

# **Android PROGRAMMING THE FAST WAY**

The Fastest and Complete  
Android Programming Guide



Eng. Alexander Mosgov

# **Android Programming**

## **The Fast Way**

*Learn Android Programming*

*Start Coding TODAY*

*with the Ultimate*

*Android Programming for Beginners  
Guide*

by

Eng. Alexander Mosgov

**© Copyright 2015 by Alexander Mosgov- All rights reserved.**

This document is geared towards providing exact and reliable information in regards to the topic and issue covered. The publication is sold with the idea that the publisher is not required to render accounting, officially permitted, or otherwise, qualified services. If advice is necessary, legal or professional, a practiced individual in the profession should be ordered.

- From a Declaration of Principles which was accepted and approved equally by a Committee of the American Bar Association and a Committee of Publishers and Associations.

In no way is it legal to reproduce, duplicate, or transmit any part of this document in either electronic means or in printed format. Recording of this publication is strictly prohibited and any storage of this document is not allowed unless with written permission from the publisher. All rights reserved.

The information provided herein is stated to be truthful and consistent, in that any liability, in terms of inattention or otherwise, by any usage or abuse of any policies, processes, or directions contained within is the solitary and utter responsibility of the recipient reader. Under no circumstances will any legal responsibility or blame be held against the publisher for any reparation, damages, or monetary loss due to the information herein, either directly or indirectly.

Respective authors own all copyrights not held by the publisher.

The information herein is offered for informational purposes solely, and is universal as so. The presentation of the information is without contract or any type of guarantee assurance.

The trademarks that are used are without any consent, and the publication of the trademark is without permission or backing by the trademark owner. All trademarks and brands within this book are for clarifying purposes only and are the owned by the owners themselves, not affiliated with this document.

# Table of Contents

**Introduction**

**Chapter 1: Understanding Android**

**Chapter 2: Creating an Android  
Application**

**Chapter 3: Hello World**

**Chapter 4: Application Components**

**Chapter 5: Creating an excellent User  
Interface**

**Conclusion**



# Introduction

It is less than ten years that the Android operating system has been used commercially, yet with its presence in millions of devices across the world, more and more people want to learn how they can go about programming. The Android operating system is used on mobile devices (and other supported platforms including tablets) for applications.

Its significant growth can be attributed to its ability to help a user generate income through the creation of applications. Creating these applications requires some knowledge in programming.

This book is an excellent way for you to get started learning how to programme on an Android operating system. With the right information, you can begin creating apps that can carry out virtually any actions that you can conceive. A close study of this book will help you get started with creating your own apps, and making sure that they are not only informative, but also attractive and user friendly.

In addition, you will find out about what drives applications to work the way they do, by understanding the different components that make up an application.

It is important to get more knowledge on this operating system because one thing is for sure, it is here to stay. Android is a Linux Based Operating system, meaning that it is open source and available for free. This means that it is accessible for anyone who is interested in learning how to use it.

Android stands out from other operating systems for mobile devices and other supported platforms due to its ability to be customized. It allows for creativity and development of applications at a developers pace.

# Chapter 1: Understanding Android

The mobile phone has dramatically changed from the simple communication device that it used to be. In the past, mobile phones were used for two basic modes of communication, which were making calls and sending text messages. Today, mobile phones are like mini-computers, able to execute a variety of tasks and even to control lives.

With the advent of more technology, there is a new type of mobile device termed as the smartphone. A smartphone is more than a mobile phone, and that is because it has more capabilities. On a smartphone, you are able to download applications (referred to as apps), and these applications have an extensive range of capabilities.

In order to develop and use this apps, an operating system needed to be created. That is where the story of Android begins.

Google and a host of other companies came together and developed Open Handset Alliance. Open Handset Alliance are responsible for the creation and development of Android. Android is the Linux-based operating system that is used on mobile devices, which include smartphones and tablets. It is open source, meaning that it is free available to everyone.

Using the Android operating system, programmers are able to develop a host of applications that can be run on any devices which are powered by Android.

The first commercial version of Android was Android 1.0, which was released in September 2008. There have been several updates to this version over the years, one of the more recent being 4.1 Jelly Bean.

As Android is open source, the source codes are available under fee licenses. The Apache License version 2.0 is what Google uses to publish a significant amount of the code, and the rest of the code can be found under the General Public License version 2 from Linux.

The Android operating system competes with the Apple operating system (iOS).

## **Android Programming**

The demand for Android programmers has risen through the years as the Android OS begins to give iOS more competition in the apps market. With over one billion Android devices on the planet and counting, it is not hard to see why more programmers are starting to make apps for Android as a priority rather than as a secondary choice to making iOS apps.

Android has other advantages when it comes to programming. For example, the Android platform is open source and has by far the largest share of the smart devices market in the world. Android policies on app submission and device provisioning are very open and carry very few restrictions. This means that when you complete building your own app you and your friends will be able to use it immediately.

For programmers who are used to the iOS platform, learning how to program for Android should be easy. Though they differ substantially, they are similar in many ways as well. It will be an advantage to have on your next project. By knowing how to program for both platforms, you can create an application that exploits all their strengths and discards their weaknesses. The world is not just apple products anymore, the Android OS is taking over the smart devices market, and knowing how to program for both platforms is going to be invaluable.

You do not need much for you to start Android development, and all the tools you need are freely available on the internet. The following list of software is all you need:

- The latest Java Development Kit (JDK)
- Android Studio SDK
- Eclipse IDE for Java Developers (optional)
- Android Development Tolls (optional Eclipse plugin)

As the programming language used in Android programming is Java, it would be good if you had a background or some basic knowledge on the subject, but if you do not, there are plenty of tutorials available online that can help you grasp the basics.

Once you have managed to develop your Android application, you can begin to sell them through app stores online. The most popular app stores are Google Play and Amazon Appstore.

# Chapter 2: Creating an Android Application

The most important thing to do before you jump in and start coding for a new platform is to set up your environment. This is especially true for beginners who will undoubtedly hit some roadblocks along the way.

Android application development can be done on the latest versions of Microsoft Windows, Mac OS with Intel chips, and Linux operating systems. You will also have to install some software from the internet in order to get started with your Android programming.

The first thing to install before Android studio is the JDK, which is on the oracle website. You can download the Android SDK from the Android developer's page. Remember to check for updates for Android studio. Google is always updating it so the version that you install may not be the latest version.

## **The Android Virtual Device**

Once you have installed your software, you are almost ready to begin developing your Android applications. To do so, you will need to setup some Android development tools. This requires setting up a plugin. Your installation manager will guide you through the step by step process of setting up your plugin.

When you have completed those steps, you need to create a virtual Android device. With this virtual device, you will be able to test your Android applications to see whether they will work as expected. To create your virtual device, you need to have installed Eclipse IDE for Java Developers. To get the process moving, you should launch the Android AVD Managers using the Eclipse menu options. Find and use the New button and this will create a new Android virtual device. If you follow the instructions, you will see the create AVD button, and you should click on that.

When you successfully create your Android virtual device, then you are ready for Android application development.



## Chapter 3: Hello World

If you have a background in programming, then you are probably familiar with the phrase “Hello World”. It is typically the name given to the first program written in any language, and this instance is no exception. By the end of this book, you should be able to produce an app that you can share with your friends that can display your name.

One of the tools that you will use as a programmer when developing in Android is Android studio. This is an integrated development environment (IDE). When you open Android Studio, a pop-up window displays a list of tasks you can perform. Click on ‘Start a New Android Studio Project’ to begin. Under ‘Application Name’ enter ‘First Android Program’ and under ‘Company Name’ you can enter your name. Notice that the ‘Package Name’ field creates a reverse, domain-style name from your application name and company domain. The package name is an identifier used for the app using the same concept as bundle identifiers in iOS apps.

The Android SDK needs to optimize the app for a specific device. This is the next step of the setup process, which allows you to change the device type from the normal Android phone to TV and even Google Glass. There is also a handy minimum SDK menu that will show you the oldest version of Android that will be compatible with your app. It will also show you what kind of features your app will have, depending on the version of Android.

Once you have completed this process, you will have a basic “Hello World” app with all the source files that will allow you to open it. If you do not have an Android device, Android Studio comes with a free emulator that can set up a software-based version of any Android device. To create a virtual device, first open the AVD manager and select ‘Create Virtual Device’. Select one of the devices from the list, for instance, the Nexus 5, and select the Android version it should emulate.

Once you have verified your settings, the device should appear in the list. Clicking the play button in Android studio’s main view will open a new window that asks you to choose the device. Select the newly created Nexus 5 and your ‘Hello World’ app should open.

To insert your name into the application, find and open the file

*res/values/strings.xml*. This file contains three string resources, which are accessed from different places but are all stored together. Find and change the 'hello\_world' string by inserting your name and a short phrase. Something like `<string name="hello_world">Joe is now a programmer!</string>`. The next time you launch the app your name should appear.

Now that you have created your first app, and set up your working environment properly, you can start to explore the world of Android. Remember to follow various development communities as they always have a wealth of knowledge to share, and it is never too late to start conversing with the experts.

## **Running your App**

What would seem obvious once you have created your app is to immediately run it on a physical Android device. However, if you do not have one handy, you can still run your app using an emulator.

You can easily find your emulator on Android studio. The emulator is able to run apps on your computer using a software based Android device. If you want to compare or check out different specifications for your app, you can set up multiple emulators. With then you can check different screen sizes or platform versions. If you were to do this with the devices themselves, you would need a large number of them to test your app on – a rather impractical way of making sure your app is operations.

# Chapter 4: Application Components

Now that you have the basic know-how of how to create an application for Android, you need an understanding of what the components that make up the application are. For programmers, understanding the components makes it easier to amend or update aspects of the application should the need arise.

Components are essentially the building blocks that make up an Android application. There are four primary components that can be used. These shall be discussed in this chapter.

## Activities

The first component is known as activities. This component dictates what the user interface for the application will be. It also handles the user interaction that occurs on a smartphone screen.

The activity is representative of a single screen that has a user interface. Consider an application for reading stories. One activity might take you to a screen where you are able to see a list of all the stories that are available. Another activity may show you the abstracts of each of the stories, and another activity may show you the pages within the story of your choice. You are only able to see one activity at a time, and you can program your application to present one particular activity when it is launched.

The Android system will usually start a program with an activity, starting with a call on callback method. This means that there will be callback methods that start up an activity and then those that tear down the same activity. You need to be aware of the callback methods to understand those that you need to use. These callback methods will affect the way users experience your app. They include the following: -

`onCreate()`                      When the activity is first created, this is the initial callback and call.

`onStart()`                      Once the activity is visible to the user, this callback is called.

`onResume()`                      Upon the user interacting with the application, this is called.

`onPause()` When the current activity has been paused, and the previous activity resumed, the paused activity is not able to receive any user input. It cannot execute any code when it is called.

`onStop()` When the activity is no longer visible, this callback is called.

`onDestroy()` When the system is about to destroy an activity, this callback is called.

`onRestart()` Once an activity restarts after it has been stopped, this callback is called.

## Services

For you to see the working of an application, there must be something happening in the background. This is what services are associated with.

When you have long running operations, services become very crucial for Android operating systems. Take for example you are running an anti-virus programme on your smartphone, tablet or other Android supported device. The service will enable the anti-virus programme to run continuously in the background, while you use another activity within the device. Your interaction with the activity will not be halted because of the service in the background.

Services occur in one of two states. These states are explained as follows: -

*Started* – When an activity starts an application component by calling `startService()`, then a service is started. It is able to run in the background indefinitely, even though the component that started it may be destroyed.

*Bound* - An application component can bind a service by calling `bindService()`. When a service has been bound, it is possible for the client-server interface to interact with the service, send out the request and receive results from processes with interprocess communication.

When programming, you may consider creating your own service. Should you want to, create a Java class which is able to extend the service base class or one of the existing subclasses. It is the service base class that will define

the other callback methods. Although there is no need to implement all the callback methods, it is important that they are understood. Like the activities, they will affect the way your users experience your app. Some of these callback methods are outlined below: -

**onStartCommand()** When the service is to be started by another component like an activity calling `startService()`, the system will call for this method. Should you want to stop the service, and then you must call `stopSelf()` or `stopService()` methods.

**onBind()** Should another component want to bind with the service by calling `bindService()`, the system will call this method. Once you choose to implement this method, it is required that you offer an interface that the clients can use to communicate with the service. If you decide that you do not want to implement binding, then you should return null.

**onUnbind()** If all the clients should disconnect from a specific interface that is published by the service, the system will call this method.

**onRebind()** When the system receives a notification that there are new clients connected to the service following being notified that all had discontinued using the `onUnbind` Intent, the system calls this method.

**onCreate()** This is a call that is required to perform a one-step setup. It is called when the service is first created using `onStartCommand()` or `onBind()`.

**onDestroy()** When the service is being destroyed or is no longer in use, the system will call this method. This service is important for the implementation to facilitate the cleaning up of any resources including threads, receivers, registered listeners and so on.

## **Broadcast Receivers**



Your Android application may be within an exclusively Android device, or may be found on a device that has other applications as well. The broadcast receivers are there to take care of the communication that occurs between the Android operating system and applications.

These will act as not only receivers of communication, they will also respond to communication. Consider a situation where an application is downloading information or an update to the system. The broadcast receivers will let the other applications know that there has been some data downloaded onto the device, and it is available for them to use. Therefore, the appropriate actions can be taken by these apps.

Every message is broadcasted as an intent object. The messages are also sometimes referred to as an event.

In order for system broadcasted intents to work, two steps need to be made. These are: -

- Creating the Broadcast receiver

This is where a broadcast receiver is implemented as a subclass of the BroadcastReceiver class and overriding the onReceive() method where each message is received as an Intent object parameter.

- Registering the Broadcast receiver

The application will listen to a particular broadcast intent by registering a broadcast receiver in the AndroidManifest.xml file.

## **Content Providers**

Inevitably, you are likely to require and store data for your application. The content providers are the component that will handle all your data and database management issues.

When applications need data from one another, then the content provider comes into play. Usually, this data is stored on the device within a filing system or even over a network.

It is expected that different applications will run with their own processes and permissions, which in turn will keep an application's data hidden from another application. However, a situation may arise where it becomes necessary to

share the data across applications.

With a content provider, you are able to centralize content where a host of different applications can access the information when it is needed. This is quite similar to the query for information from a database. With the content provider, you are able to query the information, as well as edit its content, add or delete the content. You do this by using the following methods: insert(), update(), delete() and query().

In order to have your content provider working, you need to override the content provider class. You can use the following list of methods for reference: -

onCreate() -	When the provider starts, this is the method called.
query() -	When a request is received from a client, this method is used. The result is returned as a cursor object.
insert() method.	The content provider has a new record inserted with this method.
delete() provider	This method deletes an existing record from the content provider
update() using this method.	An existing record from the content provider is updated using this method.
getType() given URI.	This method returns the MIME type of the data at the given URI.

The above are the four primary components that come into play when developing and understanding applications. There are also some secondary components which should be considered, and these are listed below: -

- Fragments – These are able to represent the behavior of a user interface within an activity or a portion of the user interface.
- Layouts – You can use this component to view the hierarchies that control the screen format, as well as the appearance of the views.
- Views – These are the user interface elements that are drawn onscreen. They include the lists forms, buttons and so on.
- Resources – These are the external elements that may affect the app. They include constants, strings, and drawable pictures.

- Intents – When it comes to connectivity, the intents are vital. They are messages which wire components together.
- Manifest – This is the configuration file for the application.

# Chapter 5: Creating an excellent User Interface

Users are more likely to use your app, and recommend it to others if it has an excellent user interface. In fact, you will find that most of your time is spent trying to create an attractive user interface. Here is a step by step guide that you can follow to create an excellent user interface.

1. Find out where the code is for your application. For Android, this would be an activity.
2. Work out the way that the user interface components are being represented, how they are created and how the user interface can be connected to the code. Android operating systems will use a hierarchy of view objects. What is needed, in this case, is finding all those objects that are representing each user interface component, and from there, adding on event handles.
3. Now, you should explore the components that are available for building your user interface. There is a host of simple components that should be available, including buttons for example. In addition to this, there are some components that are more complex such as listview. Although it may take some time and experience to master the complex components, the simple components should all be used where they are applicable.

When you are ready to create your user interface, you can select one of three ways. These are: -

1. Creating all of the view objects in code. An activity will contain views and view groups. A view is basically a widget which is able to appear on the screen. When there are one or more views, they can be grouped together into a viewgroup.

With a viewgroup, you can view the layout in which you can order the appearance and the sequence of views.

2. Creating an XML file then having the system convert the file to

objects at run time.

3. Use the designer to interactively create the XML file if you are using Android studio.

### **Android Studio – A basic user interface**

Here is a method for you to create a simple user interface. Start Android studio then create a new blank activity project. Label it UItest. It is this file that you shall try out the different options possible until you are able to accept your user interface. You will be asked to accept defaults, accept them all and your project will be created.

The next step you should take is open the file `activity_main.xml`. It is in the `app/res/layout` folder. This will lead to the designer opening. You will the top three sections of the Palette have the most vital components for your user interface.

One is the layouts, which are the containers for other components. You will be using this once per activity. Next, the widgets section where you should spend some time. This is because it likely contains the most frequently used components, including buttons, checkboxes, etc. The third section will have text fields, which are a set of text input components.

Once you have chosen your components, you need to use the designer to discover and set the properties that will make them look the way you want. You should also learn how to work with properties in code so that you can modify a component's looks at the run time.

You will find that the default layout is called `RelativeLayout`. This is amongst the most sophisticated of layouts, so it may not be ideal for simple layouts, though it is good for practicing with. The other layouts that are supported by Android include the `LinearLayout`, `AbsoluteLayout`, `TableLayout`, `FrameLayout` and `ScrollView`.

The `LinearLayout` will arrange the views in a single row or a single column. When using the `AbsoluteLayout`, you are able to specify what the exact location of its children should be. The `TableLayout` will group the views into columns and rows. Within each of the rows, several views can exist. Every view that is placed in a row forms a cell. The overall width of every column will be determined by the largest width of each cell in that column. The `RelativeLayout` will allow you to specify how the child views are positioned



in relation to each other. The `FrameLayout` can be considered to be a placeholder on the screen which you can use to display a single view. Finally, the `ScrollView` is a specialized `FrameLayout` as it allows for the users to scroll through a list of views which occupy more space than the physical display.

The `RelativeLayout` is able to support nine different positions for a component. These are at the Top: left, center and right, Centre: left, center and right, and Bottom: left, center and right. These will allow you to set components automatically on the page.

As you progress with your skills, you will be able to set your own margins for each position manually.

Android programming has various layers and the more that you practice, the easier it becomes.

# Conclusion

Being a beginner at programming does not mean that you cannot easily learn how to go about it. One of the best ways that you can start is by programming for Android devices.

With the help of this book, programming can be done following a sequential process. It is important to have the right software downloaded on your system of choice before you begin with the programming effort. Then make a decision as to what type of application you would like to create and begin to practice.

There are many programs available to support your learning and on which you can practice on. Like any other skill you would choose to develop, Android programming requires a significant amount of practice and dedication.

With more than 70% of the smartphones in the world, in 190 countries, using the Android operating system, it is no wonder that more and more people are learning Android programming.

This book explains the basic facts about Android programming to you. Coding is not everything – a depth of understanding for what is being done and why will make it easier to get any application you are developing right. With this book, your programming skills are more holistic, making it easier to create sustainable applications.

At this point of the book, you should be able to develop your own application, starting from the environment that you setup, and including a good layout. Remember that in addition to increasing your technical programming skills, you can commercialize your application and get a reward for your efforts.

**\*\*\*\*\*Bonus\*\*\*\*\***

**2 chapters of**

**HTML5**

*The Fast Way*

The Fastest and Complete HTML5  
Programming Guide

# Chapter 2: Introducing HTML5

HTML5 is gaining both speed and popularity and is the future! Many massive software companies such as Apple, Microsoft, and Google have leant their support to this web-page writing language. You are wise to learn it, and learn it well. It is easy to learn and we will cover many examples along each step of the process.

## The Purpose of HTML5

HTML allows you to create your very own web site. In more technical terms, HTML is a markup language used to describe web documents (pages). HTML is an abbreviation that stands for Hyper Text Markup Language. This language is considered the language of the web as it tells browsers exactly how to display the contents of your web page. To do so, it uses special instructions to let the browser known when a paragraph begins, which word is must be italicized, and so forth. These instructions come in the form of tags.

### Exercise 1: Finding the Source Code of a Website

Open up your preferred browser, find a website, and check out its HTML source code.

To find the source code of a website using Safari you must:

Open the browser and Select Preferences

Under preferences select Advanced

At the bottom of the Advanced options check the box next to "Show Develop Menu bar" .

Find a website, select Develop from menu, and choose show Page Source. This will allow you to view any websites source code.

Every single website online has an HTML source code that can be easily viewed. In fact let ' s make that your first exercise assignment.

Upon doing so you will find something that looks like the image to the left. Don ' t let all of this text scare you. It will make sense to you soon once we break it down.


# Chapter 3: Creating and Understanding an HTML File

Standing alone, an HTML page is nothing more than a text file consisting of letters, numbers, punctuation, and special characters, as seen in the example above. As such you can easily create or view an HTML document using a simple text editor like Notepad.

If you're using a Windows/Vista computer, simply click your start menu button and type “ Notepad ” into the search box. It should quickly appear through this search. On a Mac computer, you can use TextEdit, which can be found under Applications.

Using just these simple text editors you can start writing your content and even test these pages all without using a live website. In order to write the content, you will need to learn the basic elements of the HTML standard.

Each HTML document starts with a [DOCTYPE declaration](#). This declarations defines the document as an HTML. Together we will dissect the example below. The source code below results in the image on the right. Web visitors would not see the below code unless they use the view source code feature we discussed previously, they would see the result.

Simple HTML document	Result
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; &lt;title&gt;Page Title&lt;/title&gt; &lt;/head&gt; &lt;body&gt;  &lt;h1&gt;My First Heading&lt;/h1&gt; &lt;p&gt;My first paragraph.&lt;/p&gt;  &lt;/body&gt; &lt;/html&gt;</pre>	

## Tags

Angle brackets are used exclusively in HTML formatting, so be sure to familiarize yourself with them as you will be using them frequently. They are located right above the comma and the period keys on the keyboard. They look like < and >. These tags let the browser know how you want your text to be formatted.

The opening and closing of angle brackets creates an HTML tag. A tag is created by giving it a name (writing code between the brackets). Tags come in pairs with one key distinction being the slash



```
< > | simple_web_page | Source Code
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <link rel="shortcut icon" href="/favicon.ico">
6 <link href="/styles.css" rel="stylesheet" type="text/css">
7 <title>Simple Web Page</title>
8 <meta name="description" content="Simple Web Page - HTML and CSS example web page." />
9 <meta name="keywords" content="HTML, CSS, webpage, example, sample" />
10 <script type="text/javascript">
11
12     var _gaq = _gaq || [];
13     _gaq.push(['_setAccount', 'UA-13243213-1']);
14     _gaq.push(['_trackPageview']);
15
16     (function() {
17         var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
18         ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-
19             analytics.com/ga.js';
20         var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
21     })();
22 </script>
23 </head>
24 <body>
25 <div class="wholeheader">
26 <div class="header">
28 <div id="search">
29 <!-- Google Search start -->
30 <div class="cse-branding-bottom" style="color:#000000">
31 <div class="cse-branding-form">
32 <form action="http://www.google.com/cse" id="cse-search-box" target="_blank">
33 <div>
34 <input type="hidden" name="cx" value="partner-pub-7808627488179486:zgxnonvz5c2" />
35 <input type="hidden" name="ie" value="UTF-8" />
36 <input type="text" name="q" size="31" title="search text"/>
37 <input type="submit" name="sa" value="Search" title="search submit button" />
38 </div>
39 </div>
40 </div>
41 </div>
42 </body>
43 </html>
```

symbol /. The first tag is known as a *start tag*, it switches on the effect you need such as starting a new paragraph. The second tag is known as an *end tag*. It stops the effect, such as finishes the paragraph. Everything between the start tag and the end tag is referred to as an element. Take a look at the tags in our example. I have added colors to help you quickly spot the pairs.

In the example above we see the DOCTYPE declaration tag **<!DOCTYPE html>**. This designates the document as an HTML. We then see the start tag **<html>** following the DOCTYPE declaration and at the very end of the HTML document we see the end tag **</html>**. The text found between the **<html>** and **</html>** tags is describes an HTML document.

Everything that is written between **<head>** and **</head>** gives us information about our document. The next tag that is in our example is the title tag (**<title>** and **</title>**). It provides us with the document 's title. Following the title we see **<body>** and **</body>**. Everything in between these tags designates the visible page content. Tags **<h1>** and **</h1>** describe the heading.

The tag **<p>** designates a paragraph tag. At the beginning of a paragraph you would place the **<p>** and at the conclusion of the paragraph you would place **</p>**. Notice that they are identical apart from the slash. Everything located between these two tags would then be formatted as a paragraph.

## Headings

Headings are designated using **<h1>** all the way through **<h6>**. The numbers next to the letter h indicate how many headers are included. Headings are important as they help break up and organize your content.

Only use headings for headings though. Do not use this tag simply to make font bigger or to make it bold. Search engine often use your headings to index both the structure and content of your web pages.

## HTML5 Code Example and Result

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
    <h1>We just learned how to create headings</h1>
```

```
    <h2>Creating Headings is easy</h2>
```

```
    <h3>This is heading 3</h3>
```

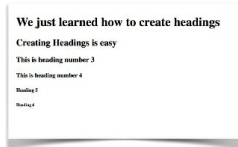
```
    <h4>This is heading 4</h4>
```

```
    <h5>Heading 5</h5>
```

```
    <h6>Heading 6</h6>
```

```
</body>
```

```
</html>
```



### Exercise2: Creating a Simple HTML document

Step 1: Open your Text Editor (either Notepad or TextEdit).

Step 2: Type up the exact example of the simple HTML document presented above.

Between the heading tags (<h1> and </h1>) type in Learning HTML Tag Review.

Between the paragraph tags (<p> and </p>) type up what you have learned about tags so far. Practice writing more than one paragraph. To do so simply insert a paragraph

## Writing &

## Viewing your First HTML Document

Now that you have written your first HTML document how to you view it?

If you are using Notepad follow the following steps:

Chose File, then Save As.

In the Save As dialog box, chose UTF-8 in the Encoding list.

In Save as type box, choose All Files instead of just text documents so you can see your files in the file list.

If using TextEdit:

Choose Format, then chose Make Plain Text

Then chose File, Save, and then Save As

Chose UTF-8 in the plain Text Encoding.

Make sure you save your file using an .htm or .html extension.

Now that you have saved the file using an .htm or .html extension, you should be able to easily open the file by double-clicking on the file name.