11:32

Hey everybody, thanks for coming out to this talk today. I realize you all could be doing better and more interesting things, so I appreciate your time. I'm Davis Ford, I'm a developer with the SIG. It's pretty cool to see some of you guys and girls from Slack. I know we have a lot of talented people here who have the ability to do amazing things with technology, and I just want to use this opportunity to draw attention to a technology that I think you may enjoy. So I'm here to talk about Electron, and I'm also going to talk a little bit about React, just because I can.

I’m going to explain what Electron is, what the benefits of Electron are, how you can get started with Electron, and why your team might want to consider using Electron for your next project.

So, what is Electron? Electron is a framework for creating cross-platform desktop applications using web technologies. It’s used by companies like Slack, Atom, and Basecamp. It’s also used by Booz Allen Hamilton – I’m on the CMS TCPI project, where we’ve deployed an Electron-powered application to clinicians across the country.

Let’s come back to that phrase – a framework for developing desktop applications using web technologies. That kinda sounds like a bunch of buzzwords, but what it boils down to is that you can write a webpage just like you normally would, and distribute it as a desktop application. When I talk about desktop applications, I want to emphasize cross-platform. This means you can use Webpack and Babel and React and Angular and Vue and anything else you enjoy writing. You could even just write HTML, CSS, and some vanilla Javascript on their own, and that will actually work too ☺

The really neat thing about Electron is that it lets you develop using tools that are familiar to you, or to any web developer, but without the restrictive sandbox constraints of a modern browser. Anyone who’s developed a sufficiently powerful web application has probably run into the constraints of a browser quite quickly – you don’t get access to the filesystem, you can barely share scripts across domains, it’s generally just a pain to do anything above the level of shuffling some HTML around.

What is Electron good for? Well, I’d rather show than tell.

*\*\* PREREQUISITE - open iterm with 2 vertically split panels, cd both into Documents/electron-talk/electron-quick-start and npm i*

*Open new tab in iTerm, cd to arborist-electron*

*PHPStorm – need arborist-electron and electron-quick-start opened*

What I’m opening up now is electron-quick-start, which is basically the easiest way to get started in electron. This is the repo that Electron has on their front page, all I’ve done is clone it and install dependencies with NPM.

I’m just going to start Electron by running npm start.

*[ open electron quick-start ] npm start*

So, this is pretty neat already. This is a google chrome window, basically (it's a little different, but I'll get to that in a minute). You can write your CSS just once, and you only need to worry about Chrome's support for Javascript. Every user will have the same experience on Windows, Mac, or Linux. The cool thing about Electron is that you can do this:

*[ open vim ] vim index.html, change title to Hi BAH! and refresh*

Now back in our Electron homepage, we'll just require fs (since we have access to the Node API)

`````

<script>

const fs = require('fs');

fs.readFile('./../features.txt', 'utf8', (err, result)=>{

result.split('\n').map(x=>document.write(`<h2>${x}</h2>`))

})

</script>

`````

This is going to open up a file called features.txt and print them out in HTML one by one.

[refresh page]

So at this point, you can proceed to build your application like normal. It's cross-platform, with a standardized development experience.

You'll notice that this is pretty barebones – there’s only a few files in our main directory. For a lot of development teams, this is where you'd throw in your Angular, React, whatever you enjoy doing.

Let's take a look at a more souped-up version of electron - electron react boilerplate.

It's a beautiful package that gives you React/Redux/Hot Module reloading, etc, all set up and ready to use out of the box.

Hot module reloading, in particular, makes electron and react development so easy. Just amazingly easy. Watch this.

[open arborist]

So, this application is actually kinda the reason I'm talking to you guys today. I wrote this at the beginning of this month. This is a real application that has users, and a real world use. It's basically a file-sorter that helps 360 degree filmmakers sort their camera files after a long shoot. So if you think about how a 360 video is built, there's usually 4 or more cameras running at the same time. Let's say you're shooting some real estate, you'll probably end up with 660-70 files on each camera. When you go home at the end of the day, now you're facing manually sorting through hundreds of video files, matching them up with each other, ignoring random test shots, etc. etc.

My good friend at Bonsai Electronics has a patent on a 360 degree camera mount, so he does a lot of 360 work, and he came to me with this problem a couple months ago. I said no worries, sounds pretty doable, and I whipped up a quick python script in a couple days that read the files and sorted them. Now, because python is a pretty strong language for file operations, this application wasn't that difficult to deliver. The problem was - it was command line. No one was going to download python, install it, make sure the path is working, and then run a command line utility to sort files. That’s simply not a realistic expectation of the end-user in this day and age.

So I set about making a GUI for this python application. I tried PytQT5, I tried PyQT4, I tried TkInter, and I tried just about every current version of python GUI libraries I could find. They were all riddled with issues, didn't compile cross-platform, and they frankly were pretty ugly by default. I tried for a few weeks to get everything working, and I eventually just fizzled out.

And then I had a thought - what about electron? Within a few days, I re-wrote this on top of the electron-react-boilerplate. I am able to distribute this to anyone, anywhere, and I know that it'll work on their computer.

Now, let me show you a little glimpse of what it’s like to work with Electron and Hot Module Reloading. So, I’m going to start this file-sorting process, which will take a minute. Now, I’m going to go in the main React component driving this page, and make a little change. What’s important to note is that the state of my application wasn’t modified –normally when you’re working with a page like this, you’d have to hit refresh and lose any changes you had on the page – in this case, you’d have to start the file transfer over. But with Hot Module Reloading, you can just update exactly what you need without touching the rest of your page.

Cool little tip: you can require your modules in a lazy manner to improve your startup times. I actually found out about this while doing research for this talk. You know how typically you end up with 5-6 module imports at the top of your files? Instead of doing that, you can import them within a function, ensuring they are only loading when needed. Just a cool little tip.

Let me bring this back to why Booz Allen might care about this.

How much developer time worth? We’re not exactly an inexpensive class of worker. How much time do your developers spend tweaking CSS and Javascript to target different versions of the browser? Wouldn’t it be better to just target one version of the browser?

In the case of the CMS TCPI project, Chris Wetterman and I were tasked with deploying an offline-first application, with the understanding that once we hit a certain benchmark, we’d convert the offline-first application to a web site. This means that we can write our webapp in electron, and later apply it to our website – cutting down on duplicative work.

Desktop applications allow you access to the user’s Operating System – imagine the possibilities you can explore when your Javascript developers are suddenly granted the same access rights as most programming languages. The built-in auto-update ability means that it’s trivial to push fixes and features to your clients, and your clients will enjoy the same application experience regardless of the operating system they run. A uniform codebase for three operating systems – that’s something worth talking about.

If you want to get started with Electron, head over to electron.atom.io and check out the electron-quick-start repository.