**A Real-World Comparison of Front-End Frameworks with Benchmarks**

Over the last couple of years we have seen an explosion of front-end frameworks. Each one of them is more than capable of building great web applications. So how do you compare and decide which one to use for your next project?

First of all, to make a meaningful comparison we need a few things:

1. **Real World App** - Something more than a “todo”. Usually “todos” don’t convey knowledge & perspective to actually build *real* applications.
2. **Standardized**- A project that conforms to certain rules. Hosted at the same place, provides a back-end API, static markup, styles, and spec.
3. **Written by an expert**- A consistent, real world project, that ideally an expert in that technology would have built. This is true, at least most of the time (see below).

So how do we get such a project? The good news is that [Eric Simons](https://medium.com/@ericsimons) already created a [RealWorld](https://github.com/gothinkster/realworld" \t "_blank) project. It’s a clone of the Medium blogging platform. Each implementation of this project uses the same HTML structure, CSS, and API spec, but a different library/framework. When it comes to expert knowledge it’s true most of the time. I wrote an implementation in ClojureScript and [re-frame](https://github.com/Day8/re-frame) and I don’t consider myself an expert. In my defense an expert reviewed my code - thanks [Daniel Compton](https://medium.com/@danielcompton).

Now we have a baseline spec, we need a standard set of tests/metrics to compare them.

1. **Performance.**How long does this App take to show content and become usable?
2. **Size.**How big is the App? We will only compare the size of the compiled JavaScript. The CSS is common to all variants, and is downloaded from a CDN (Content Delivery Network). The HTML is common to all variants too. All technologies compile or transpile to JavaScript, thus we only size this file.
3. **Lines of Code.**How many lines of code did the author need to create RealWorld app based on spec? To be fair some apps have a bit more bells and whistles, but it should not have a significant impact. The only folder we quantify is src/ in each app.

At the time of writing (Dec 2017) the RealWorld project is available in the following frameworks:

* [React / Redux](https://github.com/gothinkster/react-redux-realworld-example-app)
* [Elm](https://github.com/rtfeldman/elm-spa-example)
* [Angular 4+](https://github.com/gothinkster/angular-realworld-example-app)
* [Angular 1.5+](https://github.com/gothinkster/angularjs-realworld-example-app)
* [React / MobX](https://github.com/gothinkster/react-mobx-realworld-example-app)
* [Crizmas MVC](https://github.com/gothinkster/crizmas-mvc-realworld-example-app)
* [CLSJ Keechma](https://github.com/gothinkster/clojurescript-keechma-realworld-example-app)
* [AppRun](https://github.com/gothinkster/apprun-realworld-example-app)
* [CLJS re-frame](https://github.com/jacekschae/conduit) (This is the one I did. It’s not yet listed at RealWorld Project).

**Metric #1: Performance**

[First meaningful paint](https://developers.google.com/web/tools/lighthouse/audits/first-meaningful-paint) test with [Lighthouse Audit](https://developers.google.com/web/tools/lighthouse/) that ships with Chrome.

The sooner you paint, the better the experience for the person who is using the App. Lighthouse also measures [First interactive](https://developers.google.com/web/tools/lighthouse/audits/first-interactive), but this was almost identical for most apps.

Andrea Passwater

Poet, powerlifter, cactus enthusiast, tech writer

Dec 19

Sorry I am a real noob at coding but would really love to learn—

Freedom From Work

So! I am going to give you the “best of my understanding” answers to these questions, and hopefully if I say something wrong the internet hivemind will save us from ignorant bliss.

I installed the serverless framework on my own computer. It’s a tool that lets you interface with different serverless providers(like AWS Lambda, which is what used). The “serverless” part is kind of a misnomer-it really just means you don’t have to provision or manage the servers because a magic cloud apparatus does it for you. Or something.

Anyway! The end result is that you can basically just type ***sls deploy*** into your terminal and your code goes to live on a server where everyone in the world can use it. So yep, it does live in the cloud.

To get ready for the project, I installed Atom, homebrew, the serverless framework, and the awscli. I think that’s it! Also Node if you don’t have it already installed.