

Vue and You

Speaker Matt Danforth

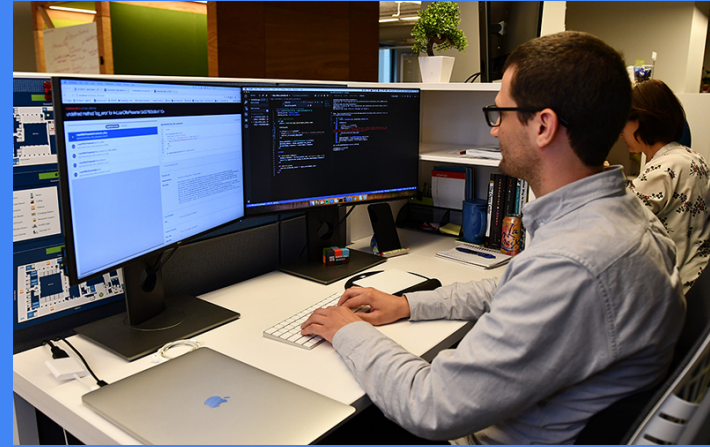
goto;
chicago



**Click 'Rate Session'
to rate session
and ask questions.**

About your speaker...

- Front End Focus throughout career
- Worked in startups, behemoths, waterfall shops and agile shops
- Remembers life before NPM, GIT, console, and several other now essential tools
- Works for Enova
- “Life is Short, Work Someplace Awesome!”
- Some of the smartest and most driven tech workers I’ve worked with yet
- Great company culture and benefits
- <https://www.enova.com/careers/>
- [!\[\]\(6302aad5aed157b291fddf37b4870784_img.jpg\) @enova](#)



In the beginning...



There was Mosaic!



Truthfully, Nexus, Midas, Lynx were all precursors to Mosaic. Since Mosaic allowed image presentation, it was the game changer that started making the web what it has become, for better for or worse.

Turns out a Users really like pictures, who knew?

And then there were Tables!

Tables everywhere! Tables were great for layout! But they werdisplayen't. Wait what?

Tables were created to tabular data. Layout is not tabular data!

But it was early and we didn't have any other great ways to get layouts to "work".

We started using floats and other markup from CSS to accomplish our layouts. It wasn't easy, but we did it.

These days we use *grid* or *flexbox* unless we're supporting legacy browsers.

Semantic HTML is the use of HTML markup to reinforce the semantics, or meaning, of the information in webpages and web applications rather than merely to define its presentation or look. Semantic HTML is processed by traditional web browsers as well as by many other user agents. CSS is used to suggest its presentation to human users.



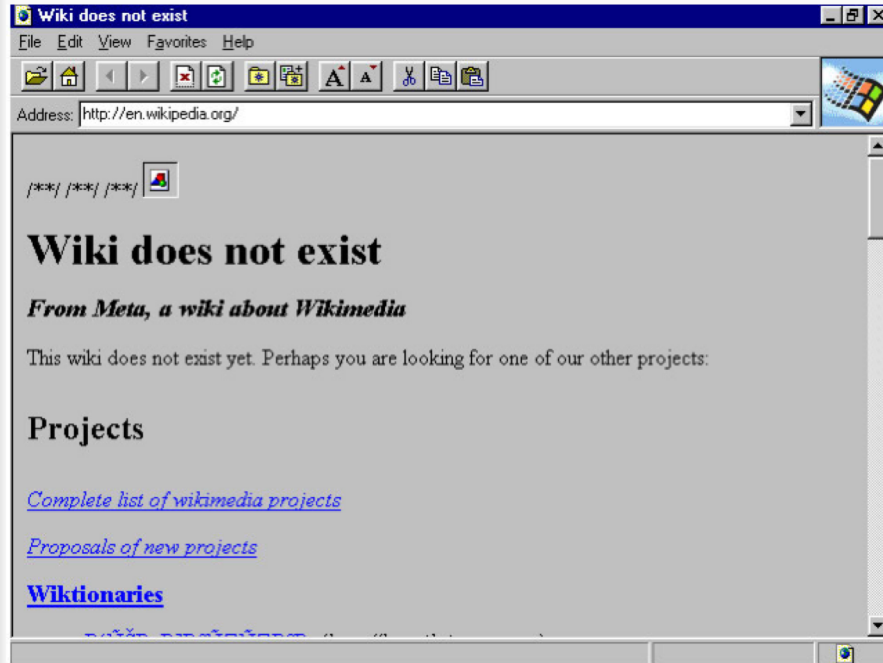
Of course a war broke out...

On the left, Internet Explorer!

On the right, the symbol chosen by Mozilla to represent killing Mosaic, which they pretty well did...

But having coded through the IE / Netscape (mozilla) war, I don't think there were any real winners.

Well, they did stop making Netscape after Netscape 4.0 broke things so miserably but hey, they make Firefox now...and Chrome, and Safari, and Opera...



JavaScript?

Java was a big deal when Javascript was written so why not call the new language of the browser Javascript?

Oh, I dunno, lots of reasons....

This guy Brendan Eich wrote Javascript in 10 days or so.

You thought images in Mosaic made the web explode?
Javascript - KABLOOEY!

Suddenly you could do all kinds of terrible things!



JavaScript - it gets better...

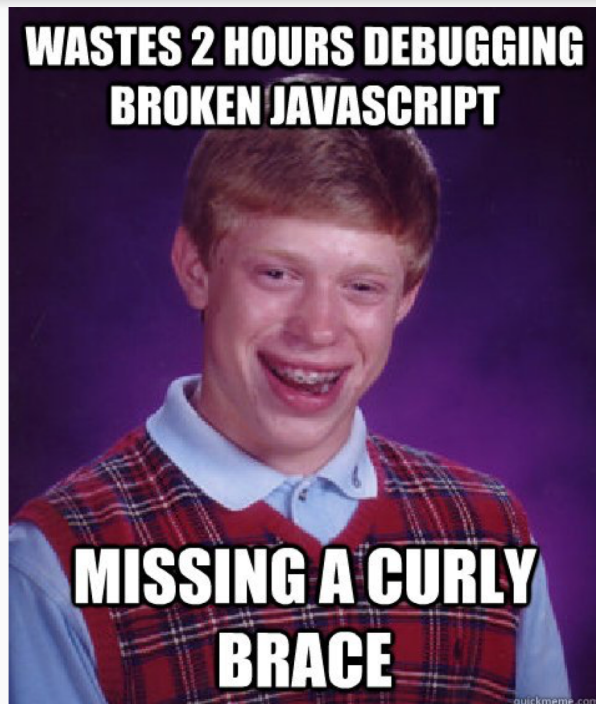
Well eventually it gets better.

Remember that war?

Well part of fighting meant implementing JavaScript differently in IE and Netscape. So something that you wrote worked great in Netscape but not at all in IE. Inverse applied of course.

And there was no console.

So debugging was interesting. Yeah...interesting.



jQuery!

jQuery wasn't the first Library. It's just the one you know about because it won another kind of battle.

There were quite a few libraries back in in 2006 when John Resig released jQuery.

Why did it prevail with options like MooTools, Prototype, and a plethora of other libraries to choose from?

It was small and it was easy to learn.

AND IT SOLVED THE INCONSISTENCIES BETWEEN BROWSERS REGARDING JS IMPLEMENTATION!!!



“jQuery is great.”

- Douglas Crockford

Author of JavaScript: The Good Parts
who typically doesn't like much of anything...

AJAX!

We're out of order here. AJAX was coined before jQuery came about. Jesse James Garrett coined the term in early 2005 predating jQuery by almost a year.

Not that it was easy to use by any stretch. XML was preferred at the time - it took a spell for JSON to really catch on.

AJAX = Asynchronous Javascript + XML

No reload is your pal!

No not this....



JSON

This file format has extended well past the front end at this point. A precursor to both jQuery and AJAX, Douglas Crockford wrote the spec.

The fact that it's easily read by humans, easy to parse, and lightweight footprint suggests it'll be around for some time to come.



SPAs!!!

Single Page Applications.

No full page reload! Header and footer don't change so why reload em?!?!

AJAX and JSON making everything so easy!

Wait. This is NOT easy. It's actually pretty complicated...

Well, not impossible or anything but it's a lot different than the old way of doing things.

No not this....



Frameworks!!!!!!!

So managing all this complexity must require a framework right?

Well, if the number of options for Frontend Frameworks is any indicator - the answer is a resounding Yes!

Not the case though. There are lots of ways to manage complexity. Hoisting it onto an opensource (or paid for) solution just moves it somewhere else.

Which isn't bad if you have a use case for it. But framework for frameworks sake is not good.



The Present



Browser Convergence

It all kinda “works” now.

Microsoft, Mozilla, Google, Apple, Opera are all implementing standards in a similar fashion so variance in coding for each has dropped considerably.

Still a market for those that code in the old way though as legacy apps run by corporations will be a while before they update to modern browser standards.

But so long as you don't have a user base using the older browsers - there is no reason to code for them.

Array.prototype.includes - OTHER

Usage: % of all users
Global 88.23%

Determines whether or not an array includes the given value, returning a boolean value (unlike `indexOf`).

Current aligned	Usage relative	Date relative	Show all	IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	UC Browser for Android	Samsung Internet
							49						
							63		10.2				
							64		10.3				4
11	16	58	65	11	11.2	all	64	11.8	6.2				
	17	59	66	11.1	11.3								
		60	67	TP									
		61	68										

From <https://caniuse.com>

A new framework every damn day...

The most frustrating part of it? Having lived through the Library wars that jQuery won it's really hard to determine who's going to be the winner. The fact that a new Framework probably just got published isn't helping either.

Fortunately things seem to be settling down on this front.

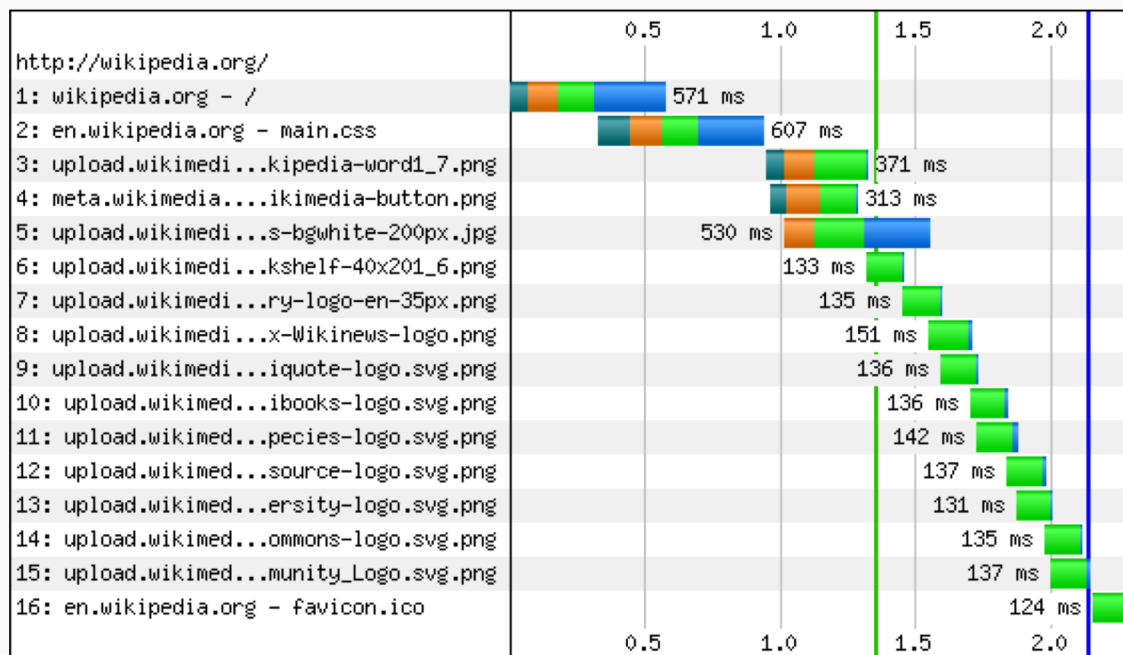
Performance!

Speed matters. It always has but now we've got business metrics to further enforce the notion.

<https://wpostats.com/> has years of data showing how faster sites earn more revenue.

It's a funny thing to watch though as in the dial up days, a whole page might be 160kb maximum. Now there are multiple images that weigh that much.

So balance is still something we fight for.



Accessibility!

This one has been a passion of mine for years. It's not a hard thing to do, until it is. Like performance, making it a priority is a balancing act where success is not guaranteed today.

It's getting better though, with webaim.org, aXe, Lighthouse, and several other players encouraging good practice and legal departments paying more attention based on recent case law - the web is becoming more accessible.



Mobile First!

It was a mantra in UX/Design a few years ago but it's finally catching on everywhere.

Mobile traffic was minimal 8 years ago but it makes up more than 50% of traffic on many brands.

So designs are less frequently coming to us as desktop but instead as mobile first!

Speaking of Mobile - mustn't forget Apps! While Native vs Web is always debated regarding where to put emphasis - both have their place and success in the market depends on having presence in both.



In 2015, the Pew Research Center found that 64% of American adults owned a smartphone of some kind, up from 35% in 2011.

Componentize all the Things!

Any Framework worth its salt has a built in notion and practice around Components.

Components encourage best practice and composable user interfaces.

A Front-End component can be expressed as a rendering function that accepts attributes and produces a visual representation to the user.

By building components - the hope is that we have portable, reusable, easy to isolate and upgrade bits that will improve our work and workflows.



Build stacks for build stacks...

It's not just flat .css, .js, and .html files anymore. Honestly, it never truly was. But for a time we pushed the complexity of putting it all together into other systems. This is no longer the case. With Node and all the packages one might find on NPM, the front end got harder to put together than many UI developers were accustomed to.

Fortunately, the tools are getting smarter and easier to use. It's just a different mindset for many but such is development where our approaches must always evolve in order to remain relevant.



JAMstack

What up Netlify!

The JAMstack is not about specific technologies. It's a new way of building websites and apps that delivers better performance, higher security, lower cost of scaling, and a better developer experience.

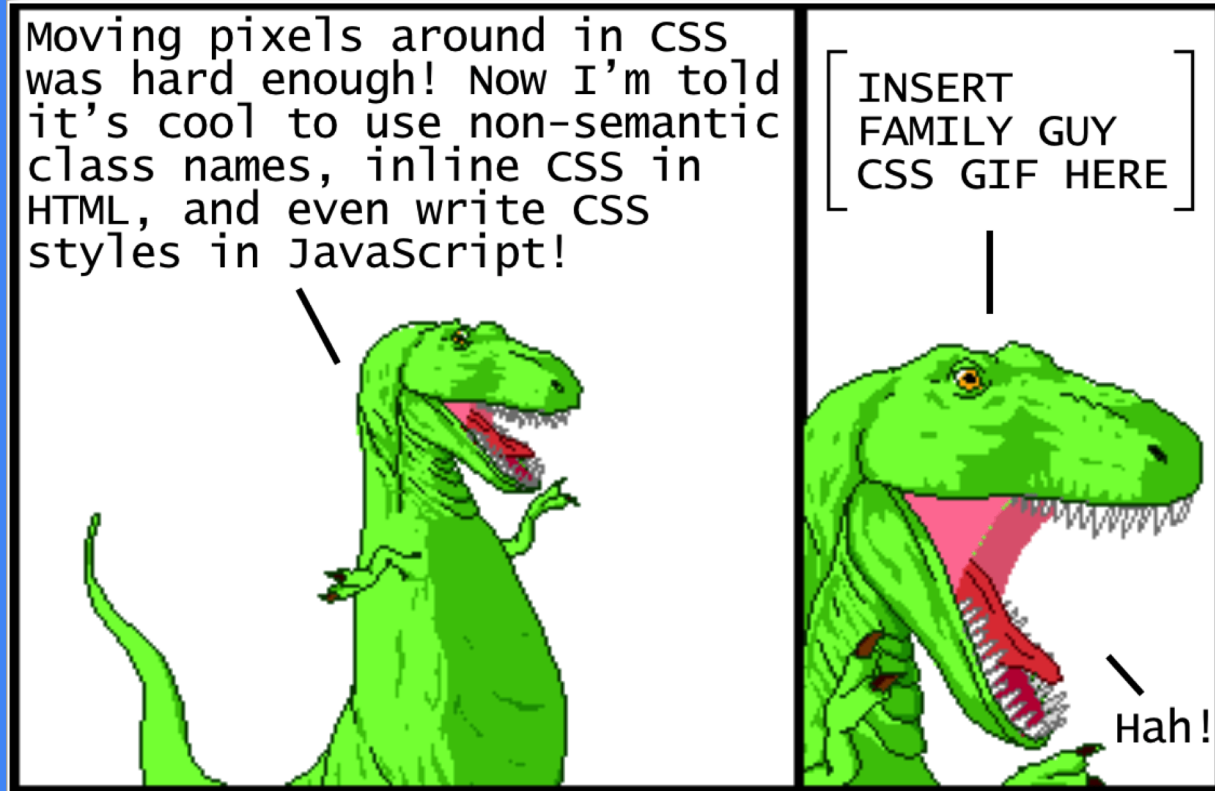
<https://jamstack.org/>



What we didn't cover:

SEO, Flash, Macromedia, Dreamweaver, slices, <blink>, spacer.gif, box model issues in browsers, the box model, Cascading Style Sheets, Microsoft Frontpage, iPads and touch, AOL, GeoCities, Yahoo!, blogs, Wordpress, imagemaps, applets, iFrames, front end validation, XSS or any security really, Jakob Nielsen, ui_builder, ui-built, the DOM, Fonts, NODE!

And so much more...



Vue and You



You...

- Are utterly exhausted by the number of choices available in building a View layer
- For all the things discussed earlier, and all the things we didn't discuss, you want to make a good choice but have a lot of other good choices to make too



You, yes you.

- Do you need to get something up and running quickly?
- Does that something have some ambiguity and scope that has you wanting a few decisions made for you already?
- Might make an App, might just make a page, might do a few different things together
- Want to use forward thinking and modern architecture in your Front-End / View layer?



Getting there

- To be responsible for a whole organizations direction – know how to weigh options.
- Read All The Articles (impossible – read enough)
- Be comfortable getting it wrong
- Stars on Github
- Maintainers
- Usability
- Developer enthusiasm

React is cool right?

- So Very Cool!
- But it's Gen1 and some of it's getting a bit crusty b/c compatibility
- FLOW? JSX? Both have some questionability while noble in their creation.
- Only solved the license problem* recently
- You will have any difficulty in finding people with experience in React.
- Hiring them is another story...
- React is certainly not a bad choice and I don't diss shops that do use it, especially those that waited to adopt *after* the license problem*

*To those that argue it was never a problem, I say that as engineers we should make sustainable choices with minimal jeopardy to our code base. If a legal battle with a behemoth like Facebook seems sustainable, I'm entertained and will admit I lack your bravery.

Angular is Google so must be great!

- Having written a lot of now unsupported code, directing developers to write a lot of unsupportable code, I'm bitter.
- It's not a bad platform, but it's opinionated and heavy.
- If you're a Java shop – think about it, because Miško is rather sharp and invented the framework – and he was very Java when he did

- <https://killedbygoogle.com/>



Search

all (161) - apps (12) - services (137) - hardware (12)



April
2019

Missing something?

We rely on contributors to proofread, check accuracy, and keep this list up to date. The Google Graveyard is ad-free and open source. Feel free to get involved on [GitHub](#). Follow us on [Twitter](#) to get instant updates.



December
2019

Google Fusion Tables

Bites the big one in 8 months, Google Fusion Tables was a web service for data management that provided a means for visualizing data in different charts, maps, and graphs. It was over 10 years old.



October
2019

Google Hangouts

"Off with their heads!" in 7 months, Google Hangouts was a communication platform which included messages, video chat, and VOIP features. Execution date is tentative. It was over 6 years old.



May
2019

Fabric

Expires in about 2 months, Fabric was a platform that helped mobile teams build better apps, understand their users, and grow their business. It was over 4 years old.

{{ Framework Invented Yesterday }}

- Maybe.
- What's your use-case and are you ready to justify the time spent on beta tech?

Vue and You and You!

Why?

Shortest Answer:

Small footprint - fast performance – developers developers developers.

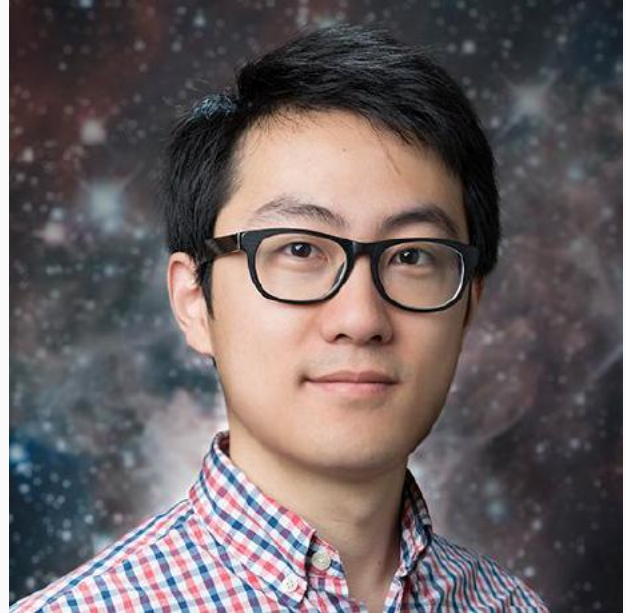
Evan You resembles John Resig.



Ok, maybe not entirely



!=



History is a good teacher

- Jason Hoffman recently wrote “Yet Another Javascript Framework” (<https://css-tricks.com/yet-another-javascript-framework/>) that does a much better job of delving into the details of ‘the old days’ than I had hoped to.
- Yes I DID start my deck with In The Beginning but feel no shame since I had started working this deck prior to his article!
- Prototype
- jQuery
- MooTools

Size and Speed

- Funny how history even escapes the writers (me in this case)
- Not that much difference in size for MooTools and jQuery
- Performance comparable as well
- So it *was* (?) maintainers and contributors?
- jQuery had the lower barrier of entry – easier to comprehend and run with

Evan You

- Inventor of Vue
- He developed in Angular, decided he wanted to take the best parts, and some from React, and make Vue
- Remind you of Resig and Prototype?



 [@youyuxi](https://twitter.com/youyuxi)

Evan You

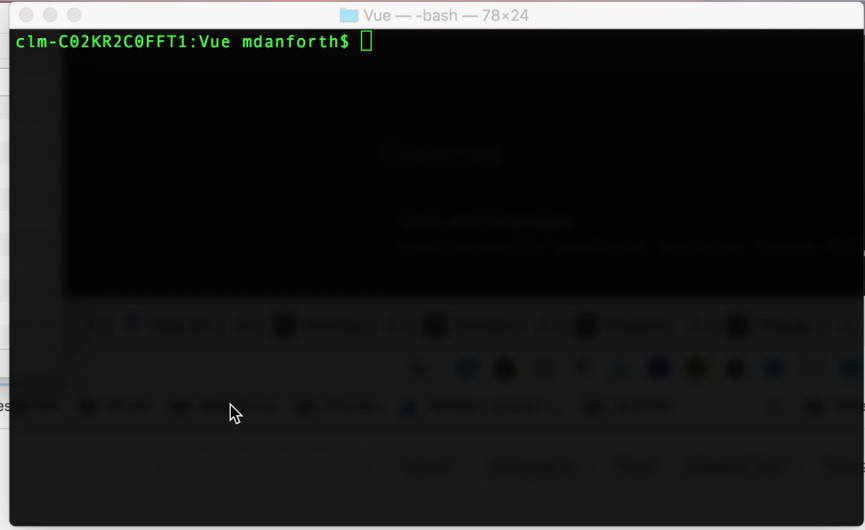
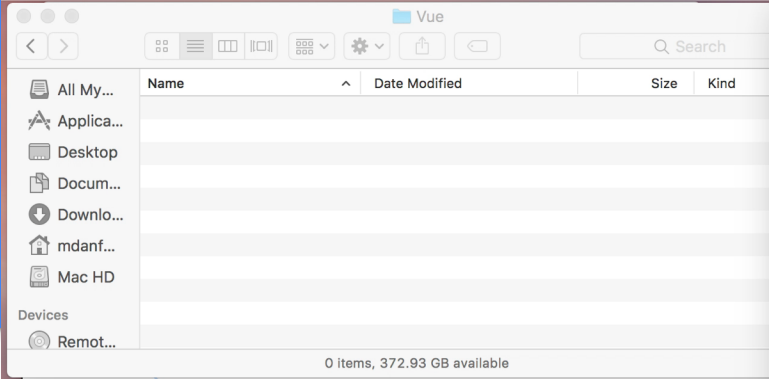
- Prolific
- Boundless in contribution and community building
- Has a clear vision of how to move this project forward
- Not ruled by corporate ownership (probably – Patreon is what it is)
- Amazing trajectory for Versions in declaring what's coming and when
- Incredible uptick in usage according to Stackoverflow
- So many Stars in GIT!

Enough about You already...

- So slides are fun but what about coding?
- Let's look at what it takes to get a Vue instance up
- And a default folder structure....
- And an HTTP request library...
- And a premade component library based on Googles Material Design...
- This is going to take WAAAAAY more time than we have, isn't it?


Live Coding Happens Here

- Ok – so it's pre-recorded – history is a good teacher and if history teaches anything it's that if something can go wrong it will. So if the video fails we actually will try to live code. Alternately, if the live coding failed, I couldn't make a video...
- Some things we learned:
 - You really can insist on State too soon
 - Use a-la-carte from Vuetify from the get-go or you're going to have to rewrite some tests
 - I can't even express how quick and easy it was to build a form with Vuetify, complete with validation – what in the old days would have taken a week took a few hours of fun





Apps View Image javascript ProgressiveApp CSS Peformance Des


Vue.js




The Progressive JavaScript Framework

 WHY VUE.JS?

 GET STARTED

 GITHUB

Special Sponsor



Pretty cool huh?

- But I could have done the same thing in React using Gatsby.
- But I could have done the same thing with x using y
- So really, does it matter?
- Perhaps not today, but it will someday when there might be a 'winning' framework. And finding developers to work on your code is always easier when it's common across orgs
- Ultimately, we win regardless of choice
 - Because we're developing components
 - Because we're pursuing best practices
 - Because we're Accessible and Performant
- So even if I haven't convinced you that Vue is for you – I hope you at least make choices that further a better internet for everyone.



Please

**Remember to
rate this session**

Thank you!

goto;
chicago



Click 'Rate Session'

Rate **5** sessions to get the
supercool GOTO reward