by daniel chudnov
Information Technology Specialist
Office of Strategic Initiatives
Library of Congress

A Mobile Strategy Web Developers Will Love

IF YOU WANT TO DELIVER YOUR LIBRARY'S CONTENT AND SERVICES TO YOUR USERS ON MOBILE DEVICES, FOCUS FIRST ON DELIVERING A SOLID SUITE OF CONTENT AND SERVICES TO THEM OVER THE PLAIN OLD WEB.

I've been thinking about iPads, iPhones, and Android a lot lately. In early April (a few weeks from now as I write) iPads will disperse from Apple headquarters in Cupertino, Calif.; you've probably seen a few already. We know from Apple's years-long PR blitz that many people have programming apps for their iPhones and upcoming iPads. Several of my hacker friends are very excited about the Android platform for both its capabilities and its openness. I love the idea of smarter, cheaper, always-connected devices in the hands of more and more people to whom we can more easily provide our libraries' content and services. As a hacker and a librarian, it's natural to want to create tools that work on these devices. So that leaves me with a question: Do I write apps for Apple's products or Android-based products?

If I'm working in a library (and I am) and I want people to be able to use the library from their always-connected devices (and I do!), it wouldn't be a waste of my time to develop software for just one of these two platforms. Sales figures released in March 2010 indicate that Android phones and iPhones are both selling like hot cakes, and each million new phones sold on either platform means a million more potential users. The good news is that if you want to use this strategy there are several paths available for you to create and deliver content and services that will fit on your users' iPhones, Droids, iPads, and Nexus Ones. Of these available paths, I think one is far and away the best choice, but I'll save that for the end. Before we get there, I'll review some of these choices, starting with the mainstream approach offered by Apple for its products and by Google for Android-based products. This might get a little bit technical, but if you're considering these questions, you should know what the options are. If your eyes start to fuzz over when I start talking about IDEs, just skim forward to the section called A Simpler Approach.

Developing for iPhone, iPad, and Android

If you want to develop software specifically for iPhones, iPads, and iPod touches, the kind of software people might be able to buy in the App Store, you will want to get to know Apple's software development tool chain. This is centered around the Xcode integrated development environment (IDE) for Apple's Mac OS X operating system, which is available for free to everyone who installs OS X (it's on one of your install discs under "Developer Tools," or see http://developer.apple.com/technologies/tools/xcode.html for more info). Xcode is not for casual code twiddlers; it's a full-on IDE, and if you don't have experience with an IDE, you'll have quite a learning curve before you can be productive in one.

An IDE supports all phases of software development (hence the "I" in IDE, for integrated),

from initial design to compilation and debugging to versioning, testing, and refactoring to building packages for distribution. Many IDEs are optimized for use with a particular platform or language, and many can be used on diverse platforms with several programming languages. Apple's Xcode is optimized for use on OS X and for developing software targeted for use on OS X. It's centered around one main language, Objective-C, but it also supports additional languages (including Java, Perl, Python, and Ruby, which are the four languages I've used most in the past 15 years). If you're using Xcode to write dedicated iPhone apps, though, you're probably going to want to use Objective-C, because that's the language Apple targets with its most complete APIs (application programming interfaces) and tools for building iPhone apps. The best multitouch, graphics, sound, phone, and GPS features are all best implemented with Apple's Objective-C tools, and the best way to use those is in Xcode.

If you want to write Android apps, you'll need to get the Android SDK (see http://developer.android.com) and install that along with the Eclipse IDE (http://eclipse.org). Eclipse is a long-running, free software project used worldwide on many platforms (works just fine on Windows, Linux, and OS X) and with many languages, though its origins are as a development tool for Java. That's a good fit for Android, though, because the Android SDK is implemented in Java. So as with iPhone apps and Objective-C, the best and most robust APIs for building Android applications are going to require you to write some Java code, and the best way to do that is with Eclipse.

Xcode and Eclipse are both excellent tools, industrial-strength by any measure, and used daily by countless programmers worldwide. I've used them both with success, though I've only done paid work in Eclipse and have just dabbled in Xcode. There's one last piece to the puzzle, though: If you're writing and testing mobile apps, you need a way to see what they'll look like on the mobile platform. Apple and Google let us do that with simulators. If you install the iPhone and iPad SDKs, you'll get what looks like a little virtual iPhone or iPad that pops up on your screen whenever you want to test your apps out. Similarly with the Android SDK, you can get simulators for several popular Android devices and see how they work too. Because of Apple's more homogeneous product line, you don't have to worry much about trying out many different versions. With Android's many devices, on the other hand, there are many simulator options, which will require a little extra configuration and more testing.

A Simpler Approach

If all of this sounds like a lot of work that might be beyond you, I agree that it's also beyond what I want to do. I could probably learn my way around Objective-C and could probably relearn Java well enough to use the Android SDK, but the learning curve for me would be long and slow. It would probably be months before I'd learned enough beginner lessons to produce something basic, months again before I could produce something good, and months beyond that to test and refine it to be reliable. If I were in business to sell apps, that would be the cost of developing apps. All evidence indicates that, for businesses selling apps, this can be a highly lucrative endeavor. But I'm not in business to sell apps and I presume you're not either. We're running libraries. We don't have half a year to let an already busy programmer go learn a new platform before he or she can turn out results. Fortunately, there's another approach that appeals to me a lot.

There's a middle ground developing between building dedicated apps using dedicated tools, like I previously described, and just resorting to building

websites and leaving it at that. In between these two extremes is the idea that you can build a smartphone app using the same tools you use to build websites—HTML, CSS, and JavaScript—and with a little bit of glue, you can fit them onto smartphones just like any other dedicated apps. iWebkit (http://iwebkit.net) lets you decorate an otherwise ordinary webpage with a set of CSS classes that turn buttons, lists, div tags, and the like into familiar iPhone-style apps when they're used on an iPhone. With iWebkit, you're just adding another few steps to developing a website—you add the iWebkit-specific patterns using HTML and CSS and then you test it out on your iPhone (or your simulator).

IF YOU'RE WRITING AND TESTING MOBILE APPS, YOU NEED A WAY TO SEE WHAT THEY'LL LOOK LIKE ON THE MOBILE PLATFORM.

Two other tools take this a step further. iWebkit is great for deploying web content onto the iPhone and iPad, but it doesn't help with Android phones. A newer set of tools addresses this specific problem. Appcelerator (www.appcelerator.com) and PhoneGap (http://phonegap.com) connect JavaScript to the iPhone and Android SDKs themselves. So instead of just adding CSS classes that would look good on an iPhone, you can write JavaScript code that hooks directly into Android or iPhone SDK features and lets your web apps use smartphone features such as accelerometers and geolocation. JavaScript is a language that many web developers already know. So the prospect of being able to write some JavaScript to get to cool phone features instead of having to learn phone platform-specific Java or Objective-C tips and tricks first to do the same thing is promising. Even better, these tools bridge the gap from JavaScript to not just one phone platform or another but to both and more (both speak BlackBerry too). Because of this, you can build an HTML, CSS, and JavaScript-based application and leverage what you might already know to help turn an application into what could seem to all of your users like any other dedicated iPhone, Android, or BlackBerry app.

The Simplest Approach

Any experienced developer who hears that will hesitate for a second, knowing that promises like this always end up meaning more debugging on more platforms. But maybe we're stuck having to do that anyway, right? In the cases of Appcelerator and PhoneGap, you're still going to have to learn a lot about your target phone platforms, maybe even a few things about the respective IDEs and SDKs, and maybe work with multiple simulators. But there's promise that you might not have to get too deep with any of these. I've done some basic work in Appcelerator to see how well it works, and I can vouch for its basic premise—I was able to tweak a little JavaScript, push a button, and get basic, but native-seeming screens and widgets onto an iPhone app without having to know much about Objective-C. There are a lot of developers for whom this path is quite attractive, so there's a lot of activity in this area right now. It's a good place to get started, but it won't come for free.

Which leads me, finally, to what I think is the best option today for developing smartphone-savvy apps for your users. The tools and techniques I mentioned here are all good, and if you're willing to invest in them, you won't be disappointed with your results. For most of us in libraries, though, that startup cost of the learning curve is too great to take on just now when there's a cheaper, easier, and simpler approach already available to us that requires no new training. Each of the new generation of smartphones includes a better-than-ever-before web browser. You can see whole webpages and zoom in and out to read text; you can fill out forms and click links easily; the whole web is available to you in your mobile device.

For my money, this is where the real action is. It has been for my entire 15-year career, and I bet it'll still be there in another 15 years. If you want to deliver your library's content and services to your users on mobile devices, focus first on delivering a solid suite of content and services to them over the plain old web. The web is a powerful force, and though Apple loves to tell us how many apps its users buy and install, what exactly are the odds that somebody's going to find, install, and remember to use a platform-specific app you've spent countless staff hours developing? If I were making choices about how to provide a mobile strategy to my library's users, I would focus first on making my library's website work well.

For validation of this approach, look no further than Apple's own marketing of its new iPad: "The best way to experience the web," it says. If it's right, isn't that all the more reason to make the web the best way to experience your library? ■

Daniel Chudnov is a librarian working as an information technology specialist in the Office of Strategic Initiatives at the Library of Congress and a frequent speaker, writer, and consultant in the area of software and service innovation in libraries. Previously, he worked on the DSpace project at MIT Libraries and the jake metadata service at the Yale Medical Library. His email address is daniel.chudnov@gmail.com, and his blog is at http://onebiglibrary.net.