Yow Connected Mobile and IoT Conference Report

(Original creator: eknochs)

The YOW Connected conference was held in Melbourne on 17th to 18th September http://connected.yowconference.com.au/ . It was a developer's conference based on Mobile and IoT topics. Since Uniface is adding Mobile devices as a deployment option in 9.7, I thought I should find out what problems existing and aspiring mobile application developers are experiencing, and how they are solving them.

Keynote presentations should give a buzz to the audience, and this conference had one on the magic of mobile devices, with plenty of Harry Potter analogies. Another keynote was on IoT wearables, which showed many examples of how technology failed to make good partners with fashion. The technical presentations for smart phones generally fell into iOS and Android camps, i.e. native app development was very important to most attendees. There was a strong belief in maximising the user experience over portability of business applications, and this meant gaining the most out of native platform features. Indeed, whilst people figure what to do with Apple watches and Galaxy gear, I can imagine that they need to use native platform features.

Given that there was strong commitment to native mobile development, it came as little surprise that using JavaScript to program the user experience was seen as second rate technology. However, when an industry heavyweight like Facebook come along with a new JavaScript library, people start to overlook their general JavaScript prejudice. So this is where I first learnt of the React library http://facebook.github.io/react/index.html . React differentiates itself from other JS libraries with the use of a virtual DOM. Your JS programming updates the virtual DOM, and then these updates are combined to update the real browser DOM. This turns out to perform much better than scanning and updating the browser DOM for each individual update. But Wait, Facebook has done more … they have also introduced React Native https://facebook.github.io/react-native/ . So, instead of using JS to manipulate HTML for rendering inside a UIWebView container, you use JS to manipulate view structures inside native components. There is still a virtual DOM equivalent, and then the actual native component objects are updated by this library. Facebook has initially developed this with iOS native component models, and thus you are limited to developing on an Apple platform, but as recently as one month ago, you could also target the Android native platform. This is where the development philosophy gets interesting. In the Uniface world we are used to "write once, deploy anywhere", and there is a suggestion that React Native might support something similar, but in fact their mantra is "learn once, write anywhere". Even though React Native has adopted standardized layout mechanisms from CSS3 using flexboxes, they really encourage you to choose native component types that exploit the best of what the native platform offers, i.e. avoiding a lowest common denominator user experience, at the expense of full portability and productivity. My personal view is that you would only choose React Native when user experience is clearly a very high priority, and use more proven platform independent tools, like Cordova, as often as possible. Perhaps this would change as React Native evolves, after all, it is still at release 0.11.0. There were numerous IoT technical presentations, and I include Virtual Reality and Augmented Reality apps in this category. The presentations covered things like how to make a LED flash on a microprocessor (see https://www.particle.io/prototype for examples) using calls to the Particle Cloud written in JavaScript (groans from the smart device developers, but IoT developers don't mind). Another presentation covered VR from high-end Oculus products to DIY Google Cardboard, once more, using JavaScript to program the API calls. This article shows the basic JavaScript libraries that you can practice with http://www.sitepoint.com/bringing-vr-to-web-google-cardboard-three-js/ .

I did initially struggle to see why these IoT technologies would be important to business applications, and thus where Uniface might be able to help, but eventually the spirit of adventure in programming comes out of you (OK, call it inner geekiness) and you want to play with it anyway. Besides, we know that other IT technologies have evolved from past experimentation, and so I wasn't surprised to find myself ordering a Google Cardboard device on the weekend https://www.google.com/get/cardboard/get-cardboard/ . Soon enough, I may be using the Uniface JavaScript API, to send data to 3D rendering JavaScript libraries, for display in my Google Cardboard device.