famo.us

FREQUENTLY ASKED QUESTIONS

What is Famo.us?

Famo.us is a free, open source JavaScript platform that enables you to build smooth, complex UIs for any screen.

It is the only JavaScript framework that includes an open source 3D layout engine fully integrated with a 3D physics animation engine that can render to DOM, Canvas, or WebGL.

What inspires Famo.us?

The perfection of Apple, the way Google is empathetic to the plight of developers, and the way Mozilla helps us understand the power and value of open source development to our community.

Is Famo.us for me?

Famo.us is for everyone. Whether you're a seasoned developer expanding your skills, a designer looking to bring your ideas to life, or just learning to code, we're building Famo.us for you. If you're new to JavaScript development, we have lots of learning resources to help you get started.

Is Famo.us free?

Yes.

Is Famo.us open source?

Yes. Famo.us is available under MPLv2 (Mozilla Public License v2).

How do you make money?

Building apps is FREE. Anything associated with building an app—for example, the Famo.us platform and any IDE we might build in the future—will be free.

Cloud services are PAID but OPTIONAL. Famo.us will provide cloud services associated with running and deploying apps—things like wrapping, hosting, analytics, monitoring, A/B testing, and other related services. These will be optional cloud services that you can add to your account. If you don't want to use our services, you can always choose your own.

What makes Famo.us unique?

Famo.us is the only JavaScript framework to have our own open source 3D layout engine fully integrated with a 3D physics animation engine built in pure JavaScript that can render to DOM, Canvas, or WebGL.

Why is having your own layout engine important?

The short answer: performance.

The long answer:
The layout engine in a browser was originally designed to render text documents with links, so it's not hard to see why it has a difficult job trying to render apps and games. It's simply the wrong tool for the job.

Creating an iOS native app that competes in today's world often requires an interface with a complexity level that cannot be reasonably managed using regular HTML5 or native Android. The result is that for users, HTML5 apps feel jittery, slow, and definitively NOT native. The result for developers is that app development is brittle.

The Famo.us layout engine was built from the ground up to render apps and games of today. It is:

- **Powerful**. Famo.us exposes powerful APIs to leverage the performance of HTML5 and guarantee 60FPS performance for motion and dynamic data.
- **Portable**. Famo.us requires no download, plugin. or install. It comes inside the payload of the app.
- **Capable**. Famo.us can handle app complexity on par with or better than native iOS and better than Android.
- **Accessible**. Famo.us is easier to learn than native development.
- **Useful**. Famo.us is cross-platform.
- **Efficient**. Famo.us can build the same app, with the same performance, using half the code you need in native.

Why is having your own 3D physics engine important?

The quality of animation directly impacts the interface and thus the experience of the product. The best animation engines in the world are console gaming engines, and they all use 3D physics to generate their animations.

This is how the mobile app ecosystem stacks up in comparison:

- **Browsers** use CSS3 as their animation engine. CSS3 technology and techniques are roughly equivalent to what gaming animation used in the mid 1980s.
- **Native iOS** uses a compiled 2D physics engine called Box2D which is not integrated with its layout engine. It is roughly based on 2007 technology and works with native only.
- **Famo.us** uses a JavaScript 3D physics engine that is fully integrated with our 3D layout engine and is based on 2014 technology. It works with DOM, Canvas, and WebGL and is even abstracted to work with OpenGL, DirectX, or anything that renders to a GPU.

Aren't you afraid of Google and Apple using your concepts?

At first, it seems like something to be worried about—but in reality, we hope they do. It might surprise you, but we've met the browser companies and we freely give them our concepts and ideas. We see JavaScript as a new way to start a conversation. If the browsers embed any of our techniques, then we can simply take advantage of those improvements and make performance even better for Famo.us apps.

We will never stop inventing.

Does Famo.us integrate with third-party libraries?

Yes. Our goal is that anything we build is the very best, and we also always want to be inclusive to the larger development community.

The Famo.us/Angular integration is now publicly available at http://famo.us/angular.

Famo.us can also be integrated with modern MVCs including Ember.js, and Backbone.js. We work with Meteor.js and we work very well with Firebase. We can work directly with jQuery as long as you don't do any DOM manipulation in it.

How do I learn Famo.us?

We are committed to training. There are several ways to learn Famo.us:

Reference documentation. Many senior engineers prefer reference documentation as their primary way to learn something new. Full reference documentation is available at http://famo.us/docs.

Demo gallery. Many people enjoy learning through examples. Our demo gallery, available at http://famo.us/demos, includes live UI examples and source code from several of the most beautiful and difficult iOS apps out there. This library of examples will continue expanding over time and is a great way to get inspired and learn.

Famo.us University. The idea behind Famo.us University is to immediately immerse you in live code so you get to your aha! moment as quickly as possible.

Courses include full tutorials on selected animation demos as well as modularized examples for those who wish to move faster in more discrete areas.

Public courses in Famo.us University will always be free. At first, Famo.us itself will populate the courses. In the long run, we hope to work with the community to generate a wide variety of courses. The idea is to enable anyone to have access to free, self-serve education.

What is Famo.us University?

Famo.us University was inspired by tools like Code Academy and CodeSchool, but we did it in our own way.

The primary difference is that in Famo.us University, you get a live coding environment with live updates to a viewport—so you are really coding. The lessons are presented in a three-panel tool that includes a description, live editor, and viewport.

How do I install Famo.us?

Simply click the Download button at the top of any page on the Famo.us website.

What can I build with Famo.us?

We are providing official support for:

- Mobile apps on phones and tablets
- iOS, Android, FirefoxOS, and Kindle devices (and very soon Microsoft devices)

However, if you are an advanced engineer, there's really no limit. In the future we will make it easier to build ads, widgets, and data visualizations, as well as interfaces for desktop browsers, televisions, and other devices with screens.

What devices are supported?

Mobile phones and tablets of the iOS 4.3+, Android JellyBean+, Kindle, and FirefoxOS varieties.

What impact does Famo.us have on battery life?

In 2013, Apple VP of Engineering Craig Federighi showed Famo.us in his keynote at WWDC. He talked about technologies like Famo.us being the future but showed just how much CPU Famo.us used. Since then, we have worked diligently to reduce our CPU usage in both active and passive states to reduce load on the battery.

Here is the history of our progress. We used our original periodic table demo as the initial baseline, but now we use Twitter built on the different versions of Famo.us as our measuring stick.

As of December 2012

Active CPU usage: 36%
Passive CPU usage: 27%

As of March 2014

Active CPU usage: 7%
Passive CPU usage: 1%

How we've made the improvements:

- Rigorous tuning of the critical sections of code which execute every frame
- More aggressive per-branch optimizations
- General improvements in elegance have allowed the code to be simpler and thus run in less time

What prevents some evil company from forking you?

Nothing, but it would be pretty uncool on their part. Our MPLv2 license protects us a little bit.

Why should Angular developers use Famo.us?

If you want to tap into Famo.us's cross-platform rendering power inside a new or existing Angular app, or if you'd like to approach Famo.us with a familiar approach and workflow (i.e. with AngularJS,) then Famo.us/Angular is a great choice. See http://famo.us/angular for more information.

How do I do animations with Famo.us/Angular?

There are two primary ways to perform animations in F/A: declaratively and imperatively. The declarative way uses a timeline model, allowing you to declare how several different attributes over several different elements should respond to a single timeline. The imperative way uses Famo.us Transitionable objects, allowing you to specify animations in a more vanilla-Famo.us way. See http://famo.us/angular for more information.

More information is available in the docs, and there will be more detail on these two approaches in the forthcoming Famo.us University course.

What can native do that Famo.us cannot?

The two primary weaknesses of Famo.us are due to the fact that we rely on JavaScript and the browser itself.

1. JavaScript is not multi-threaded.

2. Garbage collection is not elegantly handled in browsers but is getting better.

3. We don't have WebGL integrated in our scene graph yet (we need WebGL support turned on by default in the browser to do this, so we're waiting on Safari)—which means we have pretty limited control over raster (changing colors, using non-rectangular geometries, and changing fundamental geometries of renderables).

The reality is that there are certain UI features that require more than one thread and proper garbage collection to be handled elegantly. And in those cases, Famo.us will not be as good as native.

While there are some tricks, none are as good as the options afforded to iOS.

What's a native app that Famo.us cannot do yet?

A good example of an app that we are just shy of being able to do at pure equivalency to native is Facebook itself. Oddly, while Facebook Paper looks harder to do, it's actually quite doable in Famo.us.

It's the newsfeed that's the killer.

The newsfeed scrollview utilizes three threads simultaneously in native iOS (one for data, one for measurement, and one for rendering) to achieve the beautifully smooth scrolling that you see in the Facebook app. Kudos to the Facebook team for doing an astounding job.

We've re-created the exact conditions of the Facebook newsfeed using Famo.us, and the result we get is that we can nearly match 60FPS with infinite scroll while adding data. But because of the single-threaded nature of JavaScript and garbage collection, we get an annoying tick that you don't see in the native app.

We expect to resolve this soon, though. That's why we say we are just shy of being able to do it.

You claim you fix HTML5. What makes you so great when so many others failed?

Think about how many times an industry saw failures before someone finally broke through and succeeded.

Facebook wasn't the first social network. Google wasn't the first search engine.

To solve big problems, it sometimes takes a few tries before someone sees the whole picture correctly.

What can I expect on each browser and device?

You should expect the following performance conditions, assuming you coded your app correctly:

BRAND	**DEVICE**	**OS**	**BROWSER**	**PERF**
iOS	iPhone 5+ iPad 2+	iOS 6+	Safari 6+	60FPS
Android	2013+ phones tablets	JellyBean +	Chrome 31+	60FPS
Android	2013+ phones tablets	JellyBean +	Chrome 30+	40-60FPS
Android	2013+ phones tablets	JellyBean +	Chrome 24 -> 29	30-60FPS
Android	2013+ phones tablets	pre JellyBean	Any	Unsupported
Microsoft	phones tablets	any	any	preserve-3d bug > Win 8* target: 60FPS
Kindle	2014+	any	any	50-60FPS
Firefox	Keon phones	FirefoxOS	Firefox	30FPS**

Tizen	2013+ phone	Tizen	Tizen	Works but unsupported
WebOS	any	WebOS	WebOS	Works but unsupported

* We are working diligently on resolving this with Microsoft.
** 30FPS is equivalent to embedded apps on this device.
Performance depends heavily on the specific device used.

How is the Famo.us physics engine any different than all of the others?

There are many physics engines. We recommend that you check to see how many have all of these features:

1. Renders to DOM, Canvas, and WebGL
2. Supports building your own physics primitives
3. Integrates directly with the layout engine
4. Built for mobile first

You'll see it's a list of one.

Is Famo.us subject to cross-browser quirks?

Unlike many other HTML5 platforms, we view it as our job to create a stable environment for you. You shouldn't have to be memorizing the weird esoteric rules of each browser.

If you experience a cross-browser quirk, it's a bug on our side. Please report it.