Ember Octane HI 1-Pager

October 27, 2020 | Chris Krycho

Ember Octane is a significant modernization and simplification of writing Ember apps, which comes with significant performance improvements as well as developer experience benefits while maintaining backward compatibility. Fully migrating to Octane is critical to unlocking future improvements to the framework in terms of both performance and developer experience.

feature	**Ember Classic**	**Ember Octane**
classes	custom object model[1]	native classes
templates	**Extremely dynamic:** • Not codemoddable • Difficult to distinguish distinct constructs visually • *O(n)* lookup costs for n construct types • Subjectively: feels dated	**Fully statically-analyzable:** • Codemoddable • Distinct constructs are visually distinct • *O(1)* lookup costs for every construct • Subjectively: feels modern[2]
components	**Ember Component** • uses custom object model • 13 lifecycle hooks with complex life cycle • two-way binding • coupled DOM and state management	**Glimmer Component** • simple native JS class • 2 lifecycle hooks: one each for setup and teardown • one-way data flow • decoupled DOM and state management
reactivity	**Observer-based system:** • Caches everything • Push-based • Annotate *consumers*	**Autotracking:**[3] • Caching on demand (PAYGo) • Pull-based • Annotate *producers*

These changes combine to produce significant improvements in many categories, including render performance, the amount of code written, code complexity, the estimated number of errors in implementation, and maintainability rating.[4] Moreover, the shift to statically-analyzable constructs enables future performance optimizations and developer experience improvements as well as easing future migrations.

---

[1] Ember Classic's custom object model was one of the key inputs to the design of JavaScript's native classes.

[2] Ember Octane components' syntax feels similar to React, Vue, Svelte, and other contemporary frameworks

[3] Autotracking uses a cutting-edge CS idea, incremental computation. See [1], [2], [3], and [4] for an introduction to the underpinnings of autotracking and [5] and [6] for a look at autotracking in practice.

[4] [We ran Plato and TracerBench on the RealWorld Ember app pre- and post-Octane conversion.]