(c) 2014 @joefrance joefrance@totalwebsystems.com

MEAN stack on the Mac

Background

What would you do if suddenly you woke up on an island, okay it's a tropical island with lots of friendly faces, with a web-based line-of-business app to write, a Mac, and no Microsoft Development stack to lean on?

You would be faced with the challenge of replacing the things in the Microsoft stack that you'd grown to know. You might not love all of them but you know them.

This was the journey I struck out on some time back and I'm here to tell you the simple conclusion of what I've found.

It can simply be described as ***"the MEAN stack on the Mac"***.

MEAN stands for **M**ongoDB, **E**xpressJS framework, **A**ngularJS, and **N**odeJS.

The short list of tools is:

Language	JavaScript (CSS, HTML, JSON)	C#, JavaScript (CSS, HTML, JSON)
IDE	WebStorm IDE	Visual Studio
Database	**M**ongoDB	SQL Server
Framework	**E**xpressJS	.NET
Front-end	**A**ngularJS	HTML, CSS
Web server	**N**odeJS	IIS - Internet Information Server
Source control	github	TFS - Team Foundation Server

I've coined the term NEAT stack as a variation of the MEAN stack. In this configuration you use T-SQL / SQL Server instead of MongoDB.

You might substitute SQL Server for the database for any of the following reasons:

- MongoDB is not well-suited to store your data
- You cannot convert from SQL Server to MongoDB in the scope of your project
- You need a transactional database and SQL Server fits the bill for you

The "T" in the term "NEAT stack" is borrowed from "T-SQL" and the "Tedious" driver for SQL Server under NodeJS. Tedious represents a spoken form of the acronym T.D.S. (Microsoft's Tabular Data Stream Protocol). There are a number of "tedious" node packages but I have settled on "mssql".

Impetus for change

I have been a professional developer for over 20 years. During that time I worked on a wide ranging set of platforms from the AS/400, Linux/Freenix, IBM mainframe, etc. Also I am a polyglot as many of us are these days. But largely I have worked in the Windows space on the so-called “Microsoft stack”.

Since the advent of the Internet I have resisted and bemoaned the idea of developing line-of-business apps in HTML. HTML was and is a mark-up language originally suited for, well, mark-up and web surfing.

As a developer community somehow we all conceded to the notion that “after all, isn’t it just easier to give the end-user a URL and let them log in and use the system”?

And this is a true statement, taken from the perspective of the end-user.

However, let’s consider the cost of such a “convenience”. I maintain it is a high cost.

While writing an app in HTML, we spend a lot of time as developers fighting with and patching for issues in “the plumbing” of HTML, CSS, etc. instead of just writing the app code. All this struggling is done in order that we can “hand the end-user the magical URL”.

There are “fat client” technologies, many from Microsoft, that let us “just code” and leave the plumbing to the plumbers. There are Flash/Flex, Java, and of course from Microsoft we have WinForms, WPF, and Silverlight. All these technologies have the advantage that we can protect the user from themselves by allowing the “client framework” to tightly control enabling/disabling entry controls, handling data binding, and a whole host of other things that we shouldn’t have to consider as developers.

But more importantly these “fat client” technologies, by and large, give the end-user a better and richer experience than they get from a “web app”.

So why don’t we use “fat/thick client” apps as developers? Very simply the classic answer is “the deployment” nightmare. Now I’m not here to argue with the headaches you might have had with deploying “thick client” apps. But, how do those deployment war-stories really compare against how we struggle with pounding HTML into submission? The world may never know.

Scott Hanselman gave a keynote address at devLINK 2013. In the talk he mentioned two things that stood out to me:

1. JavaScript could be thought of as a sort of operating system and called it a “Virtual Machine” of sorts.
2. JavaScript has really won the battle as a de facto programming language and is a serious contender in many arenas like game development, business app development, etc.

So it was around that time that I “cried uncle”. I thought, “Okay HTML/JavaScript, I give! You’ve beat out the thick-client model. But you have to give me something back”. And ultimately it did: AngularJS and NodeJS.

I had already been hearing the buzz about AngularJS for front-end development. And later I'd learned of NodeJS on the back-end. I found it novel that one could run JavaScript on the whole stack. But I wanted to know if it was practical.

The more I looked at AngularJS the more I liked it. I began to watch it and gauge its sticking power. It seems like a new silver-bullet is born every minute in the tech world.

Increasingly these points pressed themselves on me:

- Two-way binding similar to WPF/Silverlight
- Lack of focus on "the plumbing" and concentration on business coding
- Tried-and-true MVC model
- And last but not least, backing from Google and building adoption by many folks in the industry with apparently similar experiences with my own.

I then began to look for projects to use with AngularJS to see if it lived up to its promise. So far for me it has delivered the goods consistently and cleanly.

Now I won't tell you that it's easy or that it does not have a learning curve. But it is fun to learn and it is familiar, to me at least, in some areas that I've really liked from thick-client tools like WPF and Silverlight.

The search for alternatives

Leaving Windows behind and developing solely on the Mac lead me ultimately to settle on the MEAN stack. But along the way I considered some of these "Microsoft stack" alternatives:

- LAMP - Linux, Apache, MySQL, PHP
- MAMP - Mac, Apache, MySQL, PHP
- Java (although I didn't seriously consider this option)
- Drupal, Joomla, Ruby, and Python are out there too but didn't appeal to me personally

MEAN Stack - MongoDB, ExpressJS, AngularJS, NodeJS
NEAT Stack - NodeJS, ExpressJS, AngularJS, Tedious/T-SQL

There is an excellent video fly-over of the MEAN stack from WebStorm:

MEAN Stack Walkthrough and Tips by John Lindquist
https://www.youtube.com/watch?v=JnMvok0Yks8

In 12 short minutes in this video Mr. Lindquist:

- Demoes WebStorm (IDE) & creating a MEAN project
- NodeJS highlights
- AngularJS highlights
- MongoDB highlights
- Even debugging on the client and server sides