JavaScript Frameworks in the Enterprise

By Pratik Patel, Java Expert and CTO of TripLingo

The usage of JavaScript to build Single Page Web Applications, or SPAs, is on the rise throughout the software industry. In this brief, I would like to discuss two major topics that arise when I present at conferences on JavaScript and SPAs. The viewpoint I present here is from talking with mainly enterprise developers exploring this area to either extend and enhance an existing product, or to build a new product around a SPA architecture.

Single Page Apps Architecture

First, I'd like to discuss SPAs in general. The term SOFEA, or Service Oriented Front End Architecture, aptly describes modern SPA projects. Historically we've used serverside generation of HTML (and CSS and JavaScript) to build web applications, this is what I call "traditional web apps". The shift from clientserver to web applications in the late 1990's was, of course, enabled by the Internet and the Web. This shift moved, almost entirely, the code and processing of these kinds of apps to the server and proved to be a simpler, scalable model for application development and deployment than clientserver.

SPAs are the modern version of clientserver. In a SPA architecture, the "client" is the web browser, and we code it using JavaScript. The server, however, has also changed dramatically in the way we perceive it as developers and architects. We think of the server as exposing services rather than directly exposing the database, even if it is to expose data. For example, RESTful web services.

While we must develop SPAs using JavaScript on the client, the server can continue using its current technology. I'm assuming that this technology is something modestly modern, such as Java, C#, Ruby, etc. and that it has supporting libraries for building serviceoriented endpoints, specifically REST type of endpoints. Many developers feel they need to move to using JavaScript on the server (Node.js) to truly embrace a SPA architecture. However, that is not clearly the case for projects today. There is a class of SPA architecture known as "isomorphic apps" that is on the horizon where this would be a case but isomorphic app architecture is still in its infancy in both our understanding of them and proven frameworks. We won't discuss isomorphic apps further as it's a large topic by itself.

One small aside: SPAs are nominally built using JavaScript directly, but there are frameworks out there that allow you to build SPAlike apps without JavaScript. GWT and JSF come to mind. However, developers do not consider these to be true SPA frameworks. The rest of this brief does not discuss these compile to JavaScript frameworks.

The Curse of Choice

A common question I receive when talking to developers and architects about SPA architecture and development is: "What framework should I use?" The answer, of course, is "it depends!" I like to break down SPA frameworks into two broad groups: Markup-driven and Widget-based.

Markup-driven frameworks are ones where the developer creates HTML templates along with the "shell" for inserting these HTML fragments, and the framework provides utilities for assembling and manipulating a complete HTML page. The key is that the developer hand crafts the HTML markup using basic HTML tags like tables, buttons, etc., along with the CSS, to define the appearance of the UI.

Widget-based frameworks inject or produce HTML markup-based on the developer's usage of the framework's API. Widget-based frameworks are bifurcated into two types: JavaScript widgets and markup widgets. The difference being that in one you write JavaScript and create framework specific JavaScript objects (var button = new Button({buttonProps}), and in the other you insert a specific attribute (<input data-role="numerictextbox") to create that specific widget.

As of summer 2014, there are a number of frameworks that have developer mindshare. One dramatic trend I've seen over the last year is the rise of AngularJS. It is a markup-driven framework that came out of Google which has quickly gained popularity, especially among the enterprise development crowd. There are other popular markup-driven frameworks, such as Backbone.js and Ractive.js.

There are a number of widget-based frameworks that are excellent, namely KendoUI and Sencha. KendoUI has recently opensourced most of their code and provides a large number of widgets, as does Sencha. KendoUI is my current favorite widget-based framework because of their clean code style and focus on performance. Interestingly, it also integrates into AngularJS to provide its widgets.

Because of the tremendous capabilities in these frameworks, sometimes a framework may be both types, markup-driven and widget-based. The type of SPA you are building should play a major role in which type of framework you select. The example I like to give is: If you're building a shopping cart SPA, you likely want to use a markup-driven framework; if you're building a dashboard, you will want to avail of rich widgets provided by a widget-based framework. I would also recommend you visit http://todomvc.com/, it is an excellent resource you can use to compare all the major JavaScript frameworks.

About Pratik

Pratik Patel is the CTO of Atlanta based TripLingo (http://www.triplingo.com/). He wrote the first book on 'enterprise Java' in 1996, "Java Database Programming with JDBC." He has also spoken at various conferences and participates in several local tech groups and startup groups. He's in the startup world now and hacks iOS, Android, HTML5, CSS3, JavaScript, Rails, and ..... well everything except Perl. Pratik's specialty is in large-scale applications for mission-critical and mobile applications use. He has designed and built applications in the retail, health care, financial services, and telecoms sectors. Pratik holds a master's in Biomedical Engineering from UNC, has worked in places such as New York, London, and Hong Kong, and currently lives in Atlanta, GA.