PhoneGap Essentials

Starting with installing PhoneGap, you’ll develop an app that uses various device capabilities through different plugins and learn how to build an app in the cloud with PhoneGap’s Build service.

You’ll discover how to use PhoneGap to create an application view, along with how to use a camera, geolocation, and other device capabilities to create engaging apps. Next, you’ll augment applications with PhoneGap’s plugins, using minimalistic code. You’ll explore the app preparation process to deploy your app to the app store.

By the end of this book, you’ll have also learned about different UI frameworks that work for mobile devices and how to test different screen sizes.

Who this book is written for

If you are a mobile application developer in iOS or Android, or a web application developer who wants to learn how to make cross-platform mobile applications using PhoneGap, this book is perfect for you. To make the most of this book, it will be helpful if you have prior knowledge of HTML5, CSS, and JavaScript.

What you will learn from this book

- Get to grips with the fundamentals of PhoneGap to get started
- Set up a development environment for Linux, Mac OS, and Windows
- Use Cordova CLI, workflows, and the Plugin Manager to create mobile applications efficiently
- Understand the development workflow to create native cross-platform mobile applications
- Embed plugin support to transition to PhoneGap or use it to enhance existing applications
- Improve your mobile development knowledge using object-oriented programming (OOP), reusable components, and AJAX closures
- Be empowered to build your own mobile apps quickly with ease
- Discover tips and tricks to make app development fun and easy

By the end of this book, you’ll have also learned about different UI frameworks that work for mobile devices and how to test different screen sizes.
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Get up and Running with PhoneGap'
- A synopsis of the book’s content
- More information on PhoneGap Essentials
About the Author

Ivan Turkovic is an experienced developer with over 15 years of programming experience. He developed an interest for software development at an early age of 8, when he started to write Basic code on his first Commodore 64 computer. Seeing the potential of what could be done with it encouraged him to pursue his studies in computer science. He has extensive experience of mobile and web application development and has delivered many enterprise and consumer solutions. His favorite languages are Ruby, Swift, and JavaScript. Ivan has always been interested in emerging technologies. In his spare time, when he is not working on another exciting pet project, he is traveling and learning new "human" languages. He writes his thoughts at http://ivanturkovic.com.

In the past, he has helped found several technology start-ups. As a founder and digital consultant at Blaeg (http://www.blaeg.com), he helps companies, start-ups, and individuals integrate technology into their business and strategies. He also conducts trainings and workshops and writes blogs to share his knowledge with the community.
When you learn a new programming language, the traditional first step is to go through a *Hello, World* tutorial that teaches you everything you need to know to get a basic program working, such as one that prints the text *Hello, World* on the screen. In this sense, this book is the *Hello, World* tutorial to build your first PhoneGap application.

This book provides you with a great way to get started with PhoneGap and also gives detailed examples from the different areas that PhoneGap covers. It can be a great reference book for later use, in case you want to quickly refresh the essentials with PhoneGap. This book is updated with the latest releases so it supports the npm plugins too. It will take you through the process of installing everything that you need to get started with PhoneGap. It will cover the basics of PhoneGap, such as the command-line interface (CLI), PhoneGap plugins, and APIs provided by PhoneGap.

You'll find this book to be a great introduction to PhoneGap. It will give you a chance to learn some more about the Framework and give you the confidence to build something great that works across different devices.

Enjoy the book!
What this book covers

Chapter 1, Get up and Running with PhoneGap, shows you what PhoneGap is used for, a basic understanding of what you can achieve with the framework, and why it is good to use.

Chapter 2, Mobile Platform Support, will go into more detail about the various platforms that are supported in PhoneGap and what their limitations are.

Chapter 3, Command-line Interface, shows you that the Cordova command line is crucial for the effective and streamlined development of PhoneGap applications. It will give you an overview of the capabilities and how to use them.

Chapter 4, Plugin Support, will show you that accessing any platform API is disabled by default, so all features are nicely packed in separate plugins that you can add and manage based on your needs.

Chapter 5, First PhoneGap Application, shows you how to create the first basic Hello World application with PhoneGap and tells you about the file structure organization and building a fully functional application.

Chapter 6, Accessing Native APIs, discusses the different hardware APIs that are available on most of the platforms PhoneGap supports.

Chapter 7, Accessing Media Content, covers additional native features that are available through JavaScript. This chapter will cover accessing the native camera capabilities and other media content on the device.

Chapter 8, Application Development Workflow, will cover the basics on how to work with the PhoneGap application development process and how to go through the phases until the application is built.

Online Chapter, Beyond PhoneGap – Ionic, goes a little bit forward by teaching you how to apply hybrid mobile UI that will work across different platforms and different screen sizes. Ionic framework seems like a great option to apply it on the top of the PhoneGap. This chapter can be found online at https://www.packtpub.com/sites/default/files/downloads/Beyond_PhoneGap_Ionic.pdf.
PhoneGap has brought a new paradigm to the field of mobile application development since its inception. It has tried to replace the old, fragmented mobile platform development, where each mobile device is developed using different programming languages and software patterns, with simple JavaScript and HTML.

Each new major version has managed to improve the performance and support features that were available only to native devices before. PhoneGap has come so far that most of the basic needs for a mobile application can be achieved with it, without writing a single line of code in the native language.

In this chapter, we will cover the following sections:

- A brief history
- What is PhoneGap?
- Setting up a local development environment

Let's get started with a quick introduction to PhoneGap.

**A brief history**

In 2007, Apple introduced its first smartphone: the iPhone. It changed the mobile phone industry forever. iPhone was the first smartphone that provided a browsing experience comparable to desktop web browsing. Many web pages were trying to mimic iPhone's look and feel for mobile use. Originally, iPhone didn't support third-party native apps. Many tried to create hybrid applications by hosting them on web servers. The application was running inside the Safari browser.
iPhone's immense success was noticed by competitors, especially Google. Google had planned to introduce Android before iPhone. Android back then was like Blackberry OS and interacted through the keyboard, but, seeing the success of iPhone, they decided to ditch the keyboard and open source it. Android had the ability to develop native applications. Apple allowed the development of native applications with the next version of iPhone. Competing platforms have different development stacks. It requires an extra amount of work to make them work on many popular platforms. This makes even the simplest application development across multiple platforms difficult. After some time, most platforms offered the ability to communicate between inline web browsers and the application's native code. With this, compiled hybrid applications became a reality. You could create the whole application with JavaScript, HTML, and CSS, and access native libraries through native code. It wasn't the best solution, since you needed to write native code that supports it, but that was going to change with the arrival of PhoneGap.

PhoneGap was started as a project at the iPhoneDevCamp event in 2008. It was started by a team of developers wanting to simplify cross-platform mobile development. Until then, it was easy to create applications for a single platform but there was no tool to manage it for multiple platforms. In the beginning, the idea was to create project templates that could be reused inside Xcode for iOS or Eclipse for Android to develop hybrid applications. Shortly afterwards, PhoneGap supported Blackberry OS. The team behind the project was from Nitobi Software company. They started to work on PhoneGap as a more serious tool for development. It began to be used for a full development cycle including preparing builds for deployment to the app stores or for using inside enterprises. PhoneGap won the People’s Choice Award at O'Reilly Media's 2009 Web 2.0 Conference. Prior to Apple's developer license agreement version 4.0, the rules for submitting PhoneGap applications to the Apple app store were not clear and many apps were rejected for that reason. After updating the developer license agreement, Apple has confirmed that the framework has been approved for submitting PhoneGap applications.

In 2011, there were two important items of news for PhoneGap development. Adobe decided to acquire Nitobi Software as a part of the strategy for moving away from Adobe Flash on mobile devices. The other news was that they were going to open source PhoneGap and contribute it to the Apache Foundation. Since being contributed to the Apache Foundation, it changed to Apache Callback, Apache DeviceReady, and finally Apache Cordova (http://cordova.apache.org). After Adobe acquired it, the team behind PhoneGap worked full time on the project and the updates are being delivered on a monthly basis.
The PhoneGap brand has been preserved by Adobe. PhoneGap is now a fork of Apache Cordova with some extra features. PhoneGap and Cordova were basically identical until the release of version 2.x. After that, the development went into simplifying project creation. A command-line interface was created for most common actions, and the core features were separated into numerous plugins. The new process simplified the creation and installation of new plugins.

**What is PhoneGap?**

In short, PhoneGap ([http://www.phonegap.com/](http://www.phonegap.com/)) is a distribution of Cordova. Cordova is an open source mobile framework that gives an alternative to native development and the existing application is reusable on other platforms with little or no modification to the code.

To be more precise, PhoneGap is an application framework that is capable of developing, and later building, native applications that can be deployed across different mobile platforms, thus simplifying the process and saving a lot of time. PhoneGap gives you the ability to use a single programming language—JavaScript together with HTML and CSS—to build the user interface.

With PhoneGap, any web developer can start developing mobile applications with no need to learn additional skills, apart from learning about PhoneGap's command-line interface (CLI) commands and PhoneGap's API. In a matter of hours, you can create a working prototype that can be tested directly on the user's smartphone, or built and deployed to the app store as a normal native application in all major application stores (iTunes app store, Android Apps on Google Play, Amazon Appstore, and Windows Store).

PhoneGap can be used to target support for multiple platforms from day one without needing to have another developer for another technology stack. Currently it supports the following major platforms:

- **Microsoft Windows Phone** ([http://msdn.microsoft.com](http://msdn.microsoft.com))
- **Blackberry 10** ([https://developer.blackberry.com/](https://developer.blackberry.com/))
- **Tizen** ([https://developer.tizen.org](https://developer.tizen.org))
PhoneGap's basic philosophy is to use the feature of the operating system that enables communication with the native code directly from the JavaScript inside the device's web browser, which is also being used as a UI rendering engine. There is no defined standard for UI rendering, so it can be developed to look like a regular page since we are defining the whole UI interface with HTML and CSS; or we can use available solutions that were specially built for the size of the screen on mobile devices.

**Where can it be used?**

As PhoneGap is already a stable framework and is constantly evolving, there are a lot of features that were previously available only for native applications but that can be used now inside PhoneGap. Let's list some of the major features that are being supported by the latest PhoneGap version:

- Camera
- Capture photos
- Compass
- Connection status
- Contact list
- Device details
- Events sent from the native environment
- File management
- Geolocation
- Multi-language support and localization
- In AppBrowser, the ability to run another browser view inside the PhoneGap application
- Media
- Storage

Most of these features are available through calling native methods, but as HTML5 is progressing, many of these features are available through native web view HTML5 method calls, thus removing dependency on the native code. If you want to see what features are supported with HTML5, you can take a look at this page (http://html5please.com/). It shows support for the various mobile browsers.
What PhoneGap is not

It is not a solution that fits all needs. It depends on the features and utility of the application, which can determine if PhoneGap is fit for its needs. If you are looking to develop a graphic-intensive 3D game or application that needs to use all the processing power the device has, then probably it is not the best choice to start with. PhoneGap can easily be used for all applications that do not involve a lot of animations or complicated transitions that are not easily achieved inside the HTML and CSS environment.

There is another misconception: many developers think that PhoneGap compiles JavaScript code into native binary code, which is not true; it only packages the JavaScript, HTML, and CSS into a wrapper that runs the content inside a sandboxed web page.

Competition

There are a lot of alternatives with differing perspectives, ranging from using the web page shortcut on the home screen as an icon, to emulating the application feel, running a regular web page like an application or even building the JavaScript source code to a real native application.

Here we will take a look at four different solutions that all have advantages and disadvantages:

- **jQuery Mobile** ([http://jquerymobile.com/](http://jquerymobile.com/))

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popular as a mobile page</td>
<td>Could be too heavy for some smartphones</td>
</tr>
<tr>
<td>A lot of plugin support</td>
<td>No official paid support</td>
</tr>
<tr>
<td>Based on jQuery</td>
<td></td>
</tr>
<tr>
<td>Great tools, such as Theme Roller and Codiqa UI builder</td>
<td></td>
</tr>
<tr>
<td>It is used only as the visual part of the application; it can be used with multiple JavaScript frameworks</td>
<td></td>
</tr>
</tbody>
</table>
Get up and Running with PhoneGap


<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Based on ExtJS</td>
<td>Sometimes slow (client-side DOM generation)</td>
</tr>
<tr>
<td>Full MVC framework</td>
<td>No native controls</td>
</tr>
<tr>
<td>Official paid support</td>
<td>Few customization options</td>
</tr>
<tr>
<td>Sencha Architect</td>
<td>Performance</td>
</tr>
</tbody>
</table>

- **Titanium** ([http://www.appcelerator.com/titanium/](http://www.appcelerator.com/titanium/))

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapid prototyping</td>
<td>Increased complexity and costs</td>
</tr>
<tr>
<td>Native UI</td>
<td>Flexibility limitations</td>
</tr>
<tr>
<td>Web oriented</td>
<td>Native UI can be limiting sometimes</td>
</tr>
<tr>
<td>Cross-platform support</td>
<td></td>
</tr>
</tbody>
</table>

- **PhoneGap** ([http://www.phonegap.com](http://www.phonegap.com))

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single code base for all platforms</td>
<td>PhoneGap can be a complex framework if not understood well, thus making it less responsive</td>
</tr>
<tr>
<td>Rapid deployment</td>
<td>No default UI</td>
</tr>
<tr>
<td>Access to native functions</td>
<td>Limited access to more advanced features without additional plugins</td>
</tr>
<tr>
<td>Offline usage</td>
<td>Complex business logic</td>
</tr>
</tbody>
</table>

**PhoneGap or Cordova?**

It seems that there is a lot of confusion between these two because they share the same history and their paths diverged only recently. From the beginning the project was called PhoneGap, but once Adobe bought the company behind it, it decided to open source the majority of the code while keeping the name for its use and giving a new name (Cordova) to the open source project.

Since then, PhoneGap has been built on top of the Cordova project with some additional libraries and tighter integration with other PhoneGap tools and services; the most prominent for PhoneGap developers is PhoneGap Build, which we will go into in greater detail in the next chapter.
There is not much difference for the beginner between PhoneGap and Cordova, apart from the ability to build native applications for all major platforms from any operating system with the help of PhoneGap Build. This makes it more useful in some cases since the majority of people do not like switching from their favorite operating system.

Setting up a local development environment

Since this book covers all major operating system platforms, before we start to learn more about PhoneGap and its useful features we need to prepare the development environment. The PhoneGap website (http://phonegap.com/install/) has instructions on how to install it, which should be fairly easy to achieve. However, in reality, each platform requires us to install additional libraries and simulators to be able to run them for different mobile platforms.

There are two ways to develop PhoneGap applications. Until PhoneGap introduced the command-line interface (CLI), the only way was to develop plugins for the integrated development environment (IDE), such as Xcode or Eclipse. With the introduction of the CLI, it was possible to develop PhoneGap mobile applications with PhoneGap SDK inside your favorite text editor or integrated developer environment. In this book, we are going to use the latter option; since most developers have different opinions about editors, it is better to stay neutral on this topic.

However, for this book I will be using Sublime Text 3 (http://www.sublimetext.com/3), a plain text editor that is one of the most popular choices nowadays and has numerous plugins and extensions to work with. Since it supports Windows, Mac OS, and Linux, it won't be included in the installation process for a specific operating system. On the provided link, you can download the text editor and install it with the provided information. Sublime Text 3 can be used for free for as long as you like (you will be prompted to buy it from time to time); this is a fully working editor without any limits. But I would strongly advise you to get the full version and support the people behind it since they have done a great job and it only costs $70 and doesn't have restrictions on the number of machines for a single user.

There is a great free alternative to Sublime Text called Atom editor, which is open source and available at the following page: https://atom.io/. It will be sufficient to do the work but it doesn't come with lot of plugins like Sublime Text.

In the following pages, there will be instructions on how to cover the installation process for the three most popular operating systems that developers are using: Windows, Mac OS, and Ubuntu. For other Linux distributions, it should not be too hard to adapt the Ubuntu installation process, especially if you are using Debian. Let us get started.
Mac OS

In the beginning, PhoneGap started as a way to create iOS applications with JavaScript and HTML to create a fully functional iOS application without writing a single line of code in Objective-C. To get started for Mac OS, there are few more steps than the instructions on the PhoneGap website (http://phonegap.com/install/) are telling us to perform. We will need to install simulators for supported mobile devices. On this platform, we will install support for iOS and Android development. For installation purposes, it is assumed that this is being installed as a clean installation so, if you already have some of the libraries installed, it is advisable to update them to the latest version to make sure they are working properly. For Mac OS, the latest operating system version (Yosemite) will be used.

The easiest way to install the various Unix tools and open source software onto Mac OS X is via a package manager but unfortunately OS X does not come with one; however, there are some alternatives that we can use instead. We are going to use Homebrew (http://brew.sh/).

If you already have your own package manager preference or you want to compile it from the source code, you are encouraged to use the latest version for the library.

Before installing Homebrew, we need to install Xcode (https://itunes.apple.com/au/app/xcode/id497799835?mt=12) so we can build and run applications in the iOS simulator:
To install the Xcode component, perform the following steps:

1. Download the image and then drag the Xcode icon into the Applications folder. Before proceeding with the next step, you need to open Xcode and agree to the license, as it needs to install additional components:

![Installing components](image)

2. After the components are installed, you need to install the Xcode command line tools with the following command:
   
   ```shell
   xcode-select --install
   ```

3. This will open the following prompt to install the required tools. Select **Install** to install the required libraries, and wait until everything is successfully installed:

   ![Install prompt](image)
4. To download and install Homebrew, run the following installation script on the command line. It will ask you for a few details before proceeding, but you can leave the default values in use, unless you have a specific reason to change them, then wait for the script to finish:

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

5. After successful installation of Homebrew, please run the following command to check if the installation was successful:

```
brew doctor
```

6. Before installing any additional libraries through Homebrew, let us run `update` to get a list of the latest versions of the libraries:

```
brew update
```

7. Since PhoneGap is a Node.js application, we will need to install a working version of Node.js. With Homebrew, the process of installing additional libraries is easy; just run the following command:

```
brew install node
```

8. To be able to build and run applications in the Android simulator, you need to install the full Android software development kit (SDK) by running the following command:

```
brew install android-sdk
```

9. Add the Android SDK path to be used with PhoneGap with the following commands:

```
vi ~/.bash_profile
```

10. At the end of the file, add the following line:

```
export ANDROID_HOME=`brew --prefix android`
```

11. Save and close the file and run the following to reload the newest updates:

```
source ~/.bash_profile
```
12. The following command will launch the Android SDK manager window and you should see the recommended libraries already preselected:

```
android sdk
```
13. Click the **Install 2 packages** button and accept all the agreements for the required libraries. Then press the **Install** button:

14. When it's finished, quit the Android SDK Manager application.

15. Besides Android SDK, we need to install Apache Ant in order to be able to build and run the applications inside Android Simulator. Run the following command:

```
brew install ant
```

16. In order to emulate an Android device, we need to first create a virtual device. Run the following command to set up the default Android virtual device:

```
android avd
```
17. This will launch the Android Virtual Device Manager. Click the Device Definitions tab and select a sample device. From the list select the Nexus 5 profile, and press the Create AVD button as shown in the following screenshot:
18. Fill out the device profile and click OK:

![Android Virtual Device Manager](image)

19. Close the Android Virtual Device Manager window.

20. Next we need to install the required libraries (to be able to build and run applications in the iOS simulator) by installing the Node.js library to control the iOS simulator directly from the command line:

```
sudo npm install -g ios-sim
```
21. When all the required libraries and tools are installed, install PhoneGap with the following command:

```
sudo npm install -g PhoneGap
```

22. To verify we can build and launch applications in both simulators, let us create and run a quick sample application by running the following command:

```
PhoneGap create hello com.example.hello HelloWorld
cd hello
```

23. This creates a basic sample application. Next we need to verify it works on the Android and iOS simulator.

24. For iOS we need to open the iOS simulator and run the application. Run the following command:

```
phonegap run ios --emulator
```
25. To test if everything is working well on Android, we are going to run the following command, which should run Android Device Simulator as we selected before:

```
phonegap run android --emulator
```
Linux

Linux is well known for having multiple distributions that have different ways of installing new libraries on it. For this book, we are going to use one of the most popular distributions: Ubuntu. Ubuntu is built on top of Debian so installing on both of them shouldn't be much different. As for the other distributions, it shouldn't be too hard to adapt the following commands for these environments. Perform the following steps:

1. First we need to fully update Ubuntu to avoid any issues while installing the required libraries:
   ```
   sudo apt-get update
   sudo apt-get install build-essential
   ```

2. Next we install some libraries that are required to finish the installation:
   ```
   sudo apt-get install python-software-properties python g++ make
   ```

3. Since PhoneGap is a Node.js application, we will need to install the latest working version of Node.js. Make sure you install Node.js and not node, which is a different library:
   ```
   sudo apt-get install nodejs
   ```

4. While that's all you need to do to install Node.js, there is a small detail that you need to take care of. When Ubuntu installs the package, it names the Node.js executable `nodejs`. The problem is that many applications, including PhoneGap, expect the executable to be named node. To fix this inconsistency, simply create a symlink named node that points to Node.js as follows:
   ```
   sudo ln -s /usr/bin/nodejs /usr/bin/node
   ```

5. Next we need to install additional libraries for Node.js:
   ```
   sudo apt-get install npm
   ```

6. Finally, we can install the PhoneGap library by running the following command:
   ```
   sudo npm install -g phonegap
   ```

7. Now we need to install the Ant tool, required to build and run Android applications:
   ```
   sudo apt-get install ant
   ```
8. Since Android uses the Java language, we will need to install Java JRE and Java JDK, which supports it:
   ```
sudo apt-get install openjdk-7-jre
sudo apt-get install openjdk-7-jdk
   ```
9. Now it is time to install the actual Android SDK. Run the following command:
   ```
wget http://dl.google.com/android/android-sdk_r24.0.2-linux.tgz
tar zxvf android-sdk_r24.0.2-linux.tgz
   ```
10. Let us move the content of the Android SDKs to a more appropriate location:
   ```
sudo mv android-sdk-linux/ /opt/android/
   ```
11. Now we need to set up paths to be executable from anywhere inside the terminal. Let us open the `vi` editor:
   ```
vi ~/.bashrc
   ```
12. Enter the following lines at the end of the file:
   ```
export ANDROID_HOME="/usr/local/android-sdk-linux/tools"
export ANDROID_PLATFORM_TOOLS="/usr/local/android-sdk-linux/platform-tools"
export PATH="$PATH:$ANDROID_HOME:$ANDROID_PLATFORM_TOOLS"
   ```
13. To use the command in the same terminal, you need to run the following command:
   ```
source ~/.bashrc
   ```
14. Now that we have set up the Android SDK environment, we need to run the configuration application to download the required libraries and emulators. Run the following command:
   ```
android
   ```
15. Execute Android from the shell. This is a good first test to make sure you've done everything right up to now. If you get an error, rerun through the preceding steps and make sure that you've installed all the requirements and added your environment variables correctly.
If there is no error, you should see that executing Android opens up a kind of package manager that you can use to install different Android components. By default, a certain number of these will be pre-selected for installation. Leave them as is and, in addition, tick the box next to the entry named **API 4.4.2 (API 19)**. Once you’ve done that, download and install everything by accepting the license, as shown in the following screenshot:
16. Next we need to confirm installation of the requested packages, by clicking on Install:

![Choose Packages to Install](image)

17. The last step before trying it out is to create an Android emulator that will emulate a working Android device on your computer. By default no emulator is set, so we need to set it manually by running the following command:

   `android avd`

18. After opening the window, we should see an empty list. Switch to the Device Definitions tab and press Create AVD:
19. For our emulator, we will use the following settings:
20. After we have installed all the required libraries and the emulator required to run the Android app, we can finally create our first test application by running the following command:

```bash
phonegap create hello com.example.hello HelloWorld
```

```bash
cd hello
```

21. To verify everything is working fine, you need to run the emulator:

```bash
phonegap run android --emulator
```

**Windows**

If you want to develop and test applications for Windows phones, you will need to have a working Windows computer to hand because you can only run them there.

There are multiple versions of the Windows operating system but it should be fairly easy to use any of them by carrying out the following instructions, since most of the libraries and applications are available as an installer. Thus, you can simply go ahead with the installation process and the only thing you need to do manually is to set the environment variables to be reachable from any command prompt.

The easiest way is to create a Development directory in your home directory (C:\Users\yourusername\Development). For ease, you might want to drag that folder into the Favorites list in Explorer, too. We will add all the source code we work on to the examples there.

Let us start with the stable Java SE Development Kit. Perform the following steps:

2. Run the downloaded file and go through the following steps:

3. You need to make sure that you have the following paths set in your Advanced System Properties Environment Variables.
4. You need to set the path to be visible so it can be run as a command from the command prompt. Navigate to Control Panel | System and Security | System. There you need to click on the Advanced tab, as shown in the following screenshot:

You will need to press the Environment Variables button and update system variables with the following data:

1. Create a new system variable:
   JAVA_HOME with value: C:\Program Files\Java\jdk1.7.0_75

2. Next we need to edit an existing variable:
   Path and add the following value ;%JAVA_HOME%\bin
3. Add the preceding line at the end of the remaining paths and make sure it is preceded by a ; (semicolon). It is important not to forget it:

4. You can test it by running the following command:
   ```
   java
   ```

5. In the command prompt, if you see the regular Java help text printout response then the Java environment is all set. Close the command prompt.

6. Next we need to download Android SDK for Windows, which is available as an automated installer. Download the latest SDK version from:
   ```
   ```

7. Make sure you do not use the default proposed folder since you might have some problems with accessing folder content in the following steps. Instead, set the installation location to `C:\Android\android-sdk`. 
8. We now need to go back to add an additional environment variable in the Android SDK location as we did before for Java SDK.

9. First we add a new system variable:

   ANDROID_HOME with value C:\Android\android-sdk

10. Next we need to edit the existing variable:

    Path and append additional value ;%ANDROID_HOME%\platform-tools;%ANDROID_HOME%\tools

11. Again, do not forget to add a ; (semicolon) at the beginning.

12. Next we need to open another command prompt to check whether it is installed properly. Run the following command:

    android

13. This will open the Android SDK Manager, where we will need to select which Android version to download; this is going to be used inside the emulator:
14. After selecting the packages ticked in the screenshot, you need to click **Install 7 packages** and wait until they are installed.

We need to install the git command (used for many libraries) and later plugins that we will require to make it work with PhoneGap. Perform the following steps:


2. You should open the installer, follow the instructions, and select the predefined values. After installation, repeat the same process with system variables and add the variable `GIT_HOME` with a value `C:\Program Files\git`. Add additional information to the existing `PATH` by appending: `;%GIT_HOME%\bin`.

3. Again, to test whether it is working, run the `git` command in the newly opened command prompt.

4. Ant is a tool that is required by Android to build all the code into the Android application package. You need to download and install Ant from: [http://www.us.apache.org/dist/ant/binaries/](http://www.us.apache.org/dist/ant/binaries/). Select the latest version that has the .zip extension.

5. Unzip the content into the folder `C:\Android\apache-ant-1.9.4`.

6. Again set the system variables with the creation of a new variable `ANT_HOME` with the value `C:\Android\apache-ant-1.9.4`. Update the existing `PATH` variable by appending `;%ANT_HOME%\bin`.

7. In the next step, we will install the latest version of Node.js from the [http://nodejs.org](http://nodejs.org) as an installer. After it is downloaded, run the downloaded file, which will guide you through the process. The Installer should set all the path variables you need to run it. To confirm the installation worked, run the `node` command in the command prompt. It will open the node console, which you can close by pressing `Ctrl + C` twice.

8. The last step is to install the PhoneGap library by running the following command at the command prompt:

   ```
   npm install -g phonegap
   ```

9. To verify we have installed PhoneGap properly, we simply run the `phonegap` command at the command prompt; this should return details about PhoneGap.

10. There is one last thing we need to do before running the test project. We need to create Android Emulator Start by running the following command:

    ```
    android avd
    ```
11. By default, there are no emulators, so we need to create one before starting. Press Create... which will open a window to create a new emulator. Enter the following settings and press OK:

![Edit Android Virtual Device (AVD)](image)

12. A new emulator should be listed. Close the application.
13. To verify we can build and launch the application in the simulator, let us create and run a quick sample application by running the following command:

```
phonegap create hello com.example.hello HelloWorld
cd hello
```
14. This creates a basic sample application, so next we need to run it to verify it works on the Android simulator.
15. To test that everything is working well, we are going to run the following command that should run Android Device Simulator, as selected before:

```
phonegap run android --emulator
```
16. You should see it running an example.
Summary

In this introductory chapter we covered a lot. We got a quick overview of what, as a framework, PhoneGap is and is not. Before we start to do any real work with PhoneGap, we need to have installed all the required tools that we covered in this chapter in detail, for all three major platforms.

After finishing this chapter, you should have a fully functional setup to start developing PhoneGap applications on your favorite operating system.

In the next chapter, we will cover the support for different platforms and the features that each platform supports. We will come to understand what PhoneGap Build is and how to test usability on native devices.
Where to buy this book
You can buy PhoneGap Essentials from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.