Developing Ambitious Ember apps in 2019

Dilip Kushwaha
Software Engineer @LinkedIn

Twitter: @kushdilip
Github: @kushdilip

Why Ember in 2018?

Pros

- Convention over Configuration (CoC)
- Addon ecosystem (emberobserver.com)
- Out of the box routing
- Ember-data
- Ember CLI
- Built-in testing tools.
- Ember Inspector
- Active community and support.

Why not Ember in 2018?

Limitations

- Dependency on jQuery
- Steep learning curve
- Complex component API
- No Support for Native Class, Decorators etc
- Unintuitive project structure

Ember Release Cycle

Ember Release Cycle

https://emberjs.com/releases/

Ember Release Cycle: SemVer

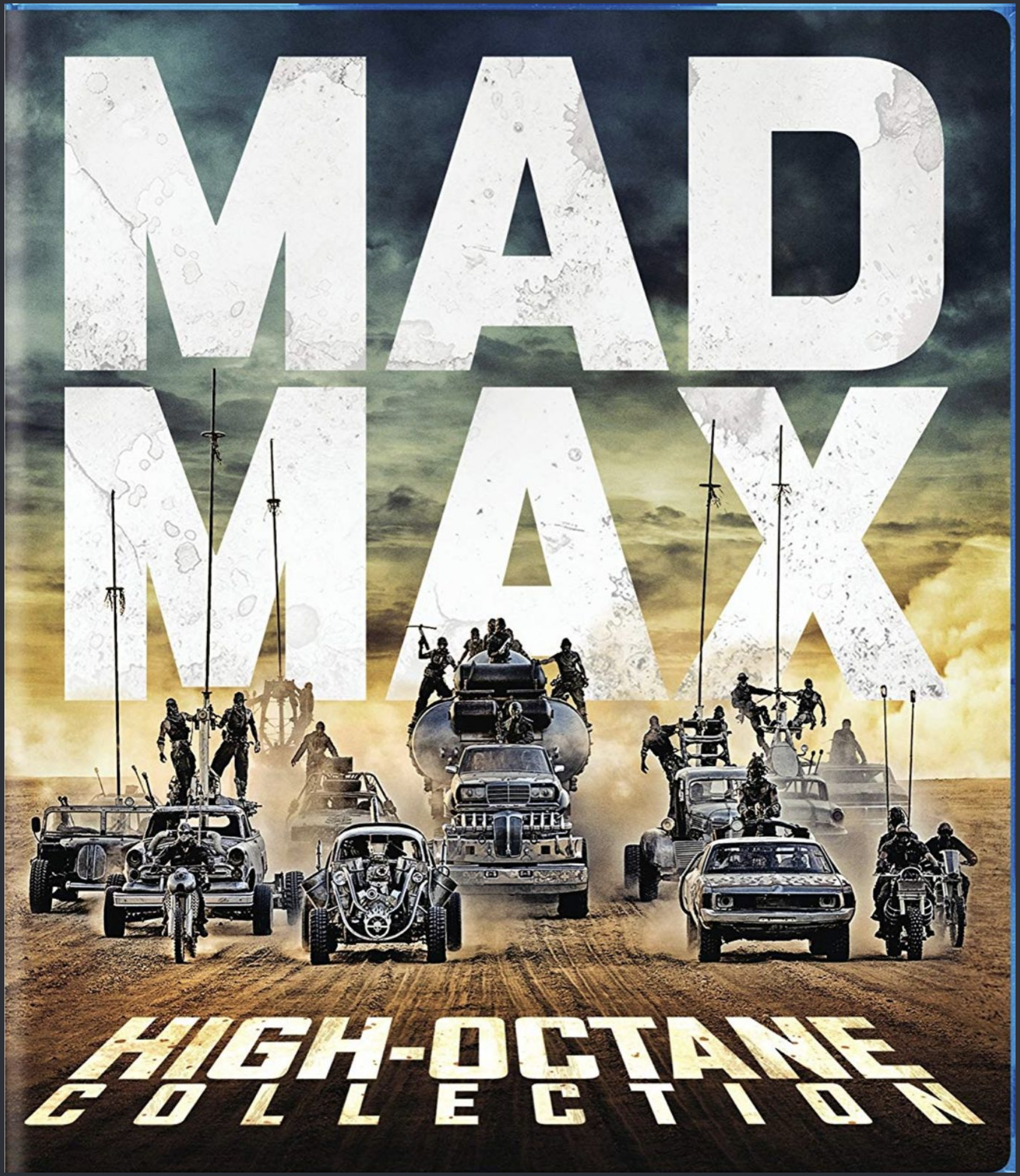
Minor

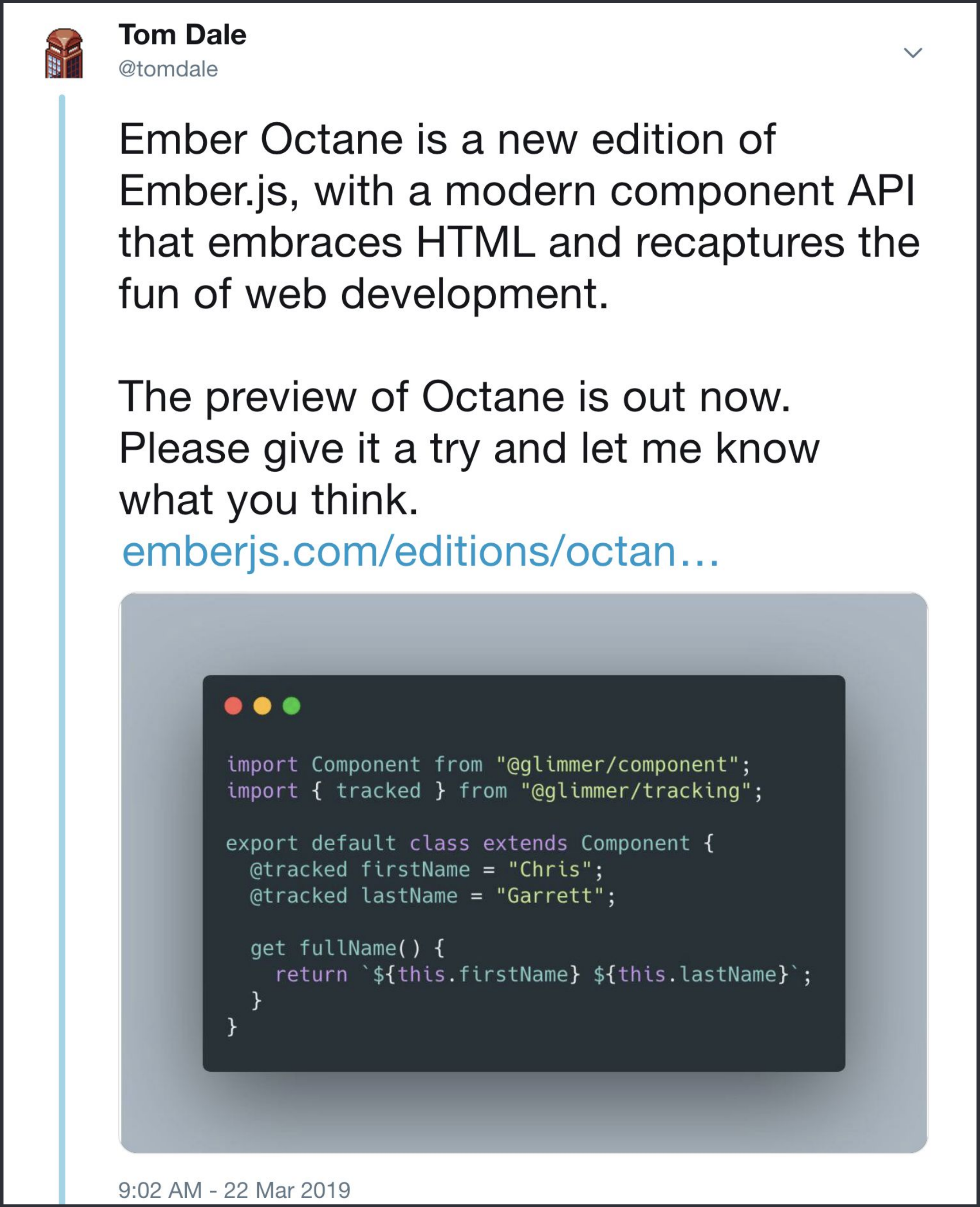
Introduce new features

Major

No new features. Only fixing deprecations.

Editions & Ember Octane


MAD MAX
HIGH-OCTANE
COLLECTION



Tom Dale
@tomdale

Ember Octane is a new edition of Ember.js, with a modern component API that embraces HTML and recaptures the fun of web development.

The preview of Octane is out now. Please give it a try and let me know what you think.
emberjs.com/editions/octan…

```
import Component from "@glimmer/component";
import { tracked } from "@glimmer/tracking";

export default class extends Component {
  @tracked firstName = "Chris";
  @tracked lastName = "Garrett";

  get fullName() {
    return `${this.firstName} ${this.lastName}`;
  }
}
```

9:02 AM - 22 Mar 2019

Why Octane?

What’s new

- Native JavaScript classes
- Decorators
- Tracked properties
- Async functions (async/await)
- Octane-style components
- ...attributes.
- <AngleBracket> syntax

Why Octane?

What's gone

- jQuery.
- Non-native classes.
- Computed properties and observers.
- Curly component invocation
- The run loop.
- Ember "inner HTML" components

How to Octane?

```
#create ember octane app

ember new octane-app -b @ember/octane-app-blueprint
```

Required tooling

```
# install Volta
curl -sSLf https://get.volta.sh | bash

# install Node
volta install node

#install ember
volta install ember-cli
```

Feature Flags

```
//config/environment.js
module.exports = function(environment) {
  let ENV = {
    'ember-resolver': {
      features: {
        EMBER_RESOLVER_MODULE_UNIFICATION: true,
      },
    },
    //more properties
    EmberENV: {
      FEATURES: {
        // Here you can enable experimental features on an ember canary build
        EMBER_MODULE_UNIFICATION: true,
        EMBER_NATIVE_DECORATOR_SUPPORT: true,
        EMBER_METAL_TRACKED_PROPERTIES: true,
        EMBER_GLIMMER_ANGLE_BRACKET_BUILT_INS: true,
      },
    },
  };
  //.... modify env based on development, prod etc.
  return ENV;
};
```

Glimmer Components

Ember's new component API in Octane

Classic component

- Curly syntax
- Wrapper Element
- Arguments
- Despite DDAU approach, 2 way binding possible

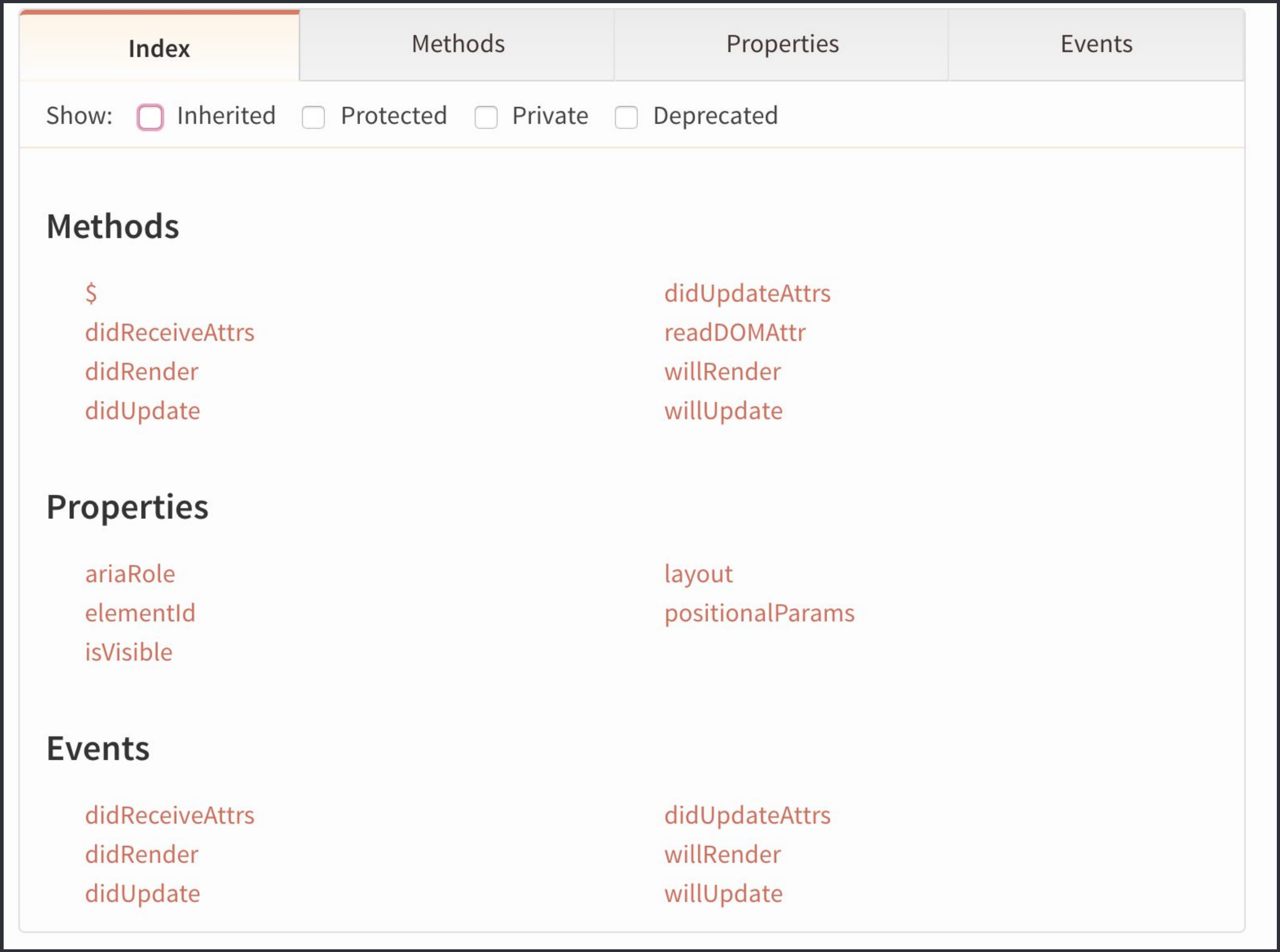
Classic component: How did it look?

```
// app/components/x-button.hbs
import Component from '@ember/component';

export default Component.extend({
  tagName: 'button',
  className: ['btn'],
});

{{!-- app/templates/components/x-dialog.hbs --}}
{#modal-dialog}}
  {{#x-button
    text="Submit"
    onclick=(action "submit")
    }}
{{/modal-dialog}}
```

Classic component

- 13 Standard lifecycle hooks
- 29 Event handlers
- 9 element/element customization properties
- 21 standard framework functions

Classic component



Index | Methods | Properties | Events

Show: ☐ Inherited ☐ Protected ☐ Private ☐ Deprecated

Methods

- $
- didReceiveAttrs
- didRender
- didUpdate
- didUpdateAttrs
- readDOMAttr
- willRender
- willUpdate

Properties

- ariaRole
- elementId
- isVisible
- layout
- positionalParams

Events

- didReceiveAttrs
- didRender
- didUpdate
- didUpdateAttrs
- willRender
- willUpdate

https://api.emberjs.com/ember/3.10/classes/Component

Glimmer Component

- Just 2 lifecycle hooks:
  - constructor
  - willDestroy
- Only 3 properties:
  - args
  - isDestroying,
  - isDestroyed
- Outer HTML Semantics
- Namespaced Arguments
- Unidirectional Dataflow
- Stateless Template-Only Components

Glimmer Component

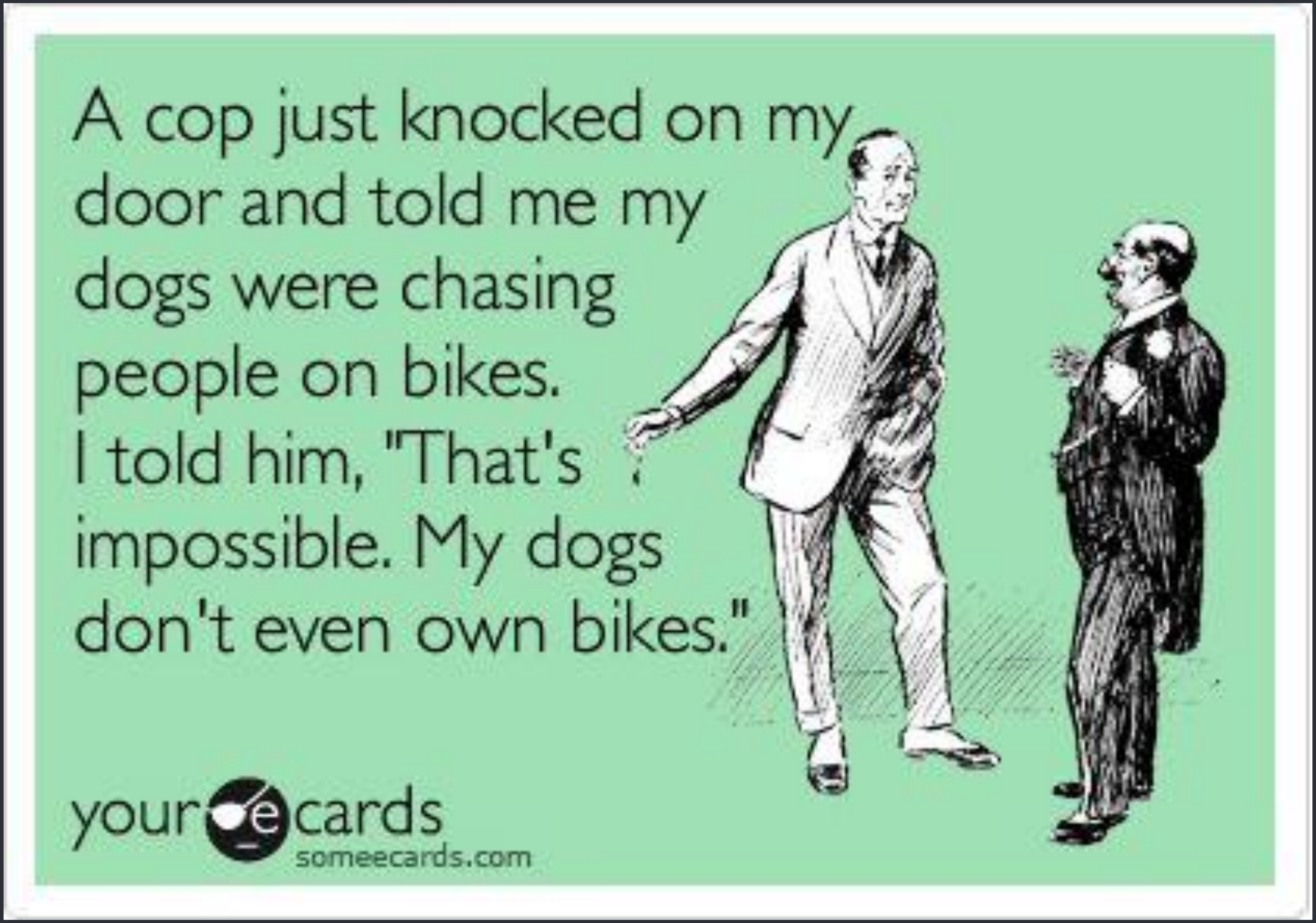
```
// app/components/hello-button.js
export default class HelloButton extends Component {
    constructor() {
        super(...arguments);
    }
}

<!-- app/templates/components/hello-button.hbs -->
<button class="btn" role="button">
    Hello, world!
</button>
```

Glimmer Component

```
{{!-- src/ui/components/notes/note-input/template.hbs --}}
<input
  {{did-insert @focusInput}}
  type="text"
  onkeyup={{@addNote}}
  ...attributes
 />

{{!-- Usage --}}
<Notes::NoteInput
  @addNote={{action addNote}}
  @focusInput={{action focusInput}}
  name="add-note"
  class="input is-large"
  value={{inputValue}}
  aria-label="add note here"
  disabled={{if isLoading "disabled" ""}}
  />
```

Modifiers

Element modifiers or Render Modifiers


A cop just knocked on my door and told me my dogs were chasing people on bikes.
I told him, "That's impossible. My dogs don't even own bikes."

Modifiers

- similar to Handlebars helpers
- functions or classes
- use with {{double-curlies}}.
- applied directly to elements:

Modifiers: addons

```
{{!-- `@ember/render-modifiers` --}}
an official Ember addon that provides
{{did-insert}}
{{did-update}}
{{will-destroy}}

{{!-- `ember-on-modifier` --}}
<button {{on "click" this.onClick}}>
  Click me!
</button>

{{!-- `ember-ref-modifier` --}}
<button {{ref this "button"}} data-name="foo">
  Click me!
</button>

{{this.button.dataset.name}} >> "foo"
```

Modifiers: usage

```
//function defined elsewhere
focusInput(element) {
    element.focus();
}

{{!-- src/ui/routes/index/template.hbs --}}
<NoteInput
    @focusInput={{action focusInput}}
/>

{{!-- src/ui/components/note-input/template.hbs --}}
<input
  {{did-insert @focusInput}}
  type="text"
/>
```

Native Classes & Decorators

Native Classes & Decorators

Native Classes

```
//BEFORE
import Component from '@ember/component';

export default Component.extend({
  init() {
    this._super(...arguments);
  },
});

//AFTER
import Component from '@ember/component';

export default class XButtonComponent extends Component {
    constructor() {
        super(...arguments);
    }
}
```

Decorators

- Declarative transformation of value/method
- Minimal & readable code
- Enabled by default using ember-cli-babel ⩾ v7.7.3
- ember-decorator addon: an experiment

Ember
Decorators
Useful decorators for Ember applications.

Decorators: Usage

```
import Controller from '@ember/controller';
import { action, computed } from '@ember/object';
import { alias } from '@ember/object/computed';
import { tracked } from '@glimmer/tracking';

export default class IndexController extends Controller {
  @alias('model') notes;
  @tracked isLoading = false;

  @computed('notes.@each.text')
  get allTags() { //more code }

  @action
  addNote(event) { //more code }
}
```

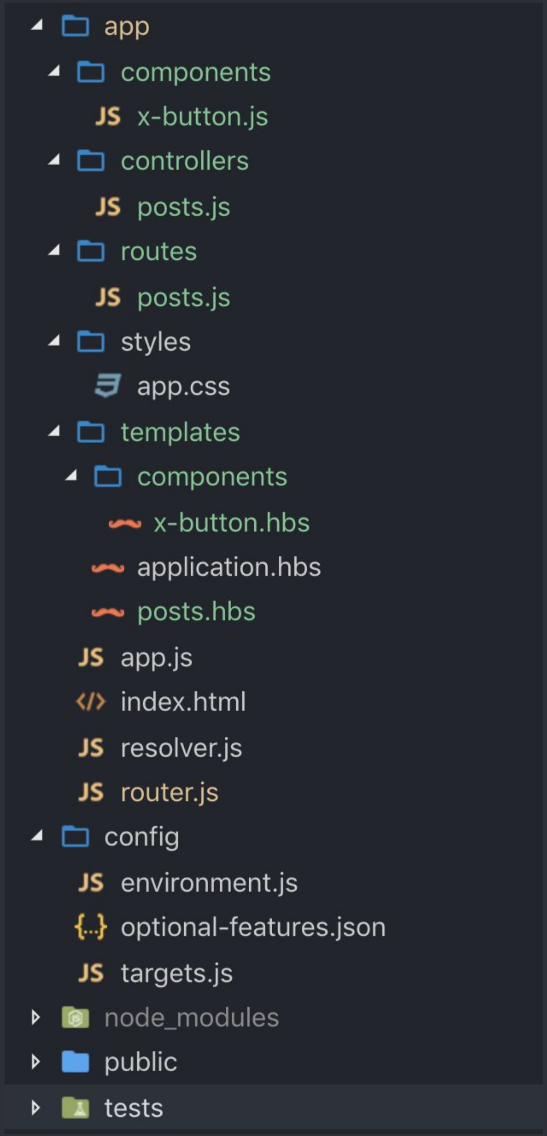
What’s next…

Or what after Octane…

Module Unification aka MU

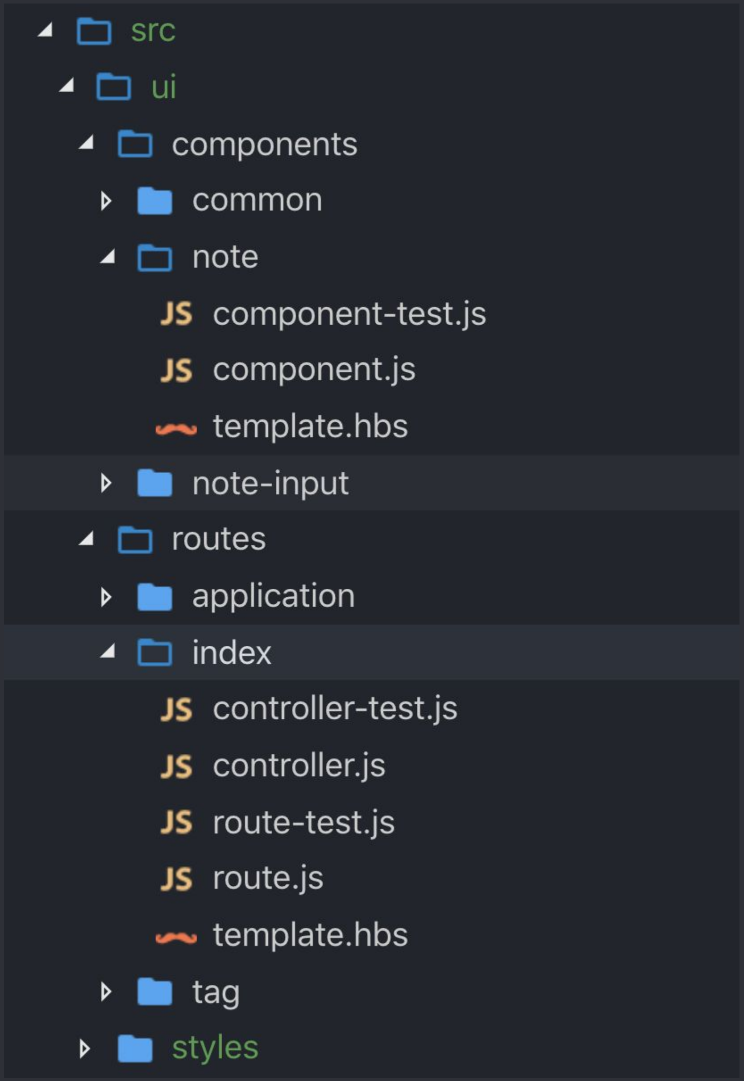
Before MU: Project structure

```
#create app
ember new my-app

#create route
ember g route posts

#create controller
ember g controller posts

#create component
ember g component x-button
```



```
app
  components
    x-button.js
  controllers
    posts.js
  routes
    posts.js
  styles
    app.css
  templates
    components
      x-button.hbs
    application.hbs
    posts.hbs
  app.js
  index.html
  resolver.js
  router.js
config
  environment.js
  optional-features.json
  targets.js
node_modules
public
tests
```

After MU : Project structure



```
src
  ui
    components
      common
      note
        component-test.js
        component.js
        template.hbs
      note-input
    routes
      application
      index
        controller-test.js
        controller.js
        route-test.js
        route.js
        template.hbs
      tag
    styles
```

After MU : Project structure

- Application code in ~~app~~ **src/ui**
- Co-located
  - route, controller, template, tests
  - component, template, tests
- single word component filename support

Upcoming

- Private/scoped components
- Declarative imports in templates

Demo Time

https://github.com/kushdilip/ember-octane-life-log

Demo steps

- Feature walkthrough
- Correlate Octane features and code walkthrough
- Create a new tag component with number of notes
- Push and demo on netlify link

Thanks!

ANY QUESTIONS?