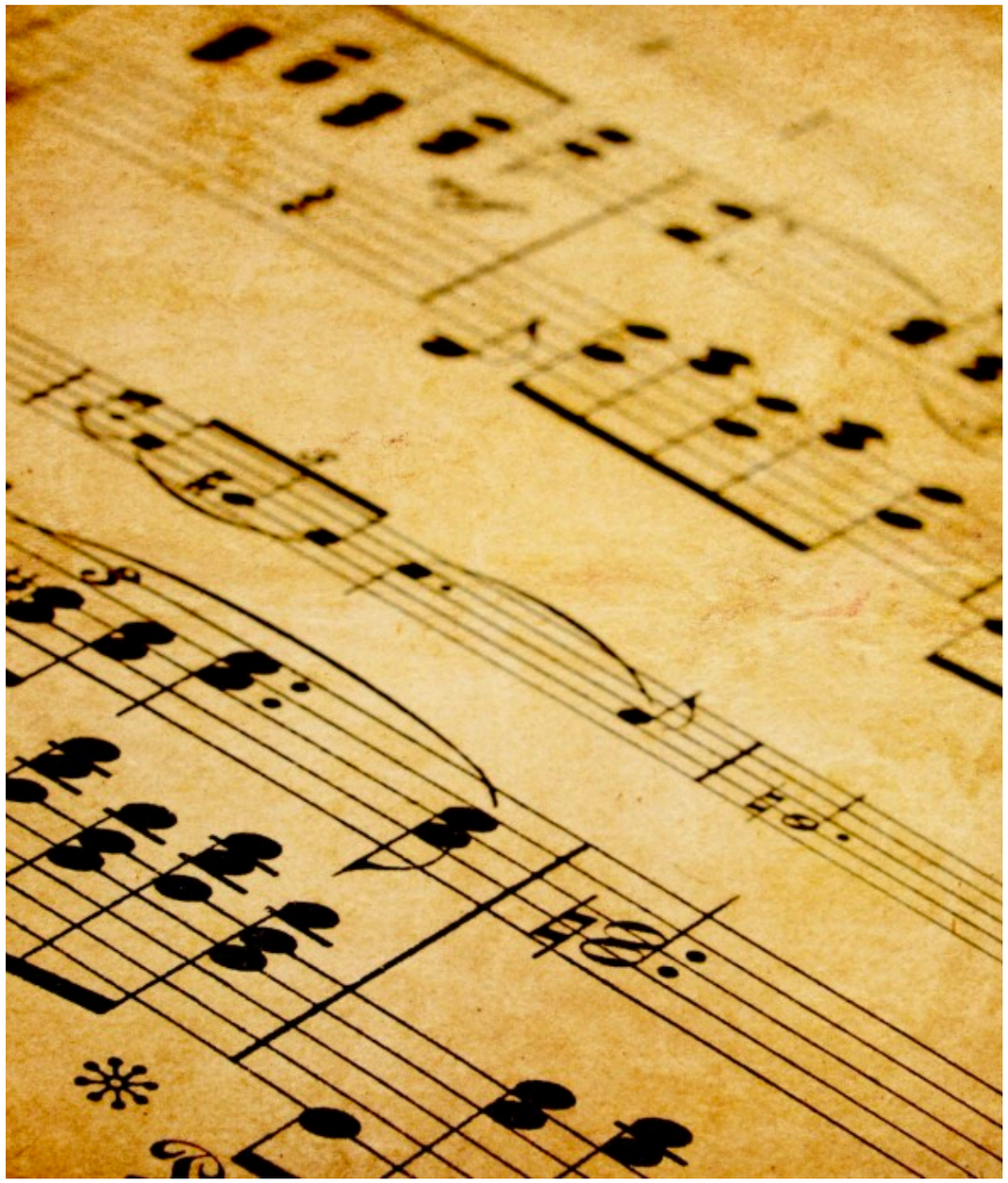
# Actions

- They’re callbacks, they’re functions.
- They’re passed down as data (functions are data)
- They’re invoked later (usually by a user action)
- `{{action}}` is a function helper for *scoping* and *lookup*.
  - scoping on the current template context
  - lookup, string => actions hash of js, reference on js

*Use actions as callbacks,*
*don’t use return values for state*

# soooo… the classical approach

- It works in theory
- … but times have changed
- Powerful component primitives
- Data Down, Actions Up



# in components we trust

*what if…*

- routes were for routing
- controllers were just a “top-level” component for the current route.
- components were smarter and could be responsible


9FAIL.COM

# the future

*most of data flow in applications*
*is through components*

*components with DDAU*

*in components we trust…*

# in components we trust

*“components are first class citizens”*

- They’re a *renderable primitive*

- Pass them around

```
{{component dynamicNameBinding
with=value}}
```

- organizable and exported by default

```
{{users/list/admin userData=user}}
```

# in components we trust

- Build bridges, not walls.

  `{{#my-component}}` **hash tag my component, hash tags are cool**

  `{{/my-component}}`

- Do work, setup a component, yield it out. Hide the implementation.

- Unless you're a leaf UI component, "boundary component", "wrapper component"you should probably be a block component.

```
{{#user-provider userType="admin" as |users|}}
  {{user-table users=users as |buildingBlocks formattedUsers|}}
    {{#buildingBlocks.table}}

      {{#each formattedUsers as |user|}}
        {{buildingBlocks.table-cell}}
          {{buildingBlocks.table-cell onClick=(action loadUserDetails)}}
        {{/buildingBlocks.table-cell}}
      {{/each}}

    {{/buildingBlocks.table}}
  {{/user-table}}
{{/user-provider}}
```

*when in scope you avoid prop drilling*

# functional components

- Stateless
- Just Render™ using args
- delete the *js* after *`ember g`*

**user-header.hbs** (the functional component)

```
<img class="user-photo" src={{url}}/>
<div class="user-name">
  {{name}}
</div>
```

**user-profile.hbs** (using the functional component)

```
{{user-header url=user.url
name=user.name}}
```

# stateful components

- Maintain state
- Don’t need a template file
- When all you need to do is yield, not really any markup.
- delete the *hbs* after *ember g*

**fetch-users.js**

```
import hbs from 'htmlbars-inline-precompile';

const layout = hbs`
yield (hash
  users=formattedUsers
  userCount=formattedUsers.length
)`;

export default Component.extend({
  layout,
  userType: null,
  users: null,

  filteredUsers: computed(‘users’, function() { … })

  actions: {
    changeType(type) { … }
  }
});
```


SCHNAPPS
MakeAGIF.com

*okay, show me something…*

⚠️ It works,

but experimental... 👩‍🔬

# emberbook

a friendly place online for all the Zoeys and Tomsters of the world

# {{with-anything}}

*give your look-up's a break*

```
{{#with-anything
  user="service:user"
  session="service:session"
  router="router:main"
  as |_|
}}
  {{#if _.session.isAuthenticated}}
    Hello {{_.user.name}}

    {{#if _.user.isAdmin}}
      [ ADMIN ]
    {{/if}}

    <button {{action (bind _.session 'logout')}}>
      Logout
    </button>
  {{else}}
    <button {{action (bind _.router "transitionTo" "login")}}>
      Login
    </button>
  {{/if}}
{{/with-anything}}
```

*get **anything** from the container*

*named as needed, data down*

*easy access to service state*

*bind on to service actions*

# Demo

# {{round-trip}}

*it's the model hook you didn't know you needed*

```
{{#round-trip model=promise as |_|}}
  {{#if _.isSuccessful}}
    {{user-list users=_.model}}
  {{/if}}

  {{#if _.isLoading}}
    {{loading-spinner}}
  {{/if}}

  {{#if _.isFailure}}
    {{error-panel error=_.error}}
  {{/if}}
{{/round-trip}}
```

***resolved*** *result*
*hey, just like the model hook*

*loading substate*

*error substate*

# {{round-trip}}

*slight syntax tweaks for loading and error, coming soon…*

```
{{#round-trip
  model=promise
  loading=(component "generic-loading")
  error=(component "generic-error")
  as |resolved|
}}
  {{!-- used resolved promise --}}
{{/round-trip}}
```

*substates in components*

*focus on resolved result*

# Demo

# One more thing...

# Demo

# {{load-warp}}

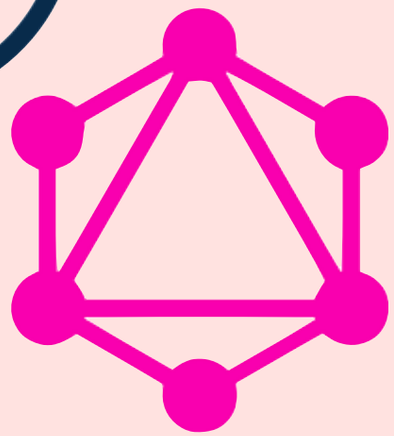
*just wait there one more second…*

```
{{#load-warp threshold=1500 model=model as |warpedModel|
}}
  {{#round-trip model=warpedModel as |_|}}
    {{#unless _.isLoading}}
      {{friends-list model=_.model}}
    {{else}}
      {{component "loading-spinner"}}
    {{/unless}}
  {{/round-trip}}
{{/load-warp}}
```

# cache strategy

emberdata

Orbit



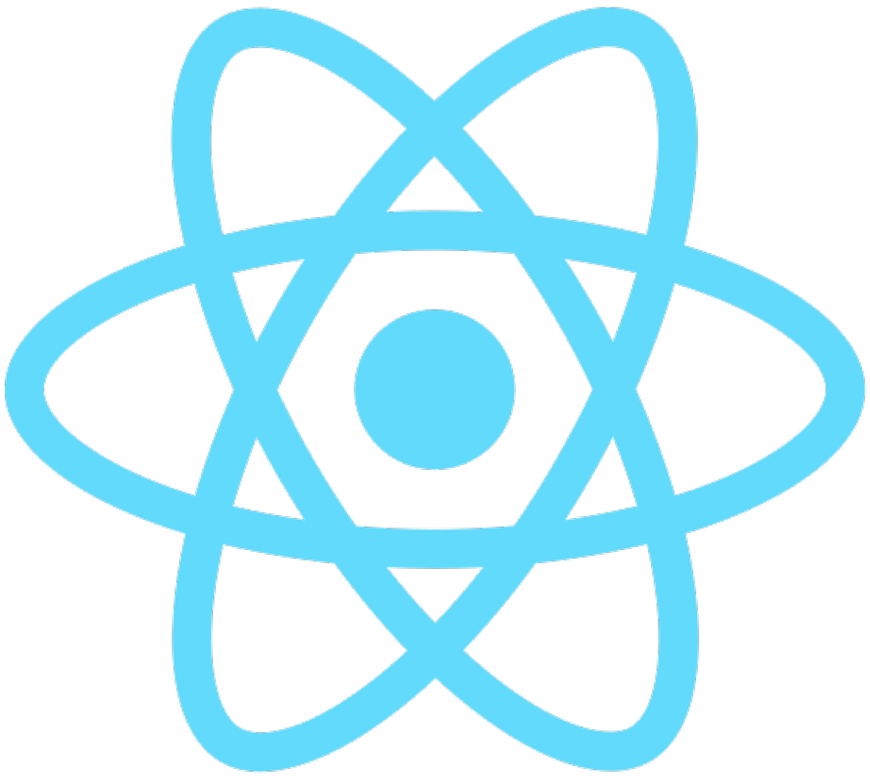
ember-redux

# Recap

- Data flows down
- Actions (callback) up
- Your data and actions can come from different places and take different forms
- The model hook blocks, difficult for fine tuning
- Cache is important
- The future looks good

# Background notes

- ember-concurrency
- ember-parachute
- ember-elsewhere
- Skeleton Screen Loading in Ember.js
- Adopting ember-concurrency or: How I Learned to Stop Worrying and Love the Task



# `ember install ember-cli-react`

Disclaimer: I am not responsible for production issues
resulting from the installation of this add-on.

kloeckner.i


@chadian

github.com/chadian/emberbook