
Re:ember

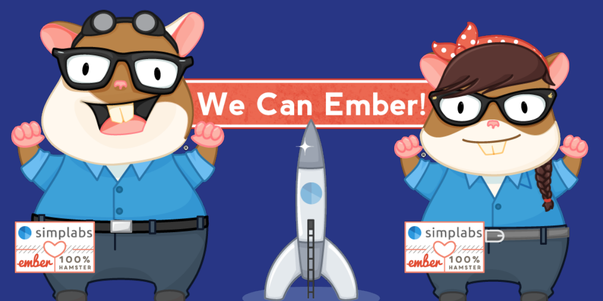
Ember.js Octane

A fresh look at the framework for ambitious web applications

Florian Pichler

Consultant for Design & Technology
simplabs GmbH

Developer, Designer, Baker, Maker
Bad at naming things



Disclaimer N°1

I work with Ember.js on a daily basis. I work for a company sponsoring the development and community of Ember.js and I worked on the current website design for Ember.js.

~~There might be some bias.~~

There is most definitely some bias.

In this talk

1. What is Ember.js and why should I care?
2. What is Octane and how can I use it?
3. Demo time and a host struggling with live code

Also included are two additional disclaimers and eight hamsters

Disclaimer N°2

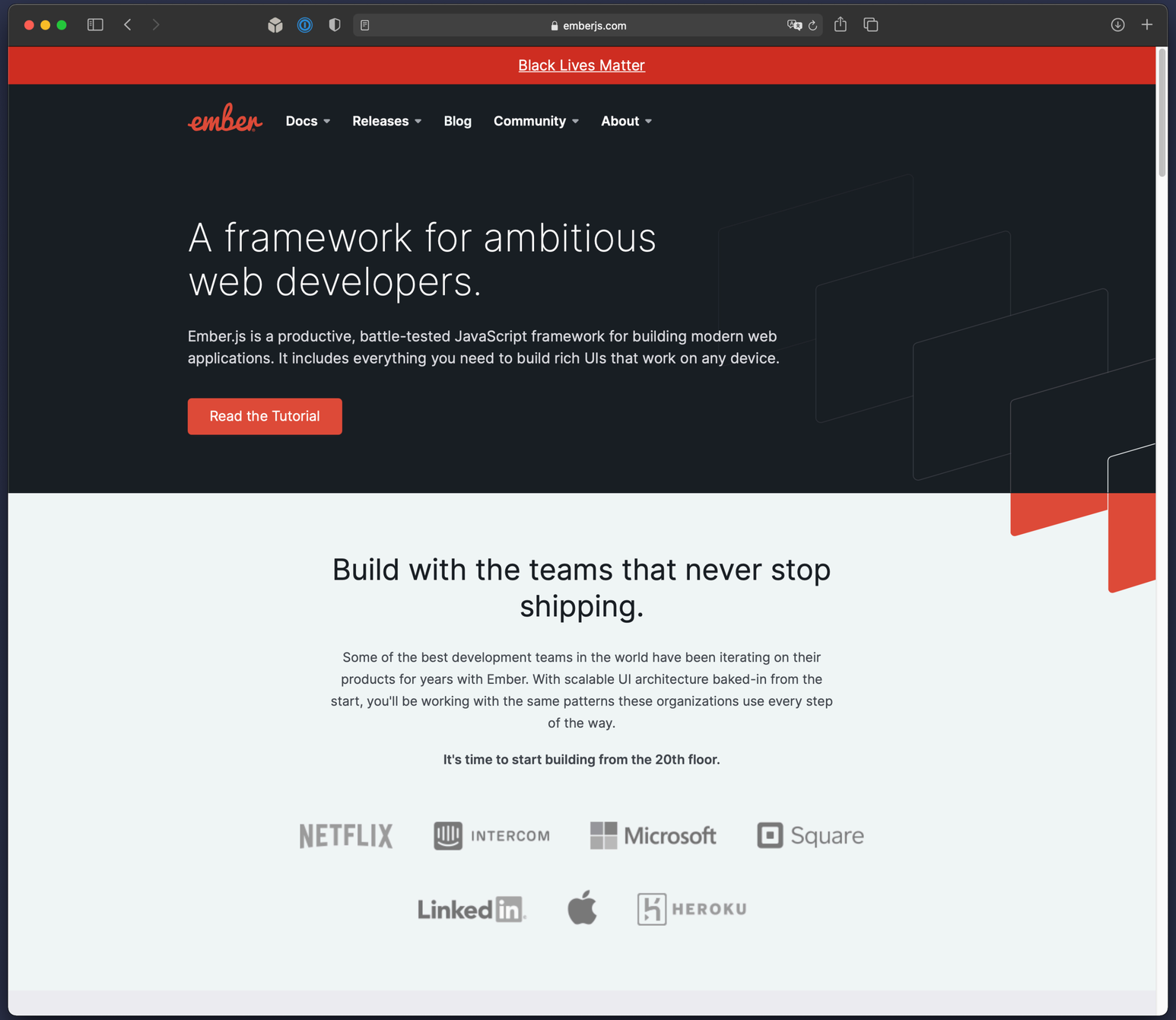
This talk will not try to provide a comparison to view layers such as React or Vue.

Glimmer – the view layer of Ember.js – only exists as a standalone project in theory or if you are LinkedIn.

Okay, this needs more explanation.

What is ember®?

JavaScript framework for single page applications


emberjs.com
Black Lives Matter
ember
Docs
Releases
Blog
Community
About
A framework for ambitious web developers.
Ember.js is a productive, battle-tested JavaScript framework for building modern web applications. It includes everything you need to build rich UIs that work on any device.
Read the Tutorial
Build with the teams that never stop shipping.
Some of the best development teams in the world have been iterating on their products for years with Ember. With scalable UI architecture baked-in from the start, you'll be working with the same patterns these organizations use every step of the way.
It's time to start building from the 20th floor.
NETFLIX
INTERCOM
Microsoft
Square
LinkedIn
HEROKU

Created, organized
and maintained
by people, not companies.

(opinionated)
batteries included

→ *View Layer*
→ *Routing*
→ *State management*
→ *Addons*
→ *Tooling*
→ *Testing harness*

View layer: Glimmer

Reactive UI components

Handlebars (HTMLBars) templates

→ making it easy to start with plain HTML

→ classNames="" is simply class=""

→ separation of concerns

It's blazing fast, too. Op-codes and data diffing.

Routing: Ember Router

URL first routing

→ `/your-app/will/have/good/urls`

→ `?query-params`

→ `/dynamic/:id/segments`

Automatic handling for loading & error states

Provides hooks to load necessary data

State Management: Services & Injections

Services: Long lived state objects which can abstract behavior (data store, session handling, notifications, ...)

Dependency Injection: Alternative to nesting and requiring to pass global state or abstractions into components manually.

Ember Data: Manage records instead of endpoints. JSON:API compliant & included by default.

Addons

ember-concurrency: Powerful abstractions
to handle asynchronous and recurring tasks

ember-css-modules: CSS that is scoped to your component

ember-simple-auth: Handle user sessions

ember-intl: Translate your app using browser APIs

ember-truth-helpers: Boolean logic for templates

Tooling

ember-cli

→ development server & production bundler

→ blueprints to generate files for your application

→ scaffolding for new apps & addons

ember-inspector

→ browser addon to debug Ember.js apps and their data

Testing harness

Ships with QUnit, other options available.

Integrated support and autogenerated stubs for:

→ Unit tests (functions and classes)

→ Integration tests (single components)

→ Acceptance tests (workflows)

Integrated test runner which tests in actual browsers.

Bonus round

Opinionated project structure and architecture make it easy to jump into most unknown Ember.js projects.

Long Term Support: Ember.js supports long lived projects through LTS releases, strict Semantic Versioning and gentle or even automatic upgrade paths.

Great community: Discord, Forum. Helpful and nice people

So what is *Ember Octane*?

Cool new stuff

Cool ~~new~~ stuff

The first edition release of Ember.js

Editions

A way to talk about features as part of Ember's release cycle without breaking things.

Features that will be enabled by default for new apps and migration paths for existing ones.

A foundational improvement to the way you use Ember.

Ember Octane

- Everything was added in non-breaking releases
- Existing apps continue to work
- Features can be enabled gradually
- Default for new apps
- Already shipped and tested

Things retiring

- jQuery (at last)
- Non-native classes
- Computed properties and observers
- Curly component invocation
- Wrapping containers for components

Core Features

- Native JavaScript classes (+ Decorators)
- Tracked properties
- Glimmer components
- <AngleBracket /> syntax
- Element Modifiers
- Documentation & tooling

Native JavaScript
Classes

Native JavaScript classes (+ Decorators)

- Replacing the old Ember.Object syntax
- Native Getters & Setters
- No more .extend(), .create()
- @decorators for injections and actions

```
// Before: Custom Objects

export default Component.extend({
  bar: 'baz',

  init() {
    this._super(...arguments);
  },

  bat: computed(function() {
    return this.get('bar');
  }),

  method() {}
});
```

```
// After: Native classes

export default class Foo extends Component {
  bar = 'baz';

  constructor() {
    super(...arguments);
  }

  get bat() {
    return this.bar;
  }

  method() {}
}
```

Tracked Properties

Tracked Properties

- Backwards compatible to the old computed properties
- Basis for template reactivity
- Marks properties that trigger template updates
- @tracked decorator
- Autotracking for component arguments

Glimmer Components

Glimmer Components

- Start as template only
- Native backing classes
- Less involved life cycle
- Autotracked `this.args` to reach for component arguments in your class

```
// Before: .set()

export default Component.extend({
  show: false,

  actions: {
    toggle() {
      this.set('show', !this.show);
    }
  }
});
```

```
// After: @tracked

export default class Foo extends Component {
  @tracked show = false;

  @action
  toggle() {
    this.show = !this.show;
  }
}
```

Improved Templating

Improved Templating

- <AngleBracket /> syntax
- @attr to discern component and HTML attributes
- Element <div {{modifiers}}>

→ Better distinction between basic HTML, component invocations and template expressions

```
{{#layout-container class="foo"}}
  <button {{action "showCanvas"}}>
    {{user-profile
      image="http://…"
      name="Jane"
      classNames="special"
    }}
  </button>

  {{#if this.showCanvas}}
    <canvas class="find-me">
    </canvas>
  {{/if}}
{{/layout-container}}
```

```
<LayoutContainer class="foo">
  <button {{on "click" this.showCanvas}}>
    <UserProfile
      @image="http://…"
      @name="Jane"
      class="special"
    />
  </button>

  {{#if this.showCanvas}}
    <canvas {{did-insert this.render}}>
    </canvas>
  {{/if}}
</LayoutContainer>
```

Available ~~now~~

Available since
December 2019

Demo time?

Disclaimer N°3

The following demo shows interesting aspects of a single page application built with Ember.js. I'll highlight new features of Octane along the way. I have some ground to cover, so I'll go fast.

Please tell me to slow down if necessary. Hold fundamental questions and remarks until after the talk. There is no more content after the demo.

Demo time!

Thanks ❤

florianpichler.de
@pichfl
fp@ylk.gd
florian.pichler@simplabs.com

Further reading #1

- https://emberjs.com
- https://emberjs.com/editions/octane/
- https://simplabs.com/blog/2020/10/05/simpler-and-more-powerful-components-in-ember-octane-with-glimmer-components/
- https://www.pzuraq.com/what-is-reactivity/

Further reading #2

- https://nullvoxpopuli.com/2020-08-08-how-does-di-work
- https://www.pzuraq.com/comparing-ember-octane-and-react/
- https://www.effective-ember.com/blog/react-hooks-and-ember/