

Hands-on with Ember Octane

A practical look at the
latest from Ember.js



Timeline

August 5th

No new features
LTS Candidate
3.13 Beta

3.12

September 16th

3.12 becomes LTS
3.14 Beta

First release that contains
all of Octane's features

3.13

First release of Ember that
defaults to Octane

Ember Inspector

3.14



Classic vs. Octane

Application Template Wrapper

Classic Mode

```

<body class="ember-application">
  <div class="ember-view" id="ember175">

    <!--your app goes here -->

  </div>
</body>

```

Octane Mode

```

<body class="ember-application">

  <!--your app goes here -->

</body>

```

```
$ ember feature:disable application-template-wrapper
```

jQuery

Classic Mode

- jQuery is included by default.
- Ember components have a `this.$` method.

Octane Mode

- jQuery is not included by default

```
$ ember feature:disable jquery-integration
```

Template-only Components

Classic Mode

- Template-only components get an implicit component class.

Octane Mode

- Template-only components have **no** (implicit or otherwise) component class.

```
$ ember feature:enable template-only-glimmer-component
```

Observers

Classic Mode

- Observers fire synchronously by default

Octane Mode

- Observer fire asynchronously by default

```
$ ember feature:enable default-async-observers
```

The background features a dark gray field with three overlapping circles of varying shades of blue. A horizontal white band cuts across the middle of the circles. The text 'Idiom Changes' is centered within this white band.

Idiom Changes

Ember vs. Standard Class Syntax

Classic Mode Idiom

- subclass from framework classes

```
Route.extend({ ... })
```

```
Controller.extend({ ... })
```

Octane Mode Idiom

- subclass from framework classes using JavaScript class syntax

```
class extends Route {...}
```

```
class extends Controller {...}
```

Migration Guide: <https://tinyurl.com/octane-migration>

Computed vs. @tracked

Classic Mode Idiom

- State that could influence the output ("mutable state") is marked using `.set` or `Ember.set` and use computed from `@ember/object` to describe derived state. Computed properties enumerate their dependent keys explicitly.

Octane Mode Idiom

- State that could influence the output is marked as `@tracked`. No other annotations are needed.

Migration Guide: <https://tinyurl.com/octane-migration>

Components

Classic Mode Idiom

- Components subclass from `@ember/component`. They use APIs like `classNameBinding` to configure the root element, and use lifecycle hooks on the component class to interact with the DOM.

Octane Mode Idiom

- Components subclass from `@glimmer/component`. They describe all elements, including the root element, in the template. They use modifiers to interact with the DOM.

Migration Guide: <https://tinyurl.com/octane-migration>

{{action}} vs. {{on}}

Classic Mode Idiom

- To handle events on a component's root element, create a method on the component that corresponds to the event. To handle events on another element, use the `{{action}}` helper and put the action in the actions hash in your component.

Octane Mode Idiom

- To handle events on an element, use the `{{on}}` helper.

Migration Guide: <https://tinyurl.com/octane-migration>

Curly Braces vs. Angle Brackets

Classic Mode Idiom

- Components are invoked using `{{component-name}}` and `{{#component-name}}` syntax.

Octane Mode Idiom

- You invoke components using `<ComponentName>` syntax.

Migration Guide: <https://tinyurl.com/octane-migration>

Implicit vs. required this

Classic Mode Idiom

- You can refer to properties on the component as `{{propertyName}}`

Octane Mode Idiom

- References to properties on a component must be `{{this.propertyName}}`

Migration Guide: <https://tinyurl.com/octane-migration>

File Layout

Classic Mode Idiom

- A component's files are in two places:

app/components/component-name.js

app/templates/components/component-name.hbs

Octane Mode Idiom

- A component's files are in the same place:

app/components/component-name.js

app/components/component-name.hbs

Migration Guide: <https://tinyurl.com/octane-migration>



Migration

A reasonable approach

Recommended Migration Order

1. Migrate away from observers before migrating to `@tracked`

Recommended Migration Order

2. The template codemods are generally easier and more automatable, so they should be done first:

- [ember-angle-brackets-codemod](#)
- [ember-no-implicit-this-codemod](#)

Recommended Migration Order

3. Classic Component Changes

Step 3.1: remove use of implicit `this`

Recommended Migration Order

3. Classic Component Changes

Step 3.2: migrate to `tagName: ''`

Recommended Migration Order

Step 3.2: migrate to `tagName:''`

1. wrap your component in a root element
2. Change `classNames` to `class="..."`
3. Change `classNameBindings` to `class={{...}}`
4. Change `attributeBindings` becomes `attr={{...}}`
5. Change methods like `click` to `{{on "click"}}`
6. move DOM manipulation logic to modifiers (or at least [ember-render-modifiers](#))

Recommended Migration Order

3. Classic Component Changes

Step 3.3: migrate `{{action}}` and the actions hash to `{{on}}` in the template and `@action` in the component.js file

Recommended Migration Order

3. Classic Component Changes

Step 3.4: migrate to native class syntax

Recommended Migration Order

3. Classic Component Changes

Step 5: migrate `.set` and computed properties to `@tracked`

Codemods



Native class
codemod



Angle bracket
codemod



No implicit
this codemod

<https://github.com/ember-codemods>

Reference Materials

- <https://ember-learn.github.io/ember-octane-vs-classic-cheat-sheet/>
- <https://codingitwrong.com/2019/07/23/ember-component-cheat-sheet.html>

Extra Support

- ember-template-lint
 - Updates for Octane support
 - New a11y rules
- #topic-octane-migration





Thank you!