ALA Editions • SPECIAL REPORTS

GOING MOBILE

Developing Apps for Your Library Using Basic HTML Programming

SCOTT LA COUNTE

ALA Editions • **SPECIAL REPORTS**

GOING MOBILE

DEVELOPING APPS FOR YOUR LIBRARY USING BASIC HTML PROGRAMMING

SCOTT LA COUNTE

AMERICAN LIBRARY ASSOCIATION
Chicago 2012

Scott La Counte is a librarian at Anaheim (Calif.) Public Library. He is the author of the book *Quiet, Please: Dispatches from a Public Librarian* (Da Capo Press, 2008), which began as a series for McSweeney's Internet Tendencies. He teaches writing online for the Gotham Writers' Workshops. His iPhone app, LibFind, gives the addresses and phone numbers of public libraries across the United States.

Printed in the United States of America

16 15 14 13 12 5 4 3 2 1

ISBN: 978–0-8389–1129-7 (paper); 978-0-8389-9318-7 (PDF); 978-0-8389-9319-4 (ePub); 978-0-8389-9320-0 (Mobipocket); 978-0-8389-9321-7 (Kindle). For more information on digital formats, visit the ALA Store at alastore.ala.org and select eEditions.

Library of Congress Cataloging-in-Publication Data

La Counte, Scott, 1978–

Going mobile : developing apps for your library using basic HTML programming / Scott La Counte.

p. cm.

Includes bibliographical references.

ISBN 978-0-8389-1129-7 (alk. paper)

1. Mobile communication systems—Library applications. 2. Library Web sites—Design. 3. Smartphones—Programming. 4. Application software—Development. 5. Mobile computing. I. Title.

Z680.5.L32 2012

025.04'2—dc23 2011027672

Series cover design by Casey Bayer.

Series text design in Palatino Linotype and Avenir by Karen Sheets de Gracia.

Composition by Dianne M. Rooney.

∞ This paper meets the requirements of ANSI/NISO Z39.48-1992 (Permanence of Paper).

CONTENTS

INTRODUCTION

There are more than one hundred thousand mobile apps for the iPhone, and Android is quickly catching up to that number. Universities have apps, sports teams have apps, authors have apps—everyone seems to be in the app market, and for obvious reasons. Unlike the previous generation, which didn't mind waiting to get their information when they got home, today's generation wants their information on the go. In many respects they could be called Generation Mobile, because for the first time, there is a group of people who can have everything they could ever want in the palm of their hand.

What's alarming, however, is of the hundreds of thousands of mobile apps available for phones, the amount of apps that librarians have built for libraries can be counted on one hand! The District of Columbia Public Library has an app to help you find books in its catalog. By the time this book is published, a few other libraries will likely have joined in, but the number isn't anywhere near where it should be.

What does this mean? That there's an entire generation that's being overlooked.

Businesses have successfully found a new generation of clients simply by promoting services to them in ways that speak to a new type of user: the mobile user. They have proved that successful apps work and should not be overlooked.

For as long as computers and the Internet have been around, libraries have been on the cutting edge of implementing them; perhaps for the first time, libraries are lagging, and it's time to catch up.

Mobile app development has become a pretty lucrative business; it's a fair statement to say that it is the twenty-first century's equivalent of a gold rush—upstart companies are making millions. In 2010, a developer for the Android phone reported that he was making $10,000 a month on an app that helped people find where they had left their car. He's not alone.

It's not in libraries' best interest to go into app development with the intention of making money, but still, there is a huge market for mobile development.

Mobile websites allow patrons to avail themselves of library services no matter where they are. The sites can be set up to allow patrons to reserve books, to surf the card catalog, and more. Mobile websites, however, are a much different interface from the sites that you're used to seeing on the Internet.

There are a lot of reasons libraries have, by and large, stayed away from phone apps. One of the biggest reasons, however, is that the skills needed to successfully build them have historically been greater than what most libraries can afford; an app programmer can cost several hundred dollars an hour. Building a phone app using the phone's native programming language can be hard work and can take hundreds of hours.

Fortunately, new open-source software has made it easy for people with more limited programming skills to take advantage of apps. Using the technologies available, it's possible for any library to create an app at a low cost and to offer patrons a way to access the wealth of knowledge and entertainment stored in every library facility.

The point of this book is to show you what an app is, how to build one, and how to successfully market your app to library patrons. In the forthcoming chapters, you will learn not just how to build an app with relatively no programming skills but also the best practices for marketing the book to your target audience, as well as both free and paid services that are available to make going mobile even simpler and seamless.

There are many challenges ahead for any library attempting to enter this new form of development—and the challenges are among the biggest roadblocks for many libraries interested in extending their library services into the mobile market. I address some of the problems and challenges in this book, but the main point is to offer cost-effective solutions to library staff who are interested in experimenting with building an app for the library.

1 BEFORE YOU BEGIN

MOBILE APP: A DEFINITION

It's difficult to write about mobile devices because that phrase can mean so many things. When I say "mobile app," does that refer to an app that runs on any phone? Or on a specific phone? Just to be clear, there are two types of cell phones:

Feature phone: A feature phone is what the average person carries in his or her pocket. Although many feature phones are becoming more sophisticated, for the most part, feature phones are the more popular basic phones that you can buy. They make calls, they offer texting, and most can take pictures and have very limited web browsing. More than 80 percent of Americans have a feature phone.

Smartphone: Smartphones are best described as computers in the palm of your hand. Several times more powerful than feature phones, smartphones not only can surf the Internet but also can surf at relatively fast, third-generation (3G) speeds. The iPhone, Android, BlackBerry, and Windows Mobile are all examples of this kind of phone.

Many of the ideas in this book are universal; they can be tried on pretty much any phone in existence. The majority of ideas, however, are maximized for use with smartphones.

What is the point of developing apps that fewer than 20 percent of the population can access? Smartphones are the future of cellular technology. Smartphone prices have dropped significantly in recent years. When the iPhone was released in 2007, it cost $599; less than five years later, it cost as little as $99. The same is true of BlackBerries and nearly all other smartphones. As costs continue to drop, the smartphone is becoming more and more affordable to the average consumer.

Mobile development has become one of the biggest growth sectors of web development. People still use computers to surf the Internet, but the amount of time they spend on a computer will decrease as other devices become more prevalent.

The advent of XHTML-MP (XHTML for the Mobile Phone) and WCSS (Wireless CSS) converted two of the most powerful tools in web design into forms that are suitable for those who need to build mobile websites. At the same time, WYSIWYG (what you see is what you get) editors currently provide templates for mobile websites, which makes it possible for just about anyone to design simple websites.

Perhaps the biggest reason libraries need to embrace this growing technology is because the youth of the world use it to speak to one another. The average teen, if given the option, would probably prefer to text a librarian than to visit the reference desk.

WHAT IS A MOBILE APP?

Another question that should be answered before proceeding any further is, What do I mean by "mobile app"? Mobile apps generally refer to two different things:

Mobile website: A mobile website is mobile phone friendly and developed taking into consideration the limitations of cellular devices. If you invested the right amount of time in your library's website, it probably looks pretty good on a ten- to twenty-inch screen (the size of the average desktop or laptop screen). But if you try to access the website from a mobile phone, it probably will look pretty lousy—if it even loads at all. Having a mobile-friendly website would simply mean that the library has a separate web address (e.g., m.library.com—the *m* standing for mobile) to point mobile phones to. This can be done easily by inserting code that redirects the device on the basis of its resolution. For example, if the site detects a device resolution of 1280 × 1024, then it's obviously a desktop, laptop, or tablet computer, and so would be directed to the main page; if the resolution is 320 × 480 or less, then it's some sort of mobile device, and so would be directed to a mobile page.

Native app: A native app is one for which the library has gone the extra step to create an app that is available for purchase in mobile app stores (e.g., iTunes, Android Market)

At the very least, every library should have a mobile-friendly website; it's a relatively easy process that I'll talk about in forthcoming chapters.

DEVELOPING A MOBILE APP

Libraries that want to take the extra initiative and do something more innovative should devote some time to developing a plan for the delivery of native apps to their users. To develop a native app, you have to understand how to develop a mobile website app—so by understanding one, you are actually learning to do two things. By the end of this book, you might realize that you don't want to have a native app, but you will at least know how to develop one, and in doing so, you will be able to develop a mobile app.

The problem with developing a native app isn't the skill level (which is actually about the same as for developing a mobile-friendly website); the problem is the number of phones that are on the market. If you want to develop a native app for all users, then you have to develop one for every phone: iPhone, BlackBerry, Android, Palm, Windows Mobile—and those are just the major ones! In short, developing a native app requires a substantial time investment, and each library needs to consider that when deciding whether to invest in a native app.

The good news is that most phone makers supply app developers with plenty of free tools and resources to make development a little easier. The bad news is that actually getting the app into the marketplace requires a bit of money; each phone requires that developers pay a fee, and some apps are compatible only with Mac operating systems, such as those for the iPhone. These costs, though, help prevent people who are not serious about app development from flooding the markets with apps. Box 1 illustrates the system requirements and fees for each of the five major phones.

The information in box 1 can be a little overwhelming, and chances are that you will not be able to get a library app on every phone. What you need to consider, however, is that if you get a native app on just two phones (e.g., iPhone and Android), then it will be compatible with most smartphone users. In the summer of 2011, more than half of

BOX 1 ■ System Requirements and Fees

Android
Fee: $25 unlimited
System requirements: The Android software development kit (SDK) is free. It requires

- Windows XP (32-bit) or Vista (32- or 64-bit)
- Mac OS X version 10.5.8 or later (x86 only)
- Linux (tested on Linux Ubuntu Hardy Heron)

BlackBerry
Fee: $200 for 10 app submissions
System requirements: The BlackBerry SDK is free. It requires

- Windows 2000 SPI or later, or Windows XP
- 32-bit Windows Vista (BlackBerry JDE v4.2.1 and higher)
- Java SE JDK v6.0

iPhone
Fee: $99 per year
System requirements: The iPhone SDK is free. It requires Mac OS X or later to run the program.

Palm
Fee: $99 per year
System requirements: The Palm SDK is free. It runs on Mac, Linux, and Windows.

Windows Mobile
Fee: $99 per year
System requirements: Windows XP or later with Visual Studio 2008 and Microsoft .NET Compact Framework v2 SP2.

smartphone users are expected to have iPhones; Android is catching up quickly, with more than 30 percent of the market. That means that if ten people walk into a store to buy a smartphone, eight or nine will walk out with either an iPhone or a phone with the Android operating system.

THE ESSENCE OF THE MOBILE WEB

Mobile websites should look and interact much differently than traditional websites. Mobile websites involve different design concerns than regular websites that you view on a computer. In some ways, designing mobile websites is like designing sites in the past, when the download speed of a page was of primary concern. Although today most mobile connections are quite fast, some mobile phone users pay by the minute for Internet connectivity; thus, it's in their best interest to access sites that download in short order.

All of this plays into simplicity, a major component of all good mobile websites. Regular websites offer a host of options to visitors, as well as graphics, images, navigation menus, Flash elements, and JavaScript elements that add functionality and style to the page. But where mobile websites are concerned, the aesthetic elements have to work hand in hand with practical concerns. On a small screen, there's no room for overblown design elements, multiple navigation menus, or a great deal of text. Unlike regular websites, mobile sites also require that you accommodate the technology that users have to view them.

The language of the Web is HTML, or HyperText Markup Language. For older mobile browsers, the language is WML, or Wireless Markup Language. Today smartphones and mobile browsers generally make use of XHTML-MP. The differences between HTML and XHTML-MP aren't too radical, and anyone with HTML experience should be able to figure out XHTML-MP with little difficulty. There are other requirements that mobile websites have to accommodate to make sure that their elements display correctly on all of the many mobile browsers on the market. Some of the features they require are the following:

- very simplified navigation
- layouts specified using Cascading Style Sheets (CSS), not tables
- compact, efficiently written content
- a color scheme that is consistent across all browsers
- reduction in bandwidth-heavy elements, such as pictures, videos, and audio
- navigation options such as "back" and "next" on every page

Most mobile websites are built around a utilitarian aesthetic. They aren't the places to show off your design department's abilities or to debut new and untested features. When people surf the mobile-friendly Internet, they're generally looking to find the information they want as quickly as possible, and that information is usually not for pure entertainment. Efficiency is always the first order of business with mobile sites.

For example, for library sites, most users will primarily look to browse the catalog and reserve items from the stacks, to call the library, and to get directions to different branches. This means that the mobile site should be integrated with the library's database to allow customers to log in and access their account.

Another concern in developing mobile websites is space. A smartphone, useful as it is, has a small-format screen. This means that you must maximize the space available and not create a mobile site that is useful only to those users who have relatively large mobile screens.

It's also important to see the actual phones and know what they look like. Box 2 provides the range of dimensions for most smartphones; the average phone has a screen dimension of 320 × 480 (about the same as a playing card); ideally, that is the resolution that you should aim to develop your mobile website for.

BOX 2 ▪ Dimensions in Pixels

- **iPhone:** 320 × 480
- **Blackberry:** 160 × 160 to 480 × 360
- **Android SDK:** 320 × 240 (because different phones use the Android platform, here the dimension is the standard resolution in the developer kit)
- **Generic Windows Mobile:** 240 × 320 (standard resolution, but varies from phone to phone)
- **Nokia:** 95 × 95 and higher (varies from phone to phone, but 240 × 320 is standard)
- **Palm Pre:** 320 × 480

SURVEYING USERS

Before you continue with developing an app, it's a good idea to understand who your patrons are and what kinds of mobile services they want. For example, what percentage of patrons have smartphones? What kinds of smartphones do they have (many people don't know which kind of phone they have, so it's a good idea to ask to see their phone)? If they don't have smartphones, do they use the Web on their feature phones? Most web analytics sites (e.g., Google Analytics, StatCounter) will indicate users' resolution, which lets you figure out whether they are using a phone to view the site. But this doesn't always help because these sites let you know only whether users are accessing the site via their phone, not what kind of phone (e.g., iPhone, BlackBerry, Droid).

A lot of this book focuses on the iPhone, because when people talk about wanting to learn how to make apps, they are usually referring to making them for the iPhone—the holy grail of app development. Depending on your patrons, however, you might want to consider starting on another device. There is no point developing an iPhone app if most of your smartphone-carrying patrons have BlackBerries.

If you decide to continue with developing an app, then you have to ask, What do your patrons want? Access to the library's catalog? Photos from events? A calendar of events? GPS tracking to find the nearest library or book? To know when the next library computer is available? Instant messaging with a librarian? It's important to consider that what librarians find important is not necessarily what patrons find important.

TRAINING STAFF

The library might face resistance early on from staff. Many people don't like change, so to avoid resistance, it's useful to get everyone on staff involved as early as possible. Another good idea is that when you survey your patrons, you can also survey your staff to find out who already understands the technology and wants to jump on board and who will need a little bit more help. Also, it helps to show, not tell. Basically, that means that it's better to show a scene than to simply talk about what is happening. So, if you simply tell staff what the library wants to do, many might not see the necessity of an app, but if you show them the app and what it can do for the library, they're more likely to get excited.

How do you show? The best way is to carry out a survey on an iPod Touch or Android tablet. SurveyMonkey and other popular online surveys are great tools for collecting data, but actually placing a device in the hand of a librarian unfamiliar with such devices will help him or her become more comfortable with using the interface. These devices also are small, so they're easy to store. And for surveying patrons, you can buy an iPod lock (e.g., the Targus Defcon Notebook/iPod Lock Combo costs $39.99) or lock an iPod to an unattended desk for patrons to fill out the survey.

The iPod and Android tablets also use the same kind of interface as the iPhone, and there are plenty of survey apps that you can purchase for them; best of all they are relatively cheap (less than $200). The Samsung Galaxy and Dell Streak tablet have received a lot of media attention, but there are many other tablets available—there are many Android tablets available on eBay for less than $150.

You can use Survey on the Spot (free for iPhone) or Askdroid and ODK Collect (free for Android) to create a survey and then have librarians use the app to ask patrons a series of questions. Or use Tally Counter (free for iPhone) or CodeArk Tally Counter (free for Android) to count the number of people using smartphones. More powerful (and detailed) tally counts are also available for a cost: Tallymander ($3.99 for iPhone) and Advanced Tally Counter Pro ($0.99 for Android) let users count several different stats at once (e.g., types of questions patrons ask at the reference desk).

The results of the surveys, in most cases, won't be as powerful or specific as those you would get from sites like SurveyMonkey, but that doesn't matter; what matters is your letting people who don't regularly use the device know what it is. Even though people might resist change, almost everyone loves to play with gadgets!

Many times libraries implement technological innovations that don't take off because the library staff members don't promote them to patrons. But in surveying staff and patrons on an iPhone or tablet, even though you might not convert anyone to a smartphone user, you will at least give people a better understanding of their importance.

2 DEVELOPING A MOBILE WEB APPLICATION

Once you know who your users are, it's time to start developing. This chapter and the following ones provide a crash course in mobile development. We'll review the essence of HTML, CSS, and JavaScript programming, and we'll learn the very basics of what it takes to develop a mobile website.

SETTING UP THE DOMAIN

Mobile websites are housed in a separate part of your server from your main site. In most cases, the site name will simply have "mobile" added on at the beginning of the domain. For example, the mobile version of the URL YourLibrary.com would be Mobile.YourLibrary.com. The "Mobile" directory holds all the files required to display your mobile site. This type of organization also makes things easier on your information technology staff, because the "Mobile" directory is a convenient way to deal with a site.

Once your domain is set up and ready to go, the hard work begins.

UNDERSTANDING THE PROCESS

Most of the process of designing a mobile website involves simplifying and reformatting the information and features that appear on your main website. This means that, to build a mobile website, you first need to prepare the text and images for the site so that they fit the site's format.

MOBILE WEB VERSUS STANDARD WEB

Everything on your mobile website will be slightly different from how it appears on your regular website. This is inevitable. Although there are some features that your users will expect from your website, some of what you offer on your main site is impractical for mobile users. It's important to remember that you're trying to replicate the functionality and content of your regular website as much as possible on your mobile site. You're not simply duplicating your main website and putting it in your mobile directory.

Your mobile website, therefore, should largely be its own entity—it will have some features from your main site, but you'll abandon others. You can start building your mobile site by streamlining your regular website.

NAVIGATION

Most websites feature navigation menus with many options on the side or top of the page. For mobile sites, you'll need to reduce the number of navigation elements down to those that are the most practical and necessary for the patrons visiting your site. You will also want to add a "Back" and "Forward" button on every page. Some mobile browsers don't have these buttons built in, so it's easy for surfers to end up at a dead end on the site, which can be extremely frustrating.

The first thing that should appear on your mobile home page is the name of your library and your location. Your telephone number should also be one of the first pieces of information that your visitors see. Many visitors to your mobile website will simply opt to call and ask their questions directly to a librarian. You can make this easy for them by inserting a link that allows visitors to dial your phone number directly from the page. To insert the link, which should appear at the very top of the page, there is a fairly simple code that will display your phone number and allow visitors to dial the library just by clicking on the link:

```
<p align="center">Call Us:
<a href="wtai://wp/mc;5555551212">555-555-1212</a><br/>
<a href="wtai://wp/ap;5555551212;City_Library">Add Us to Your Phone Book!</a>
</p>
```

By clicking on Call Us, patrons can dial the number listed. Notice that the unformatted phone number appears at the end of the line that starts "wtai" in both cases. In the second link, "Add Us to Your Phone Book!" the tag "City_Library" determines how your number will be added to the user's phone book. Providing the ability to both dial your phone number right from the mobile home page and to add your number to the visitor's phone book with just a click is a huge convenience for your patrons—and it definitely has a wow factor.

Navigation buttons on a mobile website need to be very simple. In most cases, users prefer to move through your site using navigation menus. Entering text is difficult on some phones and, therefore, users tend to avoid doing so whenever possible. Also, your navigation choices should be intuitive, and ideally, it should take a minimal amount of clicks for patrons to find the page they're looking for.

Keep the shapes of your navigation buttons as simple as possible and use fonts that are bold and readable. Some mobile screens have very low-quality graphics, so make efforts to accommodate users of those devices when you're designing your site. For example, text links are preferable to graphics links.

The home pages of some mobile websites are little more than a small section of information with links provided underneath the main content. For examples of the effective use of this design, take a look at the mobile websites of some of the larger retailers—many of them follow this design.

For libraries, it also makes sense to organize navigation menus following the conventions for organizing books. For example, if patrons can browse the card catalog on the site, then a patron trying to find an Oscar Wilde novel might follow the navigation series "W_Wi-Wo_Wilde, Oscar" to find to the desired records. Each element of this navigation could be a different button—"W," "Wi-Wo," "Wilde, Oscar"—which would allow users to easily reach their destination without typing anything into the interface.

You can also use header tags to make your content easy to navigate. Headers are preferable to long text blocks on mobile websites. For instance, convert options such as

```
<p>You can browse our card catalog by clicking below</p>
```

to

```
<h1>Browse Card Catalog</h1>
```

CONTENT

The content for mobile websites needs to be much different from that of standard websites. For example, the content on mobile websites should be much briefer than on standard sites. Think about it like this: a mobile website page is to a normal website page what a card catalog record is to the book it represents. Your mobile pages have to be designed for people who are surfing for different reasons but whom you can cater to in the same fashion.

Most of the users of a library's mobile site will seek out very specific information. It doesn't matter what they're doing at the time that they're browsing. They might be rushing from class to the library, or they might be sitting in a coffee shop in no particular rush. Both kinds of users, however, want to be able to find what they're looking for in a reasonable amount of time. This means that content on a mobile site can be drastically cut back in comparison to that of a regular website.

The home page of a regular website often contains news, notices of updated pages, and much more. Some contain an "About Us" statement or other public relations information. None of this is really necessary on a mobile website. Regular websites are designed to be something of an experience in and of themselves, but mobile sites wow users with their functionality.

So, in designing your mobile site, reduce your content to the bare minimum. You can even probably eliminate altogether most of the material about your organization that appears on the home page of your regular website.

FORMATTING CONTENT

Even though small screen sizes are always a concern with mobile websites, there are several larger screen resolutions in use among smartphones. Some of them are more common than others. At the smallest end, 128 × 160 is fairly common. This screen size applies for most of the cheaper phones on the market. At the high end, 320 × 480 is fairly common among newer-model smartphones and other more sophisticated mobile devices. These screens are of different shapes, which makes it difficult to predict the look of a given mobile website on any particular phone. Designers have ways of getting around this, though, and many companies offer several websites—one for standard computers, one for smartphones, and one for feature phones.

Designs for mobile websites should incorporate many linear elements. Keep your lines simple and your design functional. People using mobile devices with very small screens will have a hard time viewing a page that is crowded with large elements. If you position elements on your page on the basis of how they look on a specific device, you may be disappointed to find that the page looks much different when viewed on a different device. The more basic and linear the page design, the easier it will be for more of your visitors to use.

Remember that there is little flexibility in horizontal movement on mobile websites. You should orient your content so that users naturally move from the top to the bottom of the page.

INTERACTIVITY

People using mobile devices generally aren't using a mouse. Instead of a mouse, however, smartphones have many different features that users can activate by tapping on the screen, and some of them have drag-and-drop functionality.

One of the conventions of web design is to avoid including elements that only users who have the most advanced equipment can access. In design parlance, this is sometimes called accessibility. To make your site accessible to everyone, you should minimize the elements that depend on the user having touch-screen functionality.

For example, on navigation menus, most users will navigate with "Up" and "Down" buttons to move from one option to the next. This means that some of the elements common to regular websites—drop-down menus, links in discrete areas of a site—are not always suitable for mobile websites. They can be downright frustrating to users whose devices don't allow for easy interaction with those particular elements.

Arrange navigation elements logically from the standpoint of someone using "Up" and "Down" buttons to move around the site. Doing so doesn't diminish the functionality of the site at all for users with more advanced devices, but it makes life much easier for those who have older phones.

If a lot of your users don't have or aren't likely to have newer mobile devices or smartphones, you might want to consider offering your mobile site in WML. Any mobile device can render pages written in this language. Not all older devices will be able to render the XHTML-MP elements that you may intend. One difference between mobile devices and personal computers, however, is that mobile devices tend to have shorter lifetimes and that people tend to upgrade them much more frequently than they do their computers.

IMAGES

Mobile devices, and especially the newer generation of smartphones, can accommodate images on their screens. All mobile devices have two characteristics, however, that make them inherently bad venues for large images and animations:

1. They have to accommodate very short download times.
2. The screen size makes viewing large images difficult.

As much as possible, eliminate images from your site. Then, you have to compress any images that remain, such as logos or banners. Both JPEG and GIF formats are excellent for mobile devices. As much as possible, however, your site should be text based to optimize download times.

BRINGING IT ALL TOGETHER

Most WYSIWYG HTML editors today can accommodate mobile pages, and they usually include several different templates. Use one of these templates to lay out your site. You don't have to move beyond basic layout at this point, but concentrate on arranging things in logical order and, if you have a test server, test the site on different mobile devices.

It's vital at this point to make sure that your graphics display as intended and that your navigation menus are sensible and easy to move through on a mobile device. To add style to mobile websites, it's best to use CSS, which reduces the size of each page and allows you to change design elements across the whole site by altering only one file.

WORLD CONFERENCE ON SOFT COMPUTING

The World Conference of Soft Computing (WSC3) conventions on HTML design are very important when designing mobile sites as well. Check all of your code against the conventions to make sure it's valid. You can use online validators or the ones built into most of the commercially available WYSIWYG editors. Because of the tremendous variability in how a page displays from one phone to the next, it's important that your site operates as predictably as possible, and that means making sure that your code is valid.

3 USING CSS

This chapter is a crash course in Cascading Style Sheets, or CSS; if you already know how to use CSS, then this chapter will be a refresher. For a comprehensive view of CSS, I recommend that you check out *CSS: The Missing Manual*, by David McFarland (published in 2006 by O'Reilly Media). What's important for you to know is that, although CSS is certainly useful and beneficial in developing a mobile website, it is not essential. If you don't use it, you can still develop a fully functional mobile website. So here are the basics.

The CSS tool was created to solve a specific problem with HTML, and it has since proved among the most useful of all Internet innovations. In HTML, every piece of text can have its properties specified individually. For example, you can specify that a certain line of text is red, underlined, and 12 points. If you want to, you can specify these properties separately for each letter on a page. This method, however, vastly increases the page's size and, more important, creates a lot of extra work for the designers, who have to specify each instance of how a link should look, how paragraph text should be displayed, and so forth. CSS eliminates this hassle.

A Cascading Style Sheet is really just a text file. It is usually separate from the main HTML code and, on the page, there is a call for that style sheet written into it that determines which style sheet will be used and, thus, how the page will look. Using CSS with mobile devices requires the inclusion of some tweaks to the codes language. This led to the development of Wireless CSS (or WCSS), which is tailored to mobile sites. If you understand CSS fairly well, you should have no trouble understanding WCSS.

To understand how CSS works, think of your website as consisting of two main parts. The first part is the content of the site, which is specified in the XHTML-MP document that you created on the basis of your HTML site. The second part of the site is the WCSS component, which determines how your site actually looks when it's displayed in a browser. With WCSS, you can also add some functionality and text effects.

HOW POWERFUL IS WCSS?

WCSS is enormously powerful. Basically, one text file controls the entire look of your site, including every element on the page.

The syntax of CSS and WCSS are easy to understand. A selector identifies every element on your page that can be tagged with HTML (e.g., H1, P, A). These selectors have properties, such as color, size, ornamentation, and so forth, and all of those properties have values, such as red, 12 point, or underline. These properties are entered in a very simple format. For example, if you want every H1-level heading on your site to be green, the WCSS code is as follows:

```
h1 {color: green}
```

You can also use the number code (i.e., hexadecimal code) to specify color. For example, for all your paragraph text to be black, you would use the following code:

```
p {color: 000000}
```

You can use any HTML tag as a selector. To specify multiple properties for any given selector, separate them with a semicolon. In most cases, designers also place each property on a separate line, with the closing curly bracket on a line of its own. For example, for all paragraph text to be 8-point black, and specified as such in the style sheet, you would properly express it in the following way:

```
p
{
font-size: 8pt;
color: black;
}
```

This formatting of the text file does nothing to change how the WCSS renders your page. It does, however, make it easier for other designers to pick up your work and to see how you have set up everything—where mobile web pages are concerned, adhering to standards is vital.

EXPANDING THE FLEXIBILITY

ID selectors allow you to be more specific about the look of your page. An ID selector refers to a single instance of an item. So, for example, if you want one specific text element to render differently than all others, you could specify it in the CSS by using the following line (for the purposes of example, we'll name this class "special"):

```
#special
{
color: orange;
}
```

Class tags allow you to specify this for an entire class of recurring elements on your page. For example, you could make all of the text that specifies library policy its own class and set it off in red as follows:

```
.policy
{
color: red;
}
```

The power in this is that you could change any property of every instance of the specified class, or type of text, with no more than a few keystrokes. This has some creative uses as well. For instance, for the Fourth of July, you can change your mobile site's color scheme to red, white, and blue; during Halloween, you can make it orange and black. You can make special style sheets and keep them on hand for various occasions, or you can revamp your website by changing just one file.

USING CSS INTELLIGENTLY

CSS is primarily useful because it reduces the amount of work it takes to develop and maintain a site by reducing the complexity of making changes across your entire website. To use CSS effectively, you need to keep your style sheets properly formatted and your classes and IDs limited to elements that merit those distinctions. For example, the more you separate common elements of your site into separate IDs and classes, the harder it will be to make large-scale changes. As much as possible, keep your site consistent with respect to the look of the headers and other elements; doing so will make it easier to make broad changes.

You can also use CSS to format links. This is a good way to reduce your site's dependency on graphics, because you can get rid of JavaScript rollover images and other resource-consuming elements that add little in the way of functionality. Simplifying your site with CSS can reduce page-load times; keep a consistent look throughout the pages; and make it easier to put new ideas into place when you want to change the site around a bit, whether those changes are permanent or temporary.

BROWSER SIZES, CSS, AND POSITIONING

You can use CSS to determine how all the elements on a web page are positioned in relation to one another, in relation to the corners of the screen, and in other ways. Because of this, CSS more easily accommodates the needs of diverse visitors than HTML does.

CSS VERSUS TABLES

One of the easiest ways to position elements on a page in HTML is to use tables. The use of tables dates back to when the language was first designed, and older pages tended to use tables heavily. However, the use of tables has several disadvantages for mobile websites.

Tables add information to the HTML page, as their dimensions and other properties have to be defined in the document. For mobile sites, reducing the size of files is paramount, and CSS allows for this by keeping that information in an external file, not in the document itself.

Tables are also clumsier with respect to laying out objects. When the specified dimensions of objects are too large to be accommodated by a specific screen, tables tend to crowd objects together or to bleed them off the side of the page. When objects are too small, tables waste valuable page space. In this regard, CSS offers superior positioning options.

ABSOLUTE POSITIONING

Absolute positioning means specifying that elements of the page appear in specific areas of the screen, regardless of the screen size. Absolute positioning can be used to accommodate all users because the page elements display correctly on even the smallest of screens. A well-designed page will always look good on larger screens, but there will be more unused space on the display. The CSS for setting an element's position in absolute terms is written as follows:

```
h1
{
position:absolute;
left:25px;
top:50px;
}
```

This code tells the browser to display anything labeled <h1> at 25 pixels from the left of the screen and 50 pixels from the top. That way, no matter the user's screen size, the mobile device will display content tagged "h1" in that position.

RELATIVE POSITIONING

Relative positioning allows you to specify a container and position the page elements within that container. This type of positioning is sometimes used on sites to better control for smaller resolutions. For example, if you have a borderless background, you can frame your page within the confines of an 800 × 600 monitor resolution and position all the elements on your page within that container.

FLOATING

Most web designers use floating to position elements, as this simply places elements preferentially to the right or the left of the screen. If there isn't room to accommodate the elements on the screen, then they can appear in a logical order that flows from top to bottom. This accommodates the linear design requirements of mobile websites, which makes this method very convenient.

MAKING YOUR FIRST CSS DOCUMENT

All you need to make a CSS document is a text editor. The selectors, their properties, and their property values are expressed as detailed above, and there is little complexity to deal with.

Embedded style sheets appear at the top of the document, within the header. To change any given page that uses such a style sheet, you must change each style sheet individually.

Linked style sheets are stored in a separate file, which offers the most convenience when you need to make massive changes to a site. Your style sheet is called for in the header of your document. The tag looks like this:

```
<link rel="stylesheet" type="text/css" href="your_style_sheet_here.css" media="all">
```

If you're creating a style sheet manually, simply give it a name and save it in a directory on your website. In the header of your web page, paste in the previous tag. This will link the page to the style sheet, and the style sheet will then determine the look of the page. If you want to change the look of the page, simply specify a different style sheet in the link.

KEEPING IT SIMPLE

Remember that mobile devices have considerably less display area than computers. When you start putting together the layout of your site, keep it very simple.

A popular page design for libraries makes use of three elements that allow users to make the most of the mobile presence:

1. A banner with a phone number
2. Text links to other pages
3. A search field

For most libraries, these three elements make for an excellent home page. There are few elements to arrange and, as long as the elements are designed with forethought and correctly positioned, they should display effectively in every mobile web browser. Because there is so little on the page, it will also load quickly. You can even eliminate the graphic for the library logo altogether and replace it with a simple text link. This makes the site more accommodating for those who have older-generation mobile devices.

Remember that the more your navigation relies on simple text links, the less space you will lose to graphics. Of course, you can use CSS to make elegant text links, including ones that provide rollover effects and other visually interesting elements—this is yet another reason to employ the CSS tool whenever possible.

If you want to add graphical menus and other functionality to your mobile site, you'll need to learn a bit about JavaScript, which is a ubiquitous web language. However, it is also complex and takes a qualified programmer to write it. Understanding how JavaScript works and what it offers, however, is the last step in determining how to put together your page.

4 JAVASCRIPT AND MOBILE WEBSITE DESIGN

This chapter is meant to give you an overview of what JavaScript is and how it functions—not to teach you everything you need to know; Danny Goodman's *JavaScript Bible* (published by Wiley in 2010) is the most comprehensive book you'll find on the subject. As with CSS, JavaScript isn't required for developing a mobile app, and it is less essential to designing a good app.

JavaScript is a client-side scripting language that allows users' computers to execute specific functions. In a less technical sense, JavaScript tells a device to perform various tasks. For example, some search pages automatically place your cursor in the search field for you. The cursor doesn't go there on its own; the site often uses JavaScript to place the cursor there.

JavaScript does not actually contribute a great deal to the layout of your page. There are no websites, mobile or otherwise, that are built on JavaScript alone. It is, however, an important element for adding functionality to your site, and to some extent, it can add some visual appeal.

GETTING JAVASCRIPT

JavaScript is extremely popular with both designers and programmers of regular websites. The reason for this is the power that it allows designers to wield over the functionality of their pages. This is especially important for libraries, because using tools such as these can cut costs considerably. For example, JavaScript allows coders to use a much better known and more frequently used code and not take the time to learn a phones-native language to do essentially the same thing.

You can download libraries of various JavaScript resources and include them in your site, which is often easy to do and doesn't take any real programming knowledge. Usually, you only need to add a small script into the web page.

For mobile sites, though, JavaScript brings up some significant compatibility issues. Moreover, by adding code directly to your page, you increase its size. For users with browsers that lack JavaScript compatibility, that amounts to downloading extra data that offer no benefit at all.

THE BASICS OF INCLUDING JAVASCRIPT

Although you can add more advanced features with JavaScript, its support among mobile devices is spotty at best, and you should test it thoroughly before deploying the site to patrons.

JavaScript is a scripting language that works on any computer. It is included in the HTML of a page and allows browsers to perform a variety of functions. The code appears in the following format:

```
<script type="text/javascript">
document.write('Your message here!'); </script>
```

In the context of a full page, the script would be expressed in the following way:

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write(' Your message here!');
</script>
<noscript>
```

Your browser either does not support JavaScript, or you have JavaScript turned off.

```
</noscript>
</body>
</html>
```

This script would tell the browser to write "Your message here!" on the page in the position determined by the script's location in the HTML. You can specify the font and other attributes of the text, just as you can for any other element on an HTML page. JavaScript can get very complex, though, and some large scripts constitute large parts of the content of some web pages.

In the event that the visitor's browser doesn't support JavaScript, the site visitor would receive the message "Your browser either does not support JavaScript, or you have JavaScript turned off."

Some scripts are contained in the header of the HTML document and are called at the point in the body where their output is desired by a second code. These scripts can perform functions such as displaying dynamic information, providing messages based on user actions, or date and time. They can also be used to process some types of information and to perform simple functions. JavaScript can be custom coded, but even professional designers tend to make heavy use of the free libraries available on the Web, as those resources address the most common JavaScript issues.

JavaScript also is used in some cases to provide better accessibility to web page content, such as allowing users to make the page's text bigger.

JAVASCRIPT AND FORMS

JavaScript is used heavily for the handling of forms. For libraries, these might be forms that allow users to do a quick search of the site or of the card catalog or to update their library account information. There's no reason that you can't provide these functions on a mobile site; to perform them, however, the user has to provide you with specific information. This can be more resource intensive than it sounds.

If a user submits a form with missing information to the server, the server has to process the form, figure out what's missing, and send it back to the user with the missing elements indicated. For mobile users, this can use up their airtime as they wait for all this to happen. JavaScript can inspect a form as it's filled out and, if there's something wrong, point that out to the user before they send it. This saves them time and saves your server's resources.

The drawback of using JavaScript is that many mobile web browsers do not support it at all. Therefore, adding JavaScript to your pages benefits only some of your patrons and may make the site work more poorly for the rest of them. So, designing with JavaScript is all about using it intelligently.

WHERE TO USE JAVASCRIPT

If accessibility is an issue for your library, then you should have a version of your mobile website that does not employ JavaScript at all. Older phones often cannot execute JavaScript, which will make some of your site's features unavailable to some users. It's probably the case, though, that most users are using newer-generation phones.

On newer-generation phones, there are several ways you can enhance their experience on your site. For example, you can add code to your site that will detect the size of users' screen and change the style sheet used to render the site accordingly. This means that you can have a developed, ornate version of your site for the smartphone crowd and a simpler version for those using older phones. Combined with CSS, JavaScript can greatly increase accessibility in this way.

The way this works is fairly complex, but the overall concept is easy to understand. Whenever a mobile device visits your website, you can query as to what kind of device it is. When the information is sent back to the server, the server looks up the device on a list of mobile devices and matches it to the right phone on the basis of the dimensions of the screen. The phone information returned includes the screen dimensions, which the server then uses to send users the right web page for those dimensions.

JavaScript is often used to make pop-up windows—the useful kind, not the kind that was the source of much irritation in the past—animated menus, and other features that add visual interest to the page. On mobile sites, these things are best avoided, mostly because users navigate mobile websites differently than they do regular websites.

If you have different versions of your website for different devices, then you can offer some JavaScript functionality for those devices that can run it. This can make sure that devices that cannot process JavaScript do not receive versions of your page that are unusable.

Mobile devices are increasingly making use of nearly full-fledged browsers rather than specific apps for surfing the Web. This means that support for JavaScript among mobile browsers is increasing, and it will likely become a more common element in mobile web design.

Setting up a JavaScript element to detect smartphones can be useful, but it might not always work out as you plan. Some mobile browsers don't identify themselves correctly or are set up to identify themselves as a different type of browser, such as Internet Explorer. In addition, not everyone with mobile devices that support JavaScript will have the right feature turned on. In some cases, users block JavaScript altogether to enable faster page-load times and to keep their browsers more secure. These are all considerations when designing with JavaScript.

JAVASCRIPT MISSTEPS

One of the most popular uses of JavaScript is to create menus. In some cases, menus can be confusing to users. Users on mobile devices tend to navigate with buttons, not a mouse, so that's another reason to avoid going overboard with menus. For mobile devices, you should avoid rollover mouse effects (e.g., a button lights up or shifts position when you roll the cursor over it). On mobile devices, such menus require too much code and too many resources.

It's also important to make sure that your JavaScript applications are compatible with mobile devices. For example, not all mobile devices might have a substitute for the mouse, which means that the device might ignore altogether any events that are linked to the mouse, which may cause problems with the page. There are variations on JavaScript cursor actions designed to accommodate smartphones, but they are not universal. For example, the current generation of smartphones does not reliably support the drag-and-drop functionality that you can add to web pages with JavaScript.

There is expanding support for more mobile browsers, however. Free JavaScript libraries have more and more features with every revision that can provide added functionality to smartphone browsers. Some of the newest innovations include the following:

- increasing support for touch screens
- support for scroll bars being placed within browsers
- support for more sophisticated animation

Because there is so much variation among mobile devices, however, there is still great inconsistency in JavaScript compatibility.

JAVASCRIPT APPLICATIONS

JavaScript is oftentimes used to create browser-based applications, such as measurement converters and simple games. However, these applications can become very complex, especially on very developed websites that offer a great deal of user functionality.

On mobile websites, applications can be a problem because of compatibility issues. Although they're great for incorporating into your main website, the mobile versions of most major websites eliminate JavaScript applications altogether. The potential benefits of using complex JavaScript applications are simply not reliably delivered even by the most advanced smartphones, although compatibility is sure to increase with time.

IS IT VIABLE?

Presently, mobile devices do not universally support JavaScript. This means that, in terms of accessibility, a library would have to offer a JavaScript-free version of any page it currently publishes on its website to enable the widest variety of users to access it. Functions that are completely dependent on JavaScript will not be available to those whose phones cannot support it. Some phones do support JavaScript, but it may not function as intended, particularly for user-action-triggered events, such as mouse-overs and hovers. This may make JavaScript unsuitable for some libraries and for the majority of their patrons.

However, you can use JavaScript in limited deployments and offer your visitors increased functionality and more convenience, especially in handling forms and other basic needs.

You can quite easily and inexpensively test JavaScript's suitability on your website. Because of the many free JavaScript libraries available and because anyone who can edit HTML can install the scripts on a page quite easily, you can check to see whether adding a feature based on JavaScript provides a better experience for your clients. Make sure you test it yourself on various devices: smartphones use different browsers, so check the functionality of your page on all smartphones.

Again, the most important element of mobile web design is simplicity. However advanced and developed your main site is, you'll have to strip it down to the essentials to make it useful for those who prefer to access it on the go. In some ways, your mobile site mirrors what you have on your main site, but it is better thought of as a separate entity altogether. It might contain a great deal of the same content as your main site, but it will be directed toward providing the fastest, most efficient, and most no-nonsense user experience possible. However, as mobile web use rapidly increases, there will be a significantly increased demand among library users to have library resources available on mobile devices.

WHERE CAN LIBRARIES USE JAVASCRIPT?

If you are trying to decide whether you should learn JavaScript, you first need to ask how the library can use it. Two common uses for JavaScript is in developing forms and providing a chat-based reference service. It also is nice to have a place for patrons to add and send comments, or to create a library application page where patrons can apply for a library card from their phone, or to find out when the next computer is available, or to create a mobile-friendly catalog. But whether it’s worth the time to learn the language is up to you. For chat-based services especially, there are relatively cheap non-JavaScript options as well, so it might be in your interest to use one of those services and save yourself the trouble of learning the language. If you don’t already know Java programming, then you will probably be better off designing an app without it and looking into adding in JavaScript to the app later.

5 PHONEGAP

I decided that I was going to develop an iPhone app several months ago. It was the latest way to make money, so I figured, Why don't I jump on the bandwagon? I checked out some books on iPhone development and watched a few YouTube videos. After about ten hours of research, the furthest I was able to get was to download the iPhone SDK. I've had my share of programming courses in college, and I was developing websites before most people even had computers with modems. I say this so you understand just how complicated programming apps can be.

I'm not an expert in programming language; I understand the theories behind it, but I know little about how to actually write code. When mobile development was just getting started, I was out of luck. However, a relatively new program called PhoneGap is helping bridge the divide for people with no or limited programming skills.

PhoneGap got started in 2009 at the iPhoneDevCamp. The program allows developers to create a page like they would for the mobile web—with HTML, CSS, and JavaScript—and quickly convert it to a language that's supported as a native app. Although PhoneGap is simple enough for anyone to use, companies like Wine.com have successfully built powerful apps with its framework.

The best thing about PhoneGap is that it's free and open source. But regardless of whether you will develop a mobile app, PhoneGap is a valuable resource if only because it has simulators to let you see what a website looks like on a particular phone without actually having the phone.

PHONEGAP FEATURES

So, exactly what does PhoneGap do? In short, the PhoneGap software helps you turn a website into a mobile app. In theory, instead of learning several different mobile programming languages, you only need to know HTML. In practice, it's best to also know CSS and JavaScript, but knowing HTML will get you started.

PhoneGap also lets you take advantage of the features of individual cell phones (e.g., geolocation, accelerometer). Table 5.1 shows a list of all supported features of PhoneGap.

Table 5.1 Supported Features of PhoneGap

	iPhone	Android	BlackBerry	Symbian	Palm
Geolocation	X	X	X	X	X
Vibration	X	X	X	X	X
Accelerometer	X	X	OS 4.7	X	X
Sound	X	X	X	X	X
Contact support	X	X	X	X	
Camera	X	X	X	X	
Multitouch	X	X			
Copy and paste		X			

What follows is a brief explanation of the PhoneGap features and examples of how they work in practice:

Geolocation: Geolocation is a feature in every newer-generation smartphone that enables the phone to use its internal GPS to determine its real-time location. What does that mean for an app? It could, for instance, let library patrons find out where the library is located in relation to their own position and help them get directions.

Vibration: Vibration lets you take advantage of the internal vibration setting available on phones. For example, developers could create an app that notifies users of alerts through vibration.

Accelerometer: An accelerometer is a feature first made popular by the iPhone but now available in nearly all phones. This feature rotates the screen when the phone is turned on its side.

Camera: All camera phones are built differently, and using the camera feature will likely result in some glitches.

A complete list of PhoneGap's features can be found online (http://wiki.phonegap.com/Roadmap/); the site is updated often and includes extra documents and discussions where developers can find support.

VIEWING YOUR WEBSITE ON PHONEGAP

Now it's time to see how your library's website looks on a mobile device. If you have a mobile phone, then this is easy; but just because your website looks great on Android doesn't mean that it will look equally great on an iPhone or a BlackBerry.

Figures 5.1 and 5.2 show two websites from the *New York Times*; figure 5.1 shows how the site looks on the iPod and iPhone as a regular site (not a mobile one); it's nearly impossible to read without zooming in, which isn't always practical for every user. Figure 5.2 is the *New York Times* mobile site, which is much easier to read on a smaller screen.

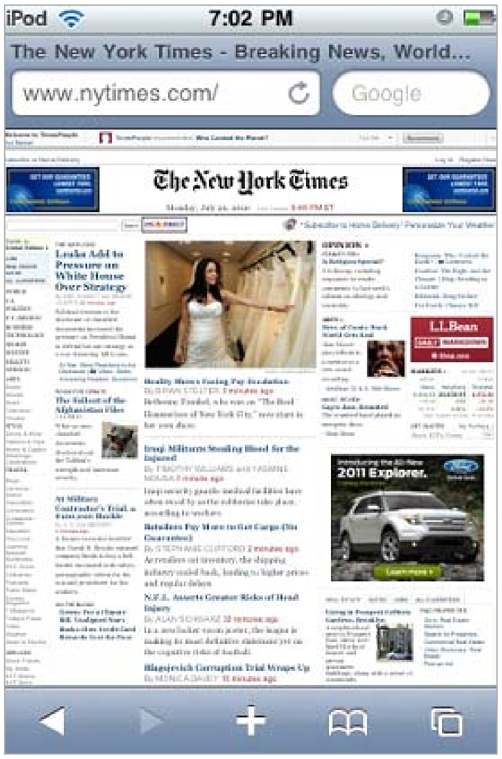
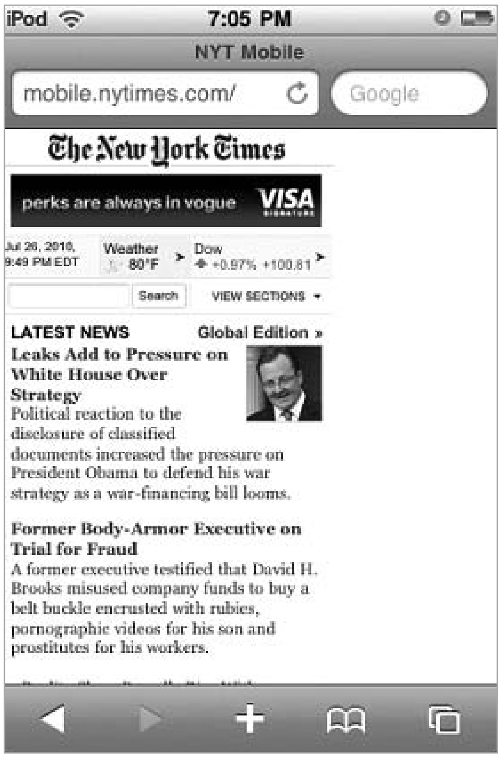

iPod
7:02 PM
The New York Times - Breaking News, World...
www.nytimes.com/
Google
The New York Times
Leaks Add to Pressure on White House Over Strategy
2011 Explorer
L.L.Bean

Figure 5.1



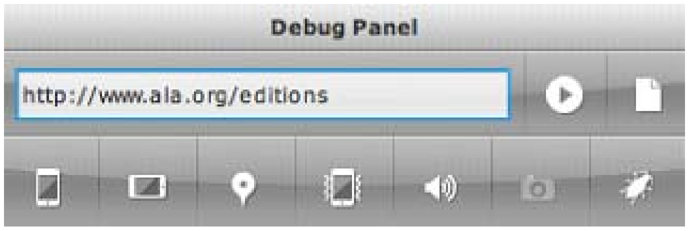
Leaks Add to Pressure on White House Over Strategy
Political reaction to the disclosure of classified documents increased the pressure on President Obama to defend his war strategy as a war-financing bill looms.

Former Body-Armor Executive on Trial for Fraud
A former executive testified that David H. Brooks misused company funds to buy a belt buckle encrusted with rubies, pornographic videos for his son and prostitutes for his workers.

Figure 5.2

You want to know what your mobile site looks like on every phone imaginable. PhoneGap makes this easy; it has cell phone skins for testing websites on nine different phones (iPhone, Android G1, Blackberry Storm, Blackberry Bold, Palm Pre, Nokia N97, Samsung Epix, Sony Ericsson Satio, Sony Ericsson Rachael).

To test a website, run the PhoneGap Simulator (www.phonegap.com/tools/). When the Debug Panel (shown in figure 5.3) appears, type in the website address in the address bar and then click the right-arrow button.



http://www.ala.org/editions

Figure 5.3

If you want to view the website using a different skin, then click once on Menu in the PhoneGap Debug Panel, select Skin, and pick the phone that you want to switch to (figure 5.4).

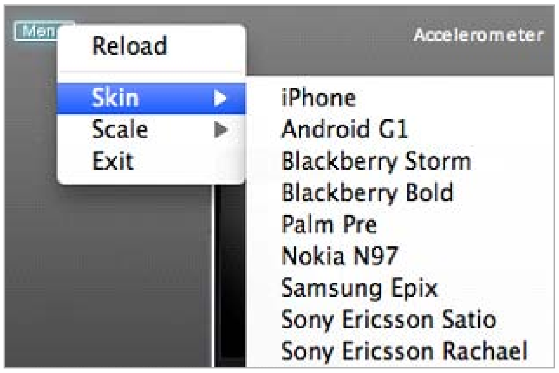


Figure 5.4

REDIRECTING TO A MOBILE SITE

If you decide to have a mobile version of your site, then you will have to decide how to get phone users to it. There are two ways. First, simply have two addresses that you give out: a computer address and a mobile address. You can tell your patrons to add "m" to the address when accessing the site from a phone (e.g., m.library.com).

The second way is a little more difficult. It requires that you add code into your page that tells the page to search for the user's browser. If they are using a mobile browser, then they will be redirected to the mobile-friendly site. You need to understand PHP programming to implement such a task. You can also buy the code for $50 (http://detectmobilebrowsers.mobi/).

IMPLEMENTING PHONEGAP

I won't go into detail here about how to get the apps onto every phone—it varies widely depending on the phone and there are often monthly updates. But you can readily find any information you need on each phone's website or check out the information provided at PhoneGap:

- **iPhone:** http://www.phonegap.com/start#ios
- **Android:** http://www.phonegap.com/start#android
- **BlackBerry:** http://www.phonegap.com/start#blackberry
- **WebOS:** http://www.phonegap.com/start#webos
- **Symbian:** http://www.phonegap.com/start#symbian
- **Windows Mobile**: http://phonegap.pbworks.com/Getting-Started-with-PhoneGap-(WiMo)

6
BUILDING YOUR FIRST NATIVE APP

CREATING A BARE-BONES NATIVE APP

This chapter will help you create a quick, bare-bones app that will run on the iPhone. I use the iPhone as an example because it is the device that most users will have for native apps—although Android is quickly catching up. The iPhone is also the easiest device to quickly develop apps for.

Despite some negative press about the iPhone app process being overly protective and the rejection of apps left and right, it is surprisingly easy and very friendly to PhoneGap's open-source framework. As long as you adhere to the iPhone guidelines, which aren't as strict as they are sometimes made out to be, you will have no problem developing your app.

The first thing you need to do before you begin developing any app, be it mobile or web, is to draft a concept of what you want it to include and look like. I recommend MockApp (www.mockapp.com), a free service that lets you use PowerPoint to develop a sample iPhone app, one that you can run on your device as a PDF. It's one thing to tell library administrators that the library needs an iPhone app; it's another thing to show them what the app will look like. Developing a sample app will let you quickly demonstrate what the app will look like and how patrons can take advantage of its features. As you develop the app, keep in mind that you don't want to have to update things too frequently.

I show here the app that I developed for this book (it's meant to look like an iPhone screen). The end of the chapter provides the code for the app.

Unlike a mobile web app, for which you have to be very careful with images, native apps can feature a few more graphics (see figure 6.1). All of the images I used for this basic app came from the Open Clip Art Library (www.openclipart.org), a great resource for libraries.

Box 3 shows the code used for figure 6.1 for your reference.

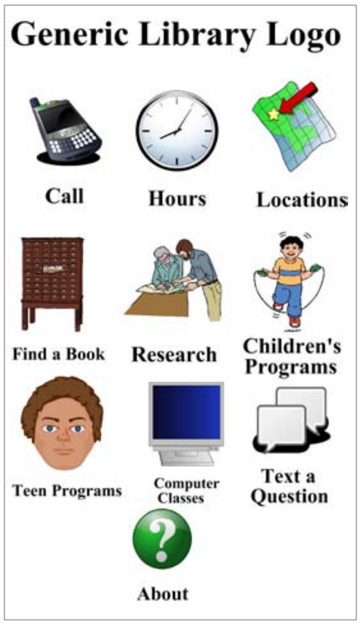


Figure 6.1

BOX 3 ■ Code for Library App

```
<HTML>
<head>
<title>Generic Library Logo</title>
<style type="text/css">
.style1 {
text-align: center;
}
.style2 {
text-align: center;
font-size: x-large;
}
.style3 {
margin-right: 0px;
}
</style>
</head>
<body>
<table style="width: 852px; height: 115px">
<tr>
 <td class="style2" colspan="3" style="height: 19px"><strong>Generic
 Library Logo</strong></td>
</tr>
<tr>
 <td class="style1" style="height: 23px; width: 283px"><strong><br />
 <img height="90" src="call.jpg" width="65" /><br />
 <br />
 Callstrong></td>
 <td class="style1" style="height: 23px; width: 283px"><br />
 <strong><img height="90" src="hours.jpg" width="90" /><br />
 <br />
 Hours</strong></td>
 <td class="style1" style="height: 23px; width: 284px"><strong><br />
 <img class="style3" height="86" src="map.jpg" width="94" /><br />
 <br />
 Locations</strong></td>
</tr>
<tr>
 <td class="style1" style="height: 23px; width: 283px"><strong><br />
 <img height="88" src="book_catalog.jpg" width="89" /><br />
 <br />
 Find a
 Book</strong></td>
 <td class="style1" style="height: 23px; width: 283px"><br />
 <strong><img height="90" src="research.jpg" width="100" /><br />
 <br />
 Research</strong></td>
 <td class="style1" style="height: 23px; width: 284px"><strong><br />
 <img class="style3" height="91" src="childrens_services.jpg" width="75" /><br />
 <br />
 Children&#39;s
 Services</strong></td>
</tr>
<tr>
 <td class="style1" style="height: 23px; width: 283px"><strong><br />
 <img height="103" src="teen.jpg" width="90" /><br />
 <br />
 Teen
 Program</strong></td>
 <td class="style1" style="height: 23px; width: 283px"><strong><br />
 <img height="92" src="computer.jpg" width="100" /><br />
 <br />
 Computer
 Classes</strong></td>
 <td class="style1" style="height: 23px; width: 284px"><br />
 <strong><img height="91" src="text_message.jpg" width="90" /><br />
 <br />
 Text a
 Question</strong></td>
</tr>
<tr>
 <td class="style1" colspan="3" style="height: 19px"><strong><br />
 <img height="50" src="help.jpg" width="54" /><br />
 <br />
 About</strong></td>
</tr>
</table>
</body>
</html>
```

So let's talk about this app. All the buttons are placed in what I considered the order of most importance to patrons. The most popular buttons will likely be "Call the Library" and "Hours." This is what will be the Home screen.

Generic Library Logo: Whenever possible, I recommend using the logo that patrons already know; try to also use similar colors as on your main web page. Remember, however, that users won't be normal computer users—people who access the app are frequently outside and their screen is harder to see. Colors are even more important on a cell phone than on a computer.

Call: When creating apps there are two kinds of buttons; the first is a hyperlink that takes you to another page. The second is an action button. Action buttons do exactly what the word *action* says. They might tell the phone that the user wants to call a number; when pressed, the phone automatically launches the dialer and begins to make a call (see the end of this chapter for the code used for the action button).

Hours: You can do one of two things with the "Hours" link: you can create a mobile page with hours of operation, which is the simplest and quickest approach, or you can create a script that counts down the number of minutes until the library either closes or reopens again.

Locations: Patrons want to know not just where the library is but also how to get there; this is where a mobile app can help. To keep things simple, I won't go into using GPS; if you want to know more, refer to the PhoneGap website tutorial (http://phonegap.pbworks.com/PhoneGap-Tutorial:-Get-GPS-Coordinates/).

Find a Book: Most libraries will be a little limited in the "Find a Book" option because, unfortunately, many vendors haven't yet gone mobile, so it's difficult to make a mobile-friendly version of your catalog. In the chapter on advanced features, I talk a little bit about a workaround to this. If your catalog vendor does not currently offer a web app, start asking why. Few vendors have caught on, and they won't until librarians continually ask for it.

Research: Many research providers have yet to go mobile, so make sure you encourage them to do so. One vendor that has gone mobile is Gale, which has a terrific app called AccessMyLibrary. The app lets patrons use the database from their phone if they are within a ten-mile radius of the library—and no log-in is required. Even if you do not decide to develop an app, you can promote this service.

Children's and Teen Programs: Obviously, you will want to reference the services you actually have in the library on this page; it is also possible, however, to put games on this page. There are a number of places you can go to get JavaScript-based games that you could implement on these pages; a Google search will give you literally thousands of pages—some are open source, some are paid. JavaScript, especially on the iPhone, can be tricky; make sure you test the page to see if the JavaScript code actually works. When you are getting code from the Internet to use on a mobile app, it can be hit or miss.

Computer Classes: The next chapter goes into greater detail on this topic, but one thing to consider is providing videos of different classes or two- to three-minute demos of how to do different things on the computer—or even a hyperlink to a helpful YouTube video on different computer-related subjects.

Text a Question: If you don't have a texting service at the library, get one! It's free! Google Voice is fine for most libraries. There's an easy code for an action button for this. Just add the following code to your app: <a href="sms:1234133424243">Text a Question</a>.

About: This is where you can explain the app's purpose and who to contact to report any bugs or glitches.

All the information on the app so far easily applies to web apps. Changing your app to an iPhone app requires only one or two extra steps.

The District of Columbia Public Library was one of the first public libraries to build an app for its book catalog; the library put the code for the app online, and it's free to copy (http://dclibrarylabs.org/projects/iphone/). You may not want to use it (or may not even like it), but it's a good reference to start with as you begin to learn about code.

PUTTING THE APP ON THE IPHONE AND IPOD TOUCH

To begin installing your app on the iPhone, you will need to download both the iPhone SDK and PhoneGap (on an Intel-based Mac computer) from the following websites:

iPhone SDK: http://developer.apple.com/iphone/index.action/

PhoneGap: www.phonegap.com

It's important that you save all of your mobile web files with the HTML extension (not HTM). If you have saved them as HTM, then make sure to rename them with the appropriate extension before proceeding. Also, make sure that your home page file is named "index.html."

Then move the mobile website you have created into the PhoneGap directory. After unzipping the PhoneGap files, open the PhoneGap directory and then open the iPhone directory (figure 6.2).

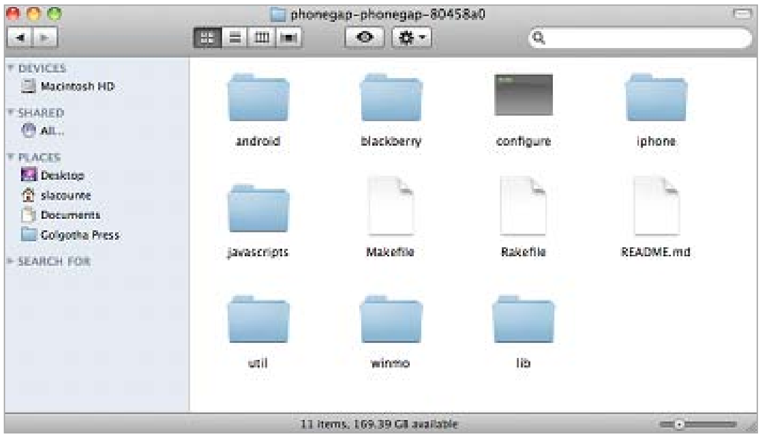


Figure 6.2

The iPhone app structure has already been created, so what you are doing at this point is simply applying your mobile web app to a prebuilt template. You can then drag and drop all of your mobile web app files into the WWW directory (figure 6.3).

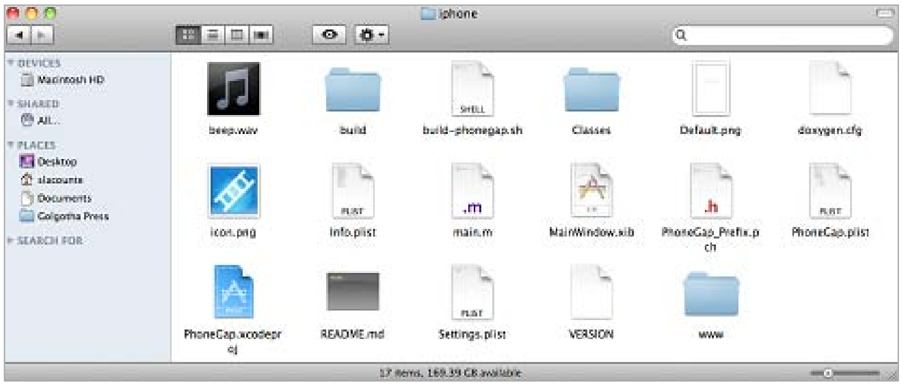


Figure 6.3

Your first iPhone app is now complete! Yes, it really is that quick! There are still some tweaks left to perform, but the description here should give you a good idea of just how easy it is to get started developing apps for the iPhone.

To see your app in action, go back to the PhoneGap and iPhone directory and open the file “PhoneGap.xcodeproj.” This will launch Xcode, the iPhone developer software (figure 6.4).

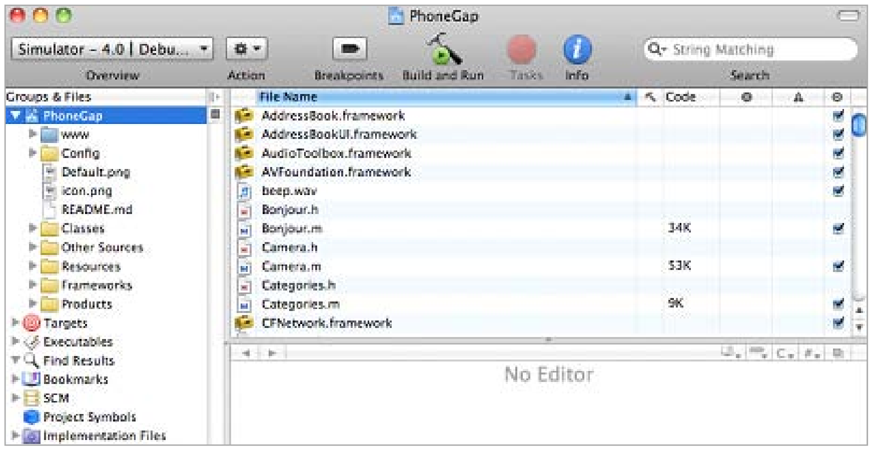


Figure 6.4

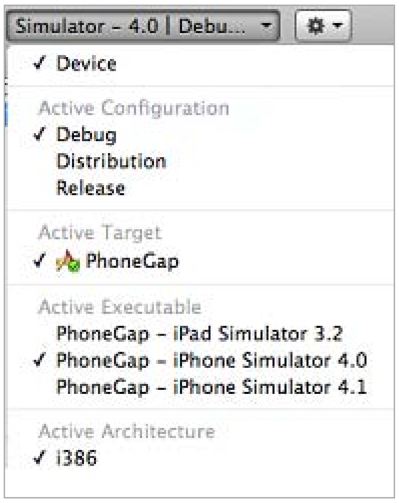


Figure 6.5

Next, click the drop-down menu in the upper-left corner of Xcode, make sure it is set to "Debug," and then select the simulator you want to run. It's best to select the most current simulator (iPhone 4.0), because that is the device that most people will have. It is also a good idea to ensure that the simulator runs on all platforms. For example, if you select iPad as your simulator, it will run the app as an iPhone app, not an iPad app, and so the app will appear significantly smaller than it would normally (figure 6.5).

You are now ready to build the app. Click the "Build and Run" button.

After about ten seconds the simulator will launch, and your app will appear. If it does not launch and you see red or yellow flags in the lower-right side of the screen, click on the flags to examine the message.

You will notice two things the first time the app loads. First, the icon is for PhoneGap; second, the loading image reads "PhoneGap." Let's see how to get rid of those two graphics and replace them with library images.

Close the Xcode program and simulator.

Now, let's make an iPhone app icon. It's simple. Create an image with a dimension of 57 × 57; use the file name "icon.png" to save the image. Move the file you created into the PhoneGap and iPhone directory. The file will replace the already-created "icon.png" and replace it with your new one.

Next, let's change the start-up image. Repeat the steps to create an icon, except this time name your image "Default.png" and make sure it has the following dimensions: 320 × 480. Move this image into the PhoneGap and iPhone directory.

Before launching the Xcode app again, rename "PhoneGap.xcodeproj" to your actual app name (i.e., "libraryapp.xcodeproj"); for our purposes, let's use "library .xcodeproj" as a file name. Once you rename it, open the file again, which will relaunch the Xcode software.

At this point, your icons and start-up screen will have changed, but the icon will still read "PhoneGap." To change the name that appears, click on the Config directory in the open window (figure 6.6).

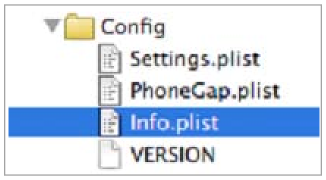
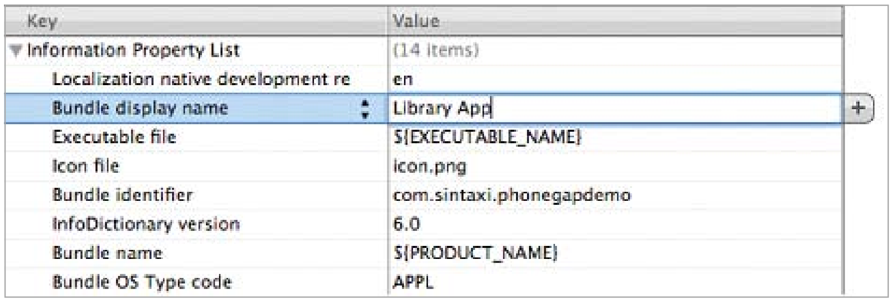


Figure 6.6

This will bring up a new window at the lower-right side of the screen (figure 6.7). Under "Bundle display name," click on PhoneGap and then type in the name that you want to appear. For our purposes, I used the name "Library App."



Key	Value
Information Property List	(14 items)
Localization native development re	en
Bundle display name	Library App
Executable file	${EXECUTABLE_NAME}
Icon file	icon.png
Bundle identifier	com.sintaxi.phonegapdemo
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL

Figure 6.7

Before loading your app in the simulator again, go to the top of the screen, click on Build, and then click on Clean All Targets (figure 6.8). It is advisable to do this before you run the simulator, just to make sure that you've saved and updated all changes.

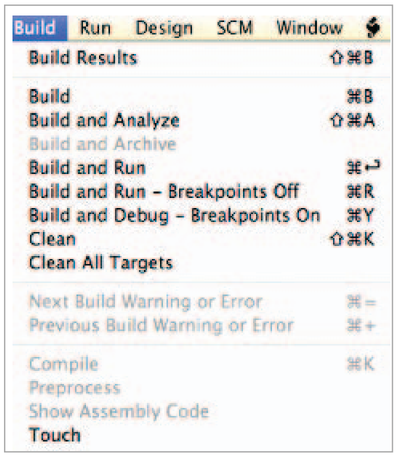

Build Run Design SCM Window
Build Results ⇧⌘B
Build ⌘B
Build and Analyze ⇧⌘A
Build and Archive
Build and Run ⌘↩
Build and Run – Breakpoints Off ⌘R
Build and Debug – Breakpoints On ⌘Y
Clean ⇧⌘K
Clean All Targets
Next Build Warning or Error ⌘=
Previous Build Warning or Error ⌘+
Compile ⌘K
Preprocess
Show Assembly Code
Touch

Figure 6.8

Finally, click on "Build and Run." Everything should now look as you want it to.

MOVING YOUR APP FROM THE SIMULATOR TO THE IPHONE

To move the app from your computer to your iPhone or iPod for testing purposes (not to put on the app store, which takes a little bit more work), you will need to obtain a provisioning profile for your app, for your device, and for yourself.

The first step is to go to the iOS Dev Center site (http://developer.apple.com/iphone/) to register to become a developer. Once you are logged in, you will see a menu bar on the right that reads "iPhone Developer Program" (figure 6.9).

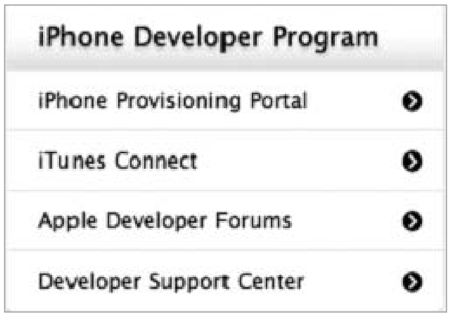
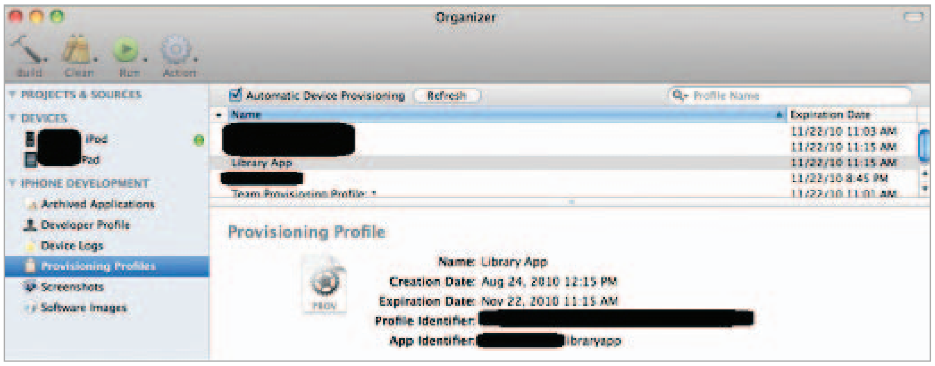


Figure 6.9

Next, click on iPhone Provisioning Profile. Then click on App IDs at the left-hand side; click on New App ID and fill in the information for your app.

Next, click on Provisioning at the left side of the screen and select New Profile. Once you've created the profile, download it to your computer. In the next few steps, we will match the profile to the key that you'll generate for the iPhone or iPod that you wish to test the app on.

After you have created your provisioning profile, you'll want to return to the Xcode Organizer (located in the Window menu) to pull the product identifier from your device (figure 6.10). Once you've plugged in the device and opened the organizer, click on Provisioning Profile and copy the app identifier. Next, go back into Xcode, click on Info .plist, and go to the box in the lower-right pane (figure 6.11). Change the bundle identifier to the one you have just copied, but leave in your user name.

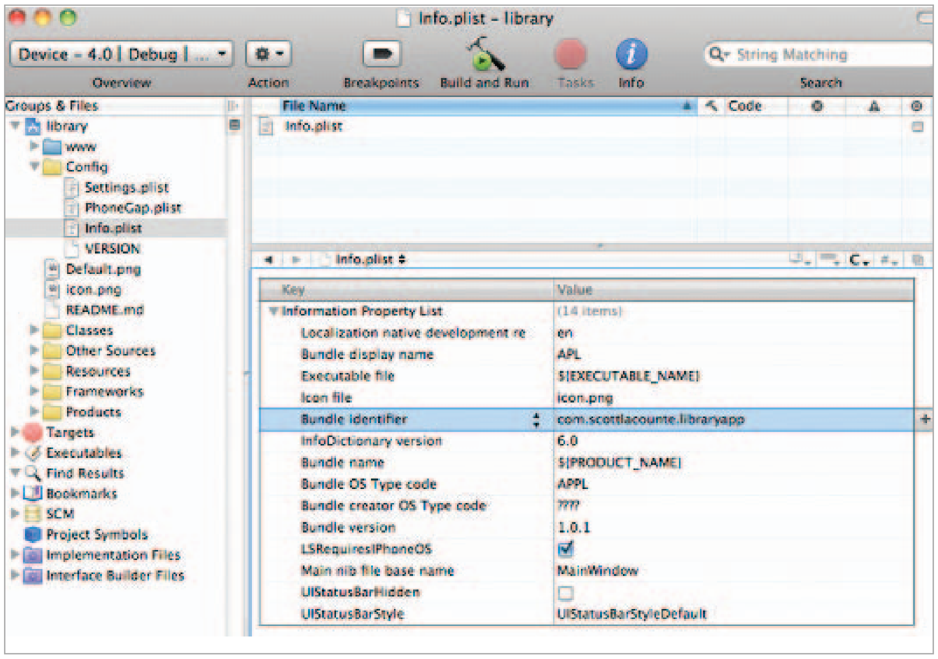


Name	Expiration Date
[redacted]	11/22/10 11:03 AM
	11/22/10 11:15 AM
Library App	11/22/10 11:15 AM
[redacted]	11/22/10 8:45 PM
Team Provisioning Profile: *	11/22/10 11:01 AM

Provisioning Profile

Name: Library App
Creation Date: Aug 24, 2010 12:15 PM
Expiration Date: Nov 22, 2010 11:15 AM
Profile Identifier: [redacted]
App Identifier: [redacted]libraryapp

Figure 6.10



Key	Value
Information Property List	(14 items)
Localization native development re	en
Bundle display name	APL
Executable file	${EXECUTABLE_NAME}
Icon file	icon.png
Bundle identifier	com.scottlacounte.libraryapp
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0.1
LSRequiresIPhoneOS	✓
Main nib file base name	MainWindow
UIStatusBarHidden	☐
UIStatusBarStyle	UIStatusBarStyleDefault

Figure 6.11

So your app ID should look something like this: IDNUMBER.com.scottlacounte.libraryapp (App ID.com.PROFILE NAME.APP NAME).

When you have installed all of the certificates and are finally read to go, the most important step is to make sure the Xcode is set to Device and not Simulator (figure 6.12). Until you switch this, the app will not load on the iPod or iPhone.

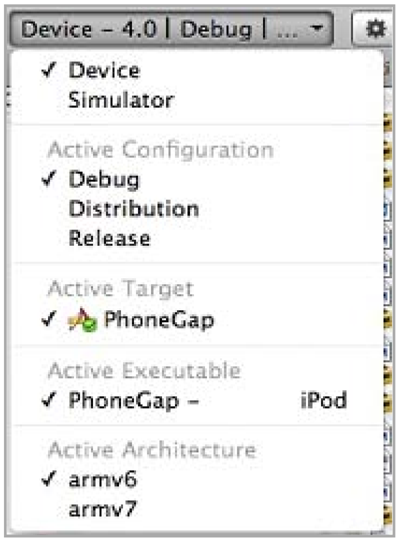


Figure 6.12

If you have done everything correctly, there will be a delay of approximately ten seconds before the iPod or iPhone turns on. If you missed a step, then in your device organizer there will be an orange circle next to the device (figure 6.13). When you click on the device, it will tell you what the error is. The most common error has to do with the provisioning profile not matching the device, which means that there is a certificate missing. Double-check the iOS Dev Center page to make sure you have downloaded all the certificates.

Once you are satisfied with your app and ready to sell it, then you will need to follow the instructions online to do so. You'll also need to pay the $99 registration fee for developers.

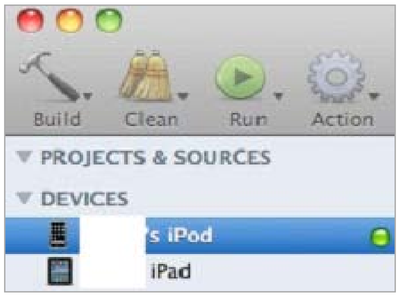


Figure 6.13

TESTING CORE USERS

Once you have an app ready to go, the final thing to consider is the best way to promote it. For example, people who will use the app are not necessarily the same people who visit the library regularly. How can you promote a service to a group of users whom you might not see on a regular basis?

One way to reach your user base is through a local online campaign. If the library does not have a Facebook page, then it should create one—fan pages regularly attract nonregular library users or people who want to know what's going on in their community but don't frequently visit the library.

Many businesspeople use smartphones—with this user base, you might collaborate with restaurants and coffeehouses near business areas of the city. Many businesses would be glad to leave out flyers advertising your app, especially if you include their name on the app in a sponsor category or work out some other arrangement. To get people into the library whom you might not usually see, you might put a coupon for a free video rental on the app—all they have to do is show the coupon on their phone and then the library would make a note that the patron has used the rental coupon.

To publicize your app, you can also look beyond the library at what businesses have done to get people on their Facebook pages or websites. Many local Chick-fil-A Facebook fan pages, for example, post code words each week on the fan page. If you go to the restaurant on a certain day and say the code word, then you get a reward. People visit the page fanatically to keep up to date just so they don't miss out on something. Thousands of other businesses have similar strategies.

Phones don't have a lot of storage or screen space. So you need to make your app appealing. If patrons have to choose between leaving the iPhone app for the Internet Movie Database on their phone and the library's app, will they really choose the library's? Why not? As you develop the app, continually ask yourself, "What will make people come back again and again?" For example, why would patrons bother downloading an app that tells them what time the library closes?

Library apps need to be interactive. If it was easy to communicate with librarians through an app, and if there are lots of phone-enabled research tools, then the app becomes a more essential tool. If it's not interactive, users will go with other apps every time.

Gale's AccessMyLibrary app is free for your patrons to use, and they probably don't even know it exists. This is a great app to make your patrons aware of through your own app. Through your own library's app, patrons can access Gale's research tools at any library within a ten-mile radius and without logging in.

If there is an incentive for patrons to download the app, then you will find users. If not, then you will have wasted a lot of resources. Mobile apps are a bit like animated graphics; when the Internet first became big, every amateur web designer showed off their talents with a flashy animated GIF. Those GIFs were definitely flashy, but after about three seconds, they became quite annoying. In a nutshell, iPhone apps work in the same way—for every excellent app, there are at least ten apps that fail. Unless your mission in developing an app is to do something cool and flashy and short lived, then think interactive.

7

BEYOND THE BASICS

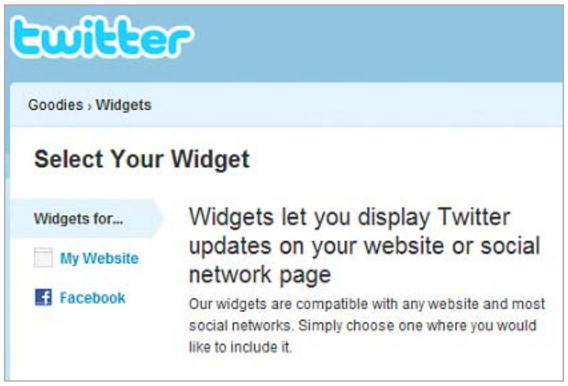
The point of this book is to make building an app as simple as possible. You can add JavaScript to your app to make it flashier, but talking in detail about JavaScript would take away from the point of this book—keeping it simple.

JavaScript is a pretty powerful language that can do a lot of things, but learning it isn't essential to building a great-looking app. If you know JavaScript, then use it; if you don't, then there's another option: widgets. If you blog or read blogs, then you likely know all about widgets, even if you don't know them by name. Widgets are the small little boxes that sit on the sidebars of blogs and offer links to things like Twitter, Facebook, and Google News streams. Widgets appear on the surface to be quite simple, but there are pages of code built into each one. The good news is that you don't have to know the code to build them.

A simple Web search for "widgets" turns up thousands of pages that offer customizable widgets; some are free, and others have small fees attached. You'll need to watch out for widgets that use Flash (which many phones do not currently support—most notably, the iPhone).

Many websites also offer widgets. Twitter is the perfect example. On the widget web page for Twitter (https://twitter.com/widgets/) you can build a custom widget in just a couple of short steps.

First, on the widget page, select My Website (figure 7.1).



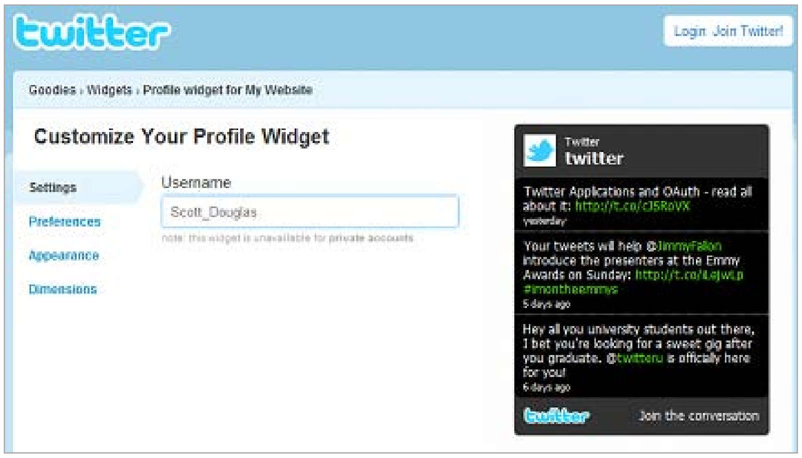
Widgets let you display Twitter updates on your website or social network page

Our widgets are compatible with any website and most social networks. Simply choose one where you would like to include it.

Figure 7.1

On the following page you will see several options; choose Profile Widget, which will display your library's most recent tweets.

Next, you can customize your widget (figure 7.2). Under the user name, you ideally would put the name of your library, but you can also put authors, magazines, or the name of anyone else with a public Twitter account. This would give your app content if you don't have time to do it yourself.



Customize Your Profile Widget

Username: Scott_Douglas

note: this widget is unavailable for private accounts

Twitter
twitter

Twitter Applications and OAuth - read all about it: http://t.co/cJ5RoVX
yesterday

Your tweets will help @JimmyFallon introduce the presenters at the Emmy Awards on Sunday: http://t.co/iLejwLp #imontheemmys
5 days ago

Hey all you university students out there, I bet you're looking for a sweet gig after you graduate. @twitteru is offically here for you!
6 days ago

Figure 7.2

Under Preferences, you can select how many tweets you display. Although the limit on Twitter is thirty tweets, it is advisable to consider the screen size of the mobile device you are developing for. You don't want users to have to scroll too much—ideally, you don't want them to scroll at all. Because of that, you might keep the number of tweets displayed to four.

Under Appearance, make sure the colors you select match the colors of your app. Under dimensions, you can adjust them to the dimensions of the phone or choose auto width if you want it to resize the widget yourself.

Finally, go to the bottom of the page and select Finish & Grab Code (figure 7.3). Now just copy the code and paste it into your app's code.

Figure 7.3

If you are thinking about providing a chat-based reference service (if you are not, you should be—by not doing so, you'll miss an entire generation of users), then consider LibraryH3lp (http://libraryh3lp.com). There is a small annual fee for the site's service (which varies depending on the size of the city you're in), but it is much cheaper than many similar services. Meebo and Google Talk offer free chat services, but their widgets use Flash, so they aren't supported on the iPhone app. In contrast, LibraryH3lp uses Java, so it's app friendly. Box 4 displays the source code of LibraryH3lp's widget.

In a mobile browser, the code from box 4 would look similar to figure 7.4.

Widgetbox (www.widgetbox.com) is one of the most popular widget builders; it's free but advertisement supported. A pro version (without the ads) is also available, starting at $29.99 a year. The site offers both Flash- and Java-based widgets.

Facebook also provides steps for embedding profiles into websites (http://developers.facebook.com/docs/guides/web/). Widgetbox, however, makes a much more user-friendly version, and if you are new to development, it is the better choice. Just make sure you copy the Java-based version of the widget and not the Flash-based one.

BOX 4 ■ Source Code for Library H3lp's Widget

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
 <meta http-equiv=content-type content="text/html; charset=UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0 user-
scalable=yes" />
 <title></title>
 <link type="text/css" rel="stylesheet" href="/css/mobile.css" />

 <style id="dynamic" type="text/css">
 </style>
 <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/
jquery.min.js"></script>
 <script type="text/javascript" src="http://d1aaqh87bn7fin.cloudfront.net/
mobile_2010011801.min.js"></script>
 <!—
 <script type="text/javascript" src="/js/lib/strophe.js"></script>
 <script type="text/javascript" src="/emoticons/emoticons.js"></script>
 <script type="text/javascript" src="/js/mobile.js"></script>
 —>
 </head>
 <body>
 <div id="header">
 <img id="presence" src="/presence/image/simpletext/unavailable" />
 <h1 id="title"></h1>
 </div>
 <div id="to" jid="my-queue@chat.libraryh3lp.com" lang="en"></div>
 <div id="recv"></div>
 <div id="send">
 <form action="#" method="post" onSubmit="return false;">
 <textarea id="msg">chat requires javascript...
</textarea>
 </form>
 <div id="buttons"><a href="#" onclick="libraryh3lp.sendLine(); return false;"><img
height="42" width="30" src="/images/mobilesend.png" /></a></div>
 </div>
 <div id="status"><a id="mail" href="#" onclick="libraryh3lp.mailTranscript(); return
false;">Email transcript
</a></div>
 </body>
</html>
```

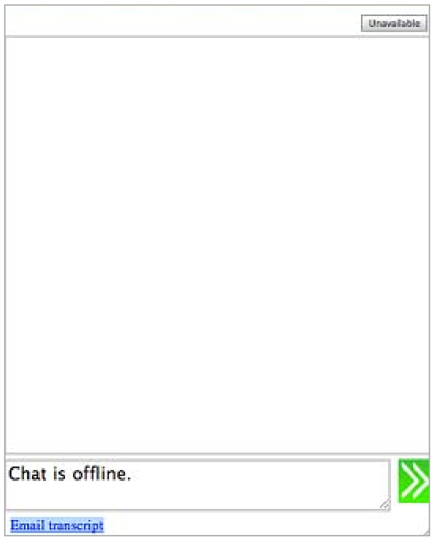


Figure 7.4

Widgetbox also makes it easy to develop a mobile app (www.widgetbox.com/mobile/). If you want a basic mobile web app (not a native app) that shows Twitter, Facebook, or Flickr updates, then Widgetbox might work well for the library; it does not, however, allow for users to edit the source code, and you might quickly find it to be too restrictive for your library.

Widgipedia (www.widgipedia.com) is another great resource for finding widgets; however, they are user submitted and customizable.

PhoneGap offers several codes that developers can copy for use in their own app; you can find these in the Community section of the website (www.phonegap.com/community/). It's frequently updated, so check back often. A useful code is for including a map, based on the phone's GPS (http://phonegap.pbworks.com/PhoneGap-Geolocation-Sample-Application).

I have tested all of the widgets and tools in this chapter to ensure that they will work with an iPhone app. As I have said before, however, JavaScript is tricky on mobile web apps and native apps, so always test before implementing. For every JavaScript that I have tested successfully on a native iPhone app, a dozen more come back with errors.

8
OTHER WAYS TO GO MOBILE

The point of this book is to show librarians how simple it can be to develop mobile applications. For the most part, mobile apps run on smartphones. However, librarians can also proactively investigate other mobile application programming interfaces (APIs).

Foursquare (for documentation information, see http://groups.google.com/group/foursquare-api/web/api-documentation/), which some bill as the next Twitter, is a location-based social media app that allows people to tell their friends exactly where they are and become "mayors" of locations they visit frequently. Gowalla (http://gowalla.com) is another social website like Foursquare that's growing in popularity; unlike Foursquare, though, Gowalla offers a little more in terms of customization.

When thinking about mobile apps, libraries should think of as many free technologies as possible for promotion—Foursquare and Yelp already have apps that you can easily link to. If your library has not staked its claim on Foursquare, why not? It's a perfect avenue for promotion—people can check into the library with the library app and get something for free—free videos, reduced fines, small prizes, and so on.

Some of the best social media sites that work on mobile phones are the same ones that are most popular on the Web: Facebook, Twitter, YouTube, and Flickr. If your library doesn't have a Facebook fan page, it's time. This is true too, though to a lesser extent, with the other three. What better way to showcase your library programming than with videos and photos? What's great about all the social media sites is that they are integrated together—you can feed Twitter updates into Facebook and share YouTube videos and Flickr slide shows instantly on Twitter and Facebook. The best thing about these sites, however, is that you don't have to do anything to go mobile. If you have a page on Twitter or Facebook, it's already mobile compatible—no extra steps are required.

Librarians don't give you answers—they merely point you in the right direction. Your app should work the same way. Save yourself time and trouble by finding code that's already finished. I hope you decide to develop a mobile app, but at the very least, I hope you make sure that your library is actively involved in the mobile technologies mentioned here.

WYSIWYG APP EDITORS

Mobile app development is a powerful library tool that every library should try out in some capacity, but not every library will have the time and/or resources to make it happen. For this reason, it's worth noting some alternate solutions. The disadvantage of using the alternates is their cost and, in most cases, the limited ability they offer to customize apps. Their advantages are that they will help you set up much quicker and nearly guarantee that your app will make it into an app store.

Some of the tools listed here will basically build an app for you; others provide an easy WYSIWYG interface. All are available for the iPhone; most are available for Android; some are available for BlackBerry and other mobile devices.

The up-front costs of these interfaces can be expensive, but the cost is next to nothing compared with what you would pay a programmer to take on the project.

Appanda

Website: www.appanda.com
Platform: Do-it-yourself (DIY) app solution for iPhone and Android
Summary: Appanda is best for libraries that want to feature only iPhone and/or Android versions of already-existing content (e.g., feeding in a blog or a video hosted on YouTube).
Pricing: $29.99 per month plus a onetime activation fee of $99

Swebapps

Website: www.swebapps.com
Platform: iPhone
Summary: The interface is a bit more visually appeasing than some do-it-yourself apps that feature a boxier look. Swebapps has all the same features as other DIY builders but also incorporates tools to create forms, lists, maps, and events pages.
Pricing: Basic plan starts at $29 a month, plus a $399 onetime fee

AppBreeder

Website: www.appbreeder.com
Platform: iPhone and web app
Summary: There are no up-front fees, but the site is limited in terms of customization and has a very generic look.
Pricing: $29–$49 per month for iPhone app or free web app

My App Builder

Website: www.myappbuilder.com
Platform: iPhone and Android
Summary: There are no up-front fees, but the website offers few screen shots and few details about features.
Pricing: $29 per month

Build an App

Website: www.buildanapp.com
Platform: BlackBerry, Android, Windows Mobile, Mobile Web, iPhone
Summary: Available on more devices than most app builders, and for a price that is cheaper than most, but the app has a very generic look and is limited in terms of customization.
Pricing: $14 a month and $19 setup fee for all apps, but iPhone app requires $199 setup fee

Mobile App Loader

Website: www.mobileapploader.com
Platform: iPhone, iPad, and Android
Summary: This is not the best-looking app builder, but it has low fees and is one of a few compatible with the iPad. Don't let the price fool you, however; extra buttons and features incur additional fees. There is also not a lot of room for customization.
Pricing: $4.99 per month and $59.99 setup fee

Game Salad

This DIY editor is for games. It isn't ideal for libraries, but it might be something to consider for teens. A program that lets teens build a game would be a great program. Best of all, it's free! Unfortunately, a Mac is required (http://gamesalad.com/landing/overview).

GOOGLE APP INVENTOR

Things were more complex for app developers just a few years ago—PhoneGap certainly has helped take away the complexity of mobile app development. In the past year, dozens of virtual companies have emerged promising to assist users who want to go mobile but lack programming skills. In this section I discuss Google's App Inventor (http://appinventor.googlelabs.com).

In my experience, developing apps for the iPhone is much simpler than developing for Android; Apple has spent much more money on app development, and it's obvious in its presentation and SDK functionality. Google is still playing catch-up. If you want to develop an app using PhoneGap, then refer to the appendix, which gives a list of where to go for tutorials on the Web. Google App Inventor, however, is entirely different from the traditional method of app development, so it's worth noting here what it is and how it works.

App Inventor was first made available on July 12, 2010; the tool claims to be so easy that elementary school children can use it with little effort. The interface uses drag-and-drop objects to create apps that run on any Android phone.

Because of App Inventor's simplicity, it is difficult to develop an app that is visually stunning. It is, however, an excellent place to test mobile development and decide what the best practices for the library will be. As Google continues to improve the drag-and-drop developer, it will most definitely get easier to use and more powerful.

At the time of this writing, Google App Inventor has only recently become public; Google is rapidly updating the program, and some instructions here will likely already be outdated.

Is it simple to use? App Inventor takes some getting used to, but it's much simpler than developing a native iPhone app. Is it cool? Not so much, but it has promise. I suspect that it will continue to get better—app development is something that Google has emphasized in recent months.

When Google first tested App Inventor, it used elementary-school-age kids to see if it was easy enough for them to use. It was. So can you develop an app quickly through the inventor? Yes. Should you? Probably not yet.

What follows is a very brief look at how the App Inventor works. It's still being updated, and it will likely look more polished in coming months when it becomes more public. As with any rapidly growing technology, what you see here will most likely not be what you see by the time this book is printed.

If you use Google Docs, then the App Inventor interface will look a little bit familiar. When you sign in, the first thing you will see is a list of all your current projects (if you have any) (figure 8.1). In the menu, you can either click on New to open a new project or on More Actions to download the code of previously built projects.

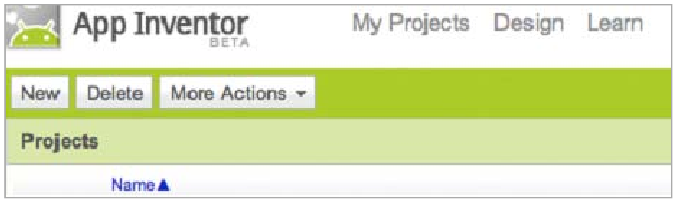
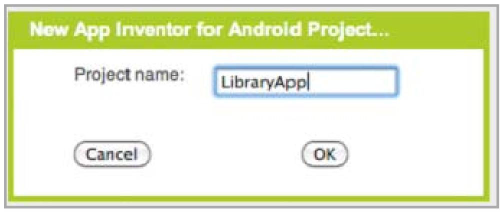


Figure 8.1

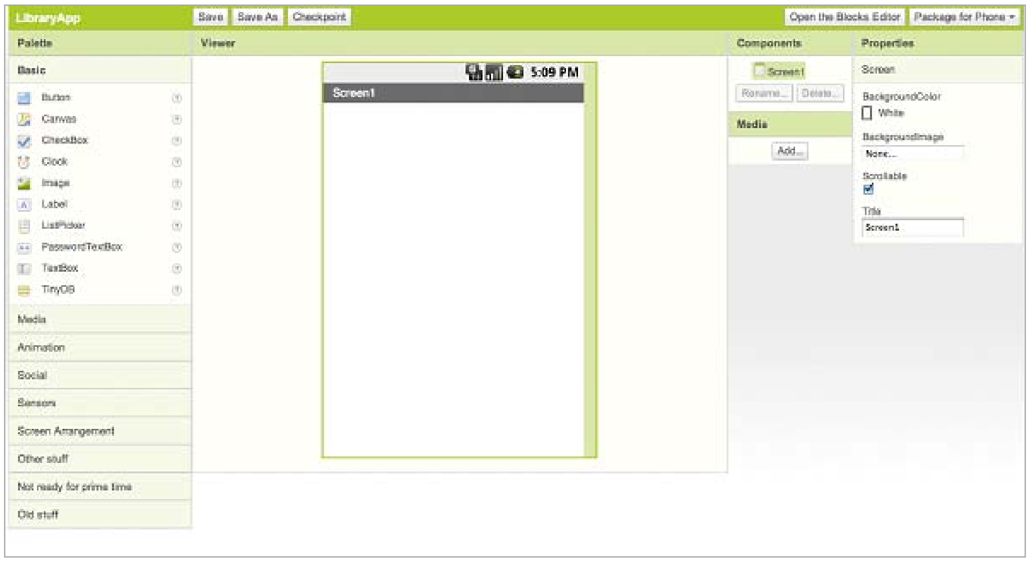
Next you need to name the project; make sure you do not include spaces in the name (figure 8.2).


New App Inventor for Android Project...

Project name: LibraryApp

Figure 8.2

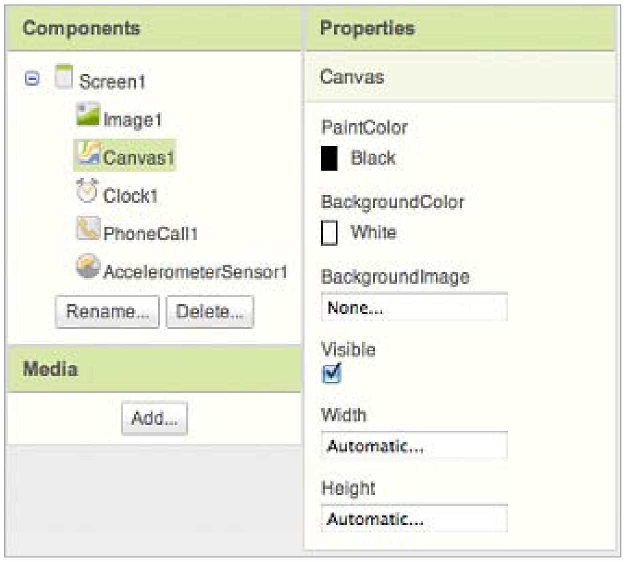
Then the project interface appears (figure 8.3). The first thing you will probably notice is how visual it is; this is because of its drag-and-drop interface.



Screen
BackgroundColor
White
BackgroundImage
None...
Scrollable
✓
Title
Screen1

Figure 8.3

If you want to include an image, then you can simply click on Image in the Palette menu on the left-hand side of the screen and drop the image into the viewer. From there, you can go into the Components menu on the right-hand side of the screen, click on Image, and then go into Properties to change the image or rename it (figure 8.4).



Canvas
PaintColor
■ Black
BackgroundColor
□ White
BackgroundImage
None...
Visible
✓
Width
Automatic...
Height
Automatic...

Figure 8.4

At any time, you can click on Learn at the top of the screen for full tutorials and videos; the project, like many at Google, is still in beta form. As I mentioned, from a commercial standpoint, it's not quite ready for primetime, but if you are considering developing a library app for Android, it's certainly worth checking it out.

Many people have speculated that this simple-to-use interface is the future of app development. One can only hope that it is, because it's certainly a step in the right direction.

INDEX

Patrons increasingly expect access to their libraries anywhere, anytime. This ALA Editions Special Report provides practical guidance for how librarians can put the library in the palms of their patrons' hands. Using the HTML skills that many librarians already have, combined with flexible development tools, technology expert Scott La Counte shows how creating a customized mobile app doesn't need to be expensive or require deep expertise. In straightforward, practical terms he

- Demonstrates how to establish a presence on the mobile web with mobile websites and phone apps
- Details open-source development tools such as PhoneGap that enable the creation of mobile apps that work on a variety of mobile operating systems
- Discusses methods for assessing a library's user base and for getting buy-in from administrators

Following the pointers in this ALA Editions Special Report, libraries can easily go wherever their patrons do!

ISBN 978-0-8389-1129-7

Scan this QR code to go to the ALA Online Store from your smartphone (app required).

ala editions

AMERICAN LIBRARY ASSOCIATION / alastore.ala.org
50 E. Huron Street, Chicago, IL 60611
1 (866) SHOPALA (866) 746-7252