



# make your own Android App

your unofficial intro to  
MIT App Inventor

by James Sherar

**By James Sherar**  
**<http://arcade.jamessherar.ca>**

**Edited by Justin Pot**



**This manual is the intellectual property of MakeUseOf. It must only be published in its original form. Using parts or republishing altered parts of this guide is prohibited without permission from MakeUseOf.com**

Think you've got what it takes to write a manual for MakeUseOf.com? We're always willing to hear a pitch!  
Send your ideas to [justinpot@makeuseof.com](mailto:justinpot@makeuseof.com); you might earn up to \$400.

# Table Of Contents

---

Introduction	4
0.1 – Who Is This Guide For?	4
0.1.1 – Educators	4
0.1.2 – Non-Programmers In General	4
0.1.3 – Advanced Programming (Professional Programmers)	4
0.1.4 – Prototyping	5
1. System Requirements	6
1.1 – Hardware & Software	6
1.2 – Java	6
1.3 – Google Account	8
1.4 – The App Inventor Software	8
1.5 – Resources	8
2. Development	9
2.1 – Hello World	9
2.1.1 – The Design Window	10
3. Coding And Testing	13
3.1 – Accessing The Blocks Editor	13
3.2 – Accessing The Emulator	14
3.3 – Coding And testing With The Blocks Editor And The Emulator	15
3.3.1 – Adding A picture	16
3.3.2 – Adding a Message	17
4. Testing	19
5. Sharing And Marketing	19
5.1 – Versioning	19
5.1 – Sharing	19
5.2 – Google Play – The Android Market	20
6. Conclusions	21

# Introduction

---

If you're like many people these days, you probably spend at least some of your day interacting with the Internet through apps on your Android device for anything from banking, games, and event planning to listening to music, texting and just about anything you can think of. For most, the underlying technology that makes an app 'tick' is shrouded in mystery. This has been a boon for programming experts and has spawned a very profitable niche for professional programmers who are paid to research, develop, and build these apps. But what if you have an idea for the "next big thing" – or even the "next little thing" for that matter – with no programming skills to speak of and, for whatever reason, you don't want to hand over your idea to a professional and pay to have it developed? In the past, if you weren't an app programmer yourself, you would have had the option to (a) do nothing, of course, (b) be brave and trust your idea in the hands of a developer, or (c) develop your programming skills and learn how to build the darn thing yourself. Well, now there is hope for non-programmers. Recently, thanks to a collaboration between Google and MIT, the world of mobile app creation has been opened to everyone with App Inventor, which is a web-based development platform, making option (c) not so out of reach for many.

## 0.1 – Who Is This Guide For?

With this free and open-source software, everyone will have the ability to become contributors to the digital world that surrounds them. App Inventor makes mobile app development highly visual, and highly intuitive. It is an easy and fun way for the uninitiated to learn about computer programming, and is at the same time a productive tool for advanced programmers alike. It is worth mentioning that despite being in its infancy – that is to say beta phase – this platform still offers a robust set of programming tools for all levels of programming ability and is ideal for use in education. Indeed, a major angle being promoted by MIT/Google is its instrumentality in teaching and for introducing anyone to programming, particularly young people, in say a high school setting. Although, it has a much broader target audience than that. That is, adult non-programmers who wish to teach themselves, and professional programmers.

### 0.1.1 – Educators

With the profusion of computers in our daily lives, I daresay all high schools now provide introductory computing science courses to students. App Inventor is an ideal vehicle to that end. By virtue of the visual nature of the platform, students are able to very quickly "snap together" their first program. Not only that, and perhaps more importantly, it's a fun and engaging way for kids to learn about computers and how they can become contributors of useful and purposeful digital content, not merely users of it.

Myriad guides, teaching resources, and testimonials from professional educators as to how to develop a lesson plan for young learners can be found at <http://appinventor.mit.edu/teach/> which gives access to a sort of curriculum-in-a-box framework for classroom and workshop settings, video tutorials, and a forum group and FAQ section specific to educators.

### 0.1.2 – Non-Programmers In General

With the abundance of tutorials, and other online resources, self-guided learning is made easy. Non-initiates can progress rapidly and to the point where they are creating advanced and relevant programs of their own design.

The App Inventor website provides access to a wealth of resources for self-guided learning:

- <http://appinventor.mit.edu/explore/blog.html> - a blog
- <http://appinventor.mit.edu/explore/content/tutorials.html> - a treasure of online tutorials
- <http://beta.appinventor.mit.edu/forum/index.html> - an online user forum

### 0.1.3 – Advanced Programming (Professional Programmers)

It may also be used for more than just as a teaching aid. Since the platform offers all the high-level components required for building robust apps as well as programming primitives, App Inventor should not be relegated as "toy" technology. With App Inventor, programmers have a way to build relevant programs that harness any and all of:

- GPS, motion, and orientation sensing

- *SMS texting*
- *Barcode scanning*
- *WiFi, Bluetooth*
- *Speech recognition and TextToSpeech*
- *Database and custom web-database connectivity (aka cloud storage)*
- *Audio/Video media*
- *Web connectivity*
- *Social site connectivity*
- *Lego's Mindstorms NXT technology*

Other technologies are also being developed on a continuing basis. For example, a couple of forthcoming components include online game server communication, and web voting capabilities.

### 0.1.4 – Prototyping

Although not officially part of the of Google/MIT purview, one thing that almost immediately came to mind for this author is that, if for nothing else, the App Inventor could be an ideal tool for consultants and systems programmers to rapidly prototype new products for review by clients, thanks to the visual editor. With the visual editor at your disposal it is possible, even easy, to create a mock up of a program without necessarily having to write even a single line of code.

So, welcome to the world of app development for Android devices. In this manual, we introduce App Inventor by covering system setup and basic aspects of the App Inventor online development environment, including an archetypal “Hello World” program, and we conclude by exploring options available for the distribution of your new apps.

# 1. System Requirements

App Inventor is an Online Development Environment (ODE), meaning app creation occurs directly in a web browser. So if you have a reasonably up-to-date system, you should already have most of the prerequisites installed. If not, never fear. The setup should be pretty straightforward.

## 1.1 – Hardware & Software

Preliminarily, you should verify that your system is one that is supported, that your browser is one of those currently supported, and that your Java is up to date and working on your computer. Incidentally, you should also make sure that you don't have a script blocker running in your browser.

To begin with, your OS should be one of:

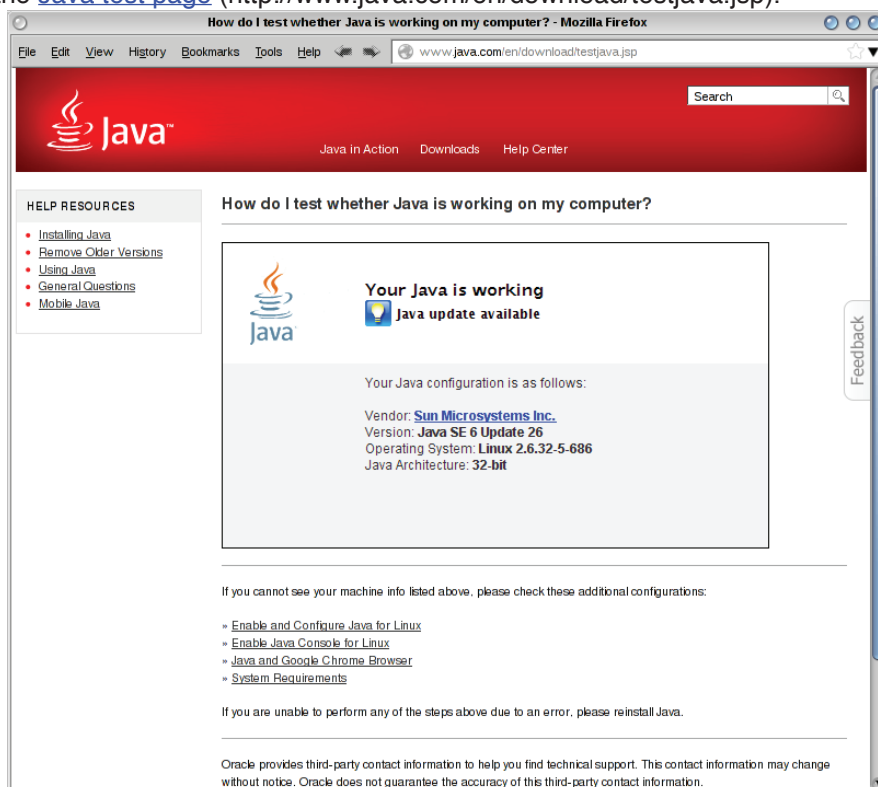
- *GNU/Linux (Debian 5, Ubuntu 8.04 or later)*
- *Mac OS/X 10.5, 10.6 or later*
- *Windows XP, Vista, 7 or later*

Minimally, you should be running one of the following browsers:

- *Firefox 3.6 or later*
- *Chrome 4.0 or later*
- *Apple Safari 5.0 or later*
- *IE 7 or later*

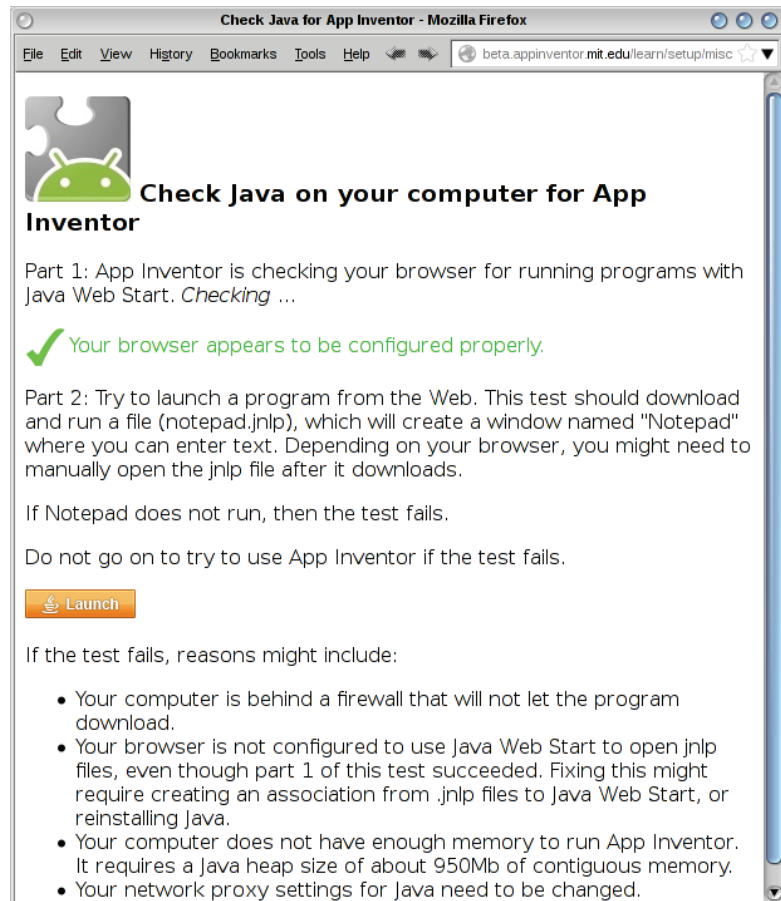
## 1.2 – Java

In terms of Java, the official guidelines state you should be using Java 6 (i.e. version 1.6). At the time of this writing Java 7 was available, however for our purposes, we will assume the use of Java 6. To verify that Java is working on your machine, go to the [Java test page](http://www.java.com/en/download/testjava.jsp) (<http://www.java.com/en/download/testjava.jsp>).



If your computer is configured properly, you should see the message “Your Java is working...”. If not, you should follow the instructions on the Java site for your system before proceeding.

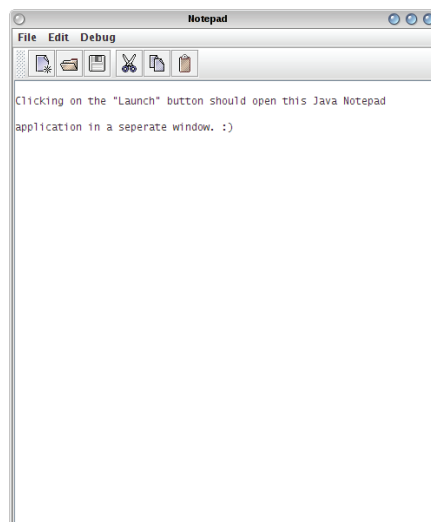
App inventor also requires that the Java Web Start framework exists on your computer, and your browser is configured to launch Java applications with it. You don't have to worry too much about the technical jargon here, just point your browser to the App Inventor [Java Web Start test page](http://beta.appinventor.mit.edu/learn/setup/misc/JWSTest/AppendixJWSTest.html) (<http://beta.appinventor.mit.edu/learn/setup/misc/JWSTest/AppendixJWSTest.html>) in order to run the test. “Part 1” of the test will run automatically when the page loads, and if your browser is configured properly, the message, “Your browser appears to be configured properly.” will be displayed.



To run “Part 2” of the configuration test, on the same page, click on the “Launch” button.



This should open a simple Java “notepad” application which you can then close.



## 1.3 – Google Account

Another thing you must have is a Google account. This is because your project(s) will be stored, at least during development, in the Google Cloud. The upshot of this is that it also means you can access your projects from anywhere with an Internet connection. If you have one, make sure you're logged in at this point. If you don't already have one, you can set yourself up at <https://gmail.com/>.

## 1.4 – The App Inventor Software

The App Inventor platform is free and open source, meaning that the software is available at no cost from the App Inventor website. Depending on your system, choose the appropriate link below and follow the installation instructions for your OS.

- GNU/Linux: <http://beta.appinventor.mit.edu/learn/setup/setuplinux.html>
- Mac: <http://beta.appinventor.mit.edu/learn/setup/setupmac.html>
- Windows: <http://beta.appinventor.mit.edu/learn/setup/setupwindows.html>

## 1.5 – Resources

Before moving on it's worth pointing out the other various resources at your disposal:

- If there is anything you need help with in the setup described in this section, stop by the troubleshooting page at <http://beta.appinventor.mit.edu/learn/troubleshooting.html>.
- The FAQ page can be found at <http://beta.appinventor.mit.edu/learn/userfaq.html>.
- Otherwise, try the "App Inventor User Forum" at <http://beta.appinventor.mit.edu/forum/>.
- Documentation (aka Help pages), can be accessed at <http://appinventor.mit.edu/explore/content/reference-documentation.html>.

## 2. Development

Development can be broken up into three phases which can, and in fact should, be undertaken concurrently.

1. *Interface development*
2. *Coding*
3. *Testing*

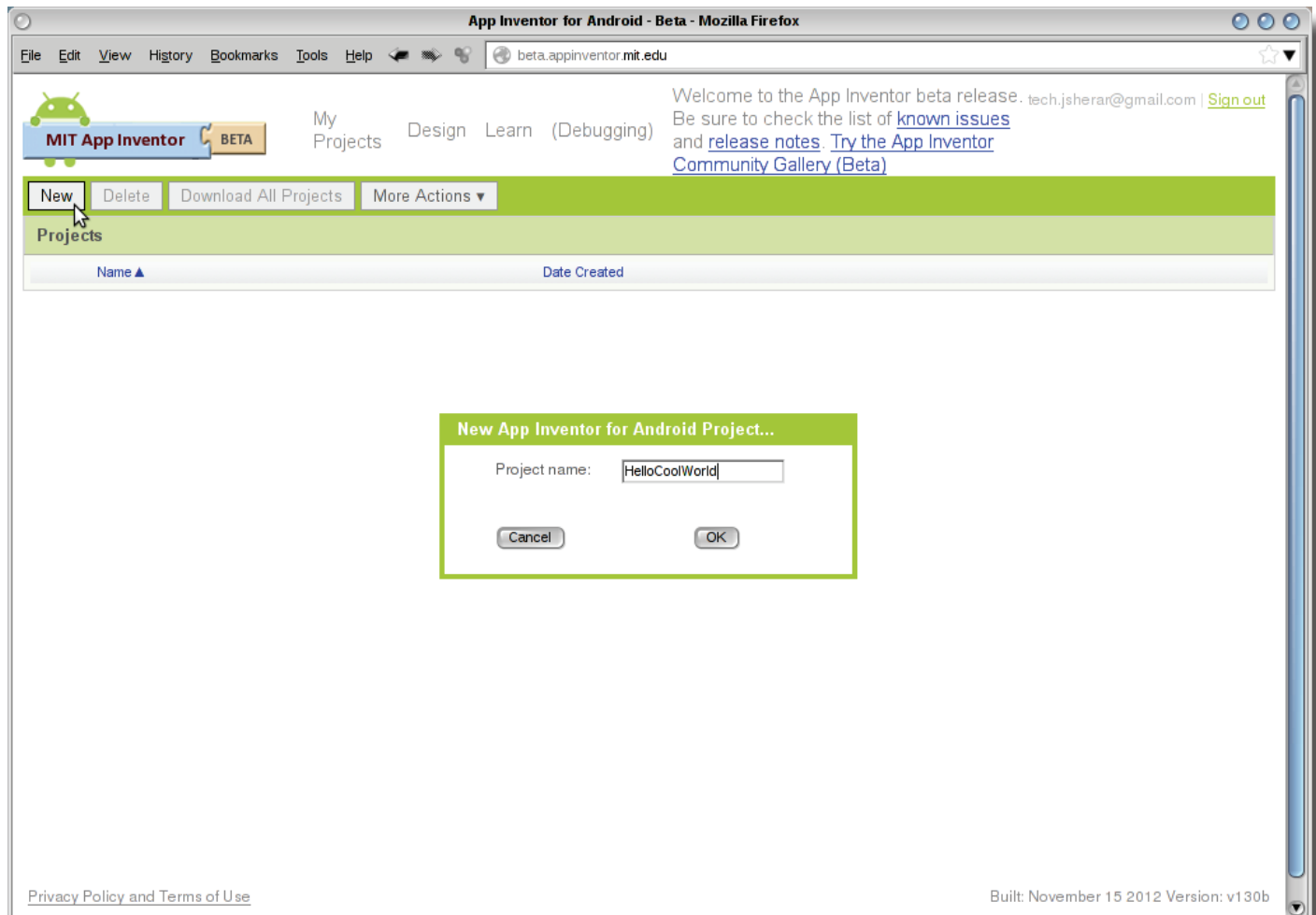
Although these will be done stepwise for the purposes of this manual, in the end you will see how convenient and important it is that the phases can be done in any order. Additionally, while app creation can be done with a connection to a mobile device, since some may have different ways of connecting their mobile device, to bypass any connectivity issues we will be using the emulator included in the software. Now, before going any further, make sure you're logged in to your Google account.

We will also be requiring the picture of a cute cat found at <http://www.publicdomainpictures.net/pictures/10000/vel-ka/peek-a-boo-cat-eyes-23441279626324N0rf.jpg><sup>1</sup> so you can download it as well at this point. Henceforth, we'll refer to it simply as the peek-a-boo-cat-eyes pic.

### 2.1 – Hello World

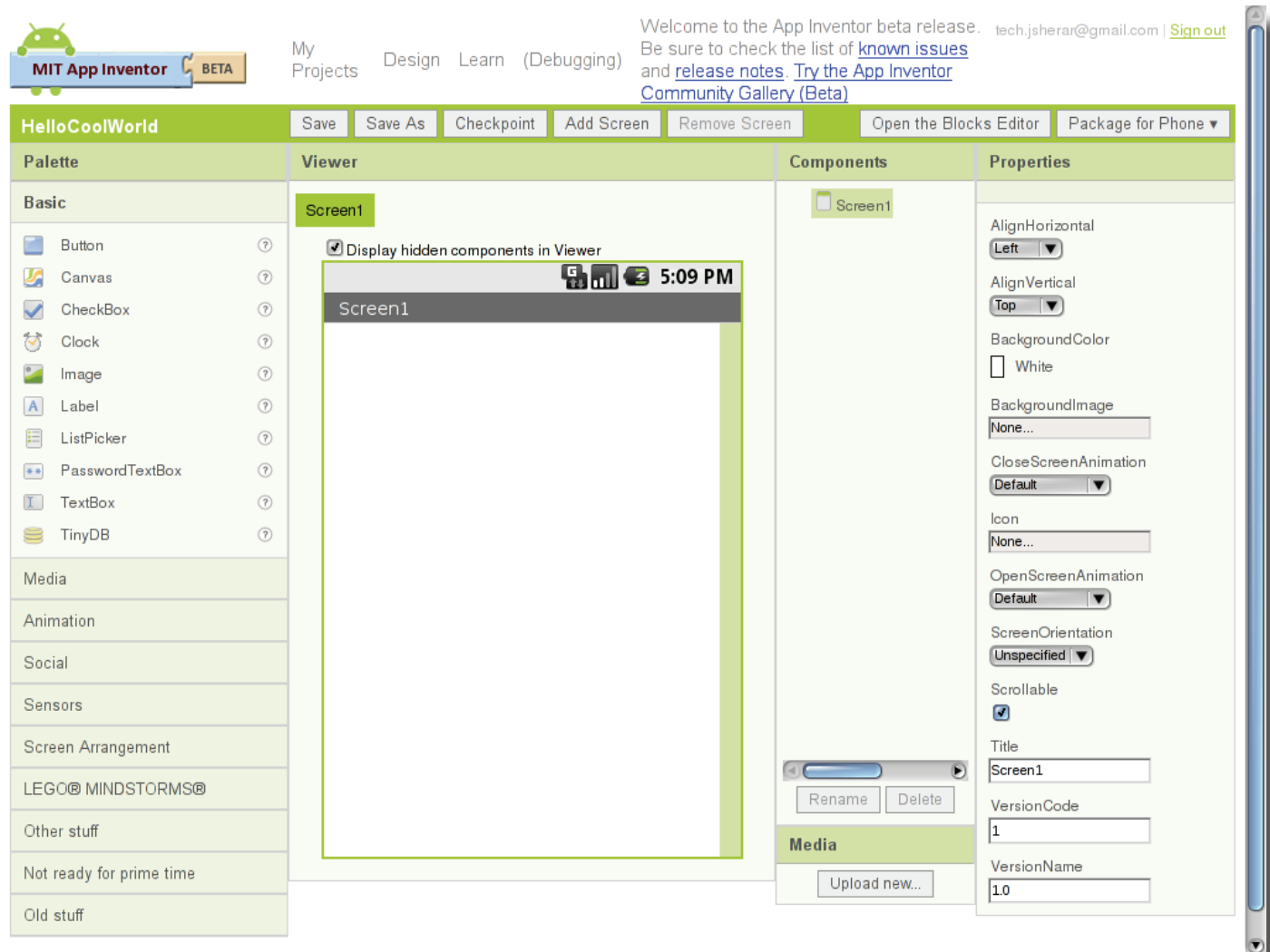
The canonical “Hello World” app we present conveys a cursory overview of coding techniques, input, output, and user interaction to familiarize you with the development environment.

To get started, head to <http://appinventor.mit.edu/> and click on the “Invent” button. This will bring you to your personal projects dashboard. Click on the “New” button in the top left corner to start a new project, and give it a name. You can call it anything you want but we'll name it “HelloCoolWorld” – all one word. Click “OK”.



## 2.1.1 – The Design Window

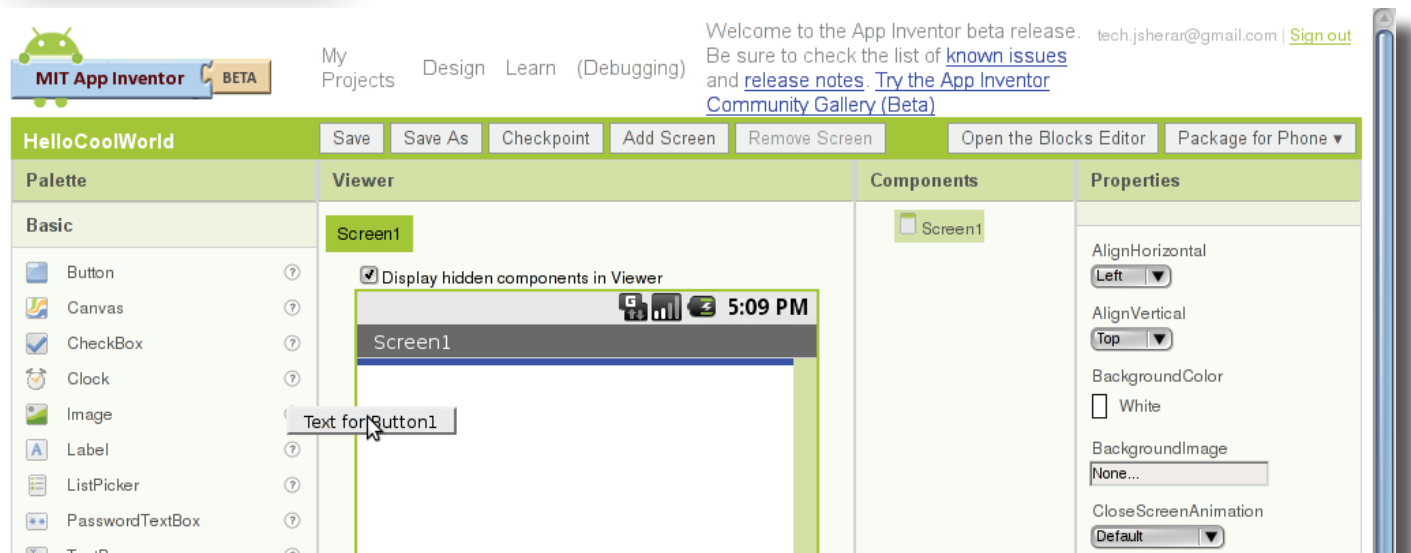
The Design window should appear in your browser once you enter the name of your app and click “OK”. The Design window is where you begin creating how your app will look, the user interface (UI).



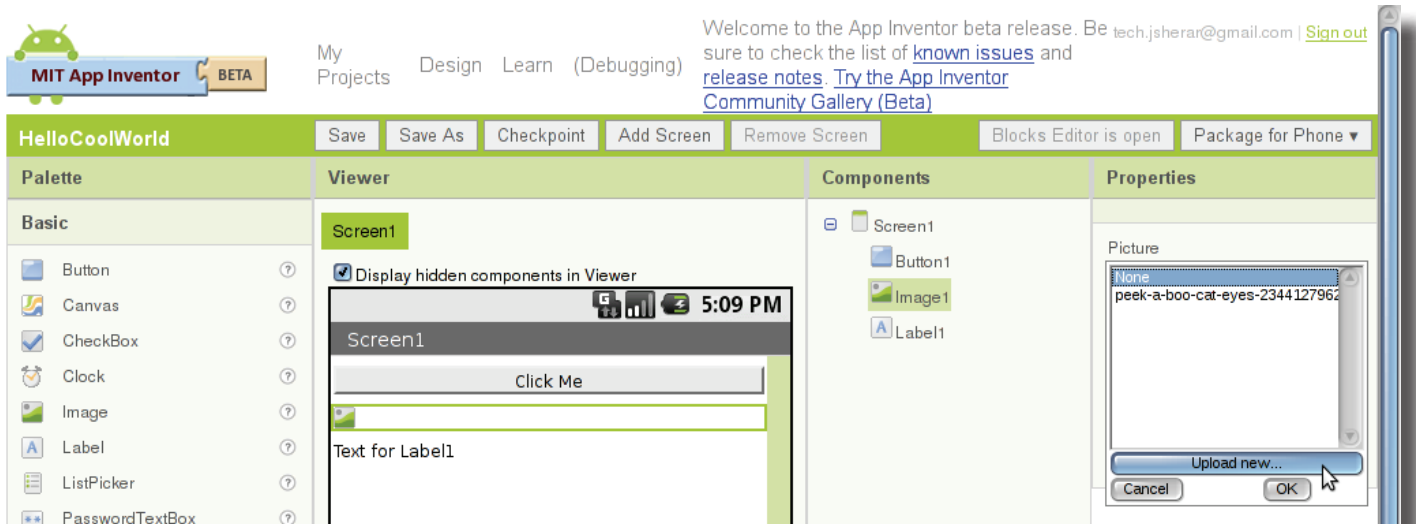
In the left panel of the screen, you find the Palette which works like many other visual platforms, where you drag-and-drop the components you require from it onto a mock-up of a mobile screen. For example, these can be text boxes, buttons, labels, database connectivity, media, or sensor objects – anything you want or need for your app to function. The components are grouped by their functional category – Basic, Media, Animation, Social, Sensors, etc. – and clicking on the group heading reveals the components therein. As you’ll notice, there’s lots of exciting stuff to play and experiment with.

In the centre you have the Viewer panel with a mock-up of the device screen named Screen1 where you’ll place component objects that make up your app, like buttons and text boxes. To the right of that, you have the Components panel which provides a listing of all the objects in your app including the object Screen1, while the Properties panel allows you to have access to and set certain parameters of objects like font, colour, or text.

First, we’ll need a way for the user to interact with our app, so click and drag the Button object from the Palette onto Screen1 in the Viewer panel, and set its Text property to “Click Me”, and switch its Width parameter to Fill parent...

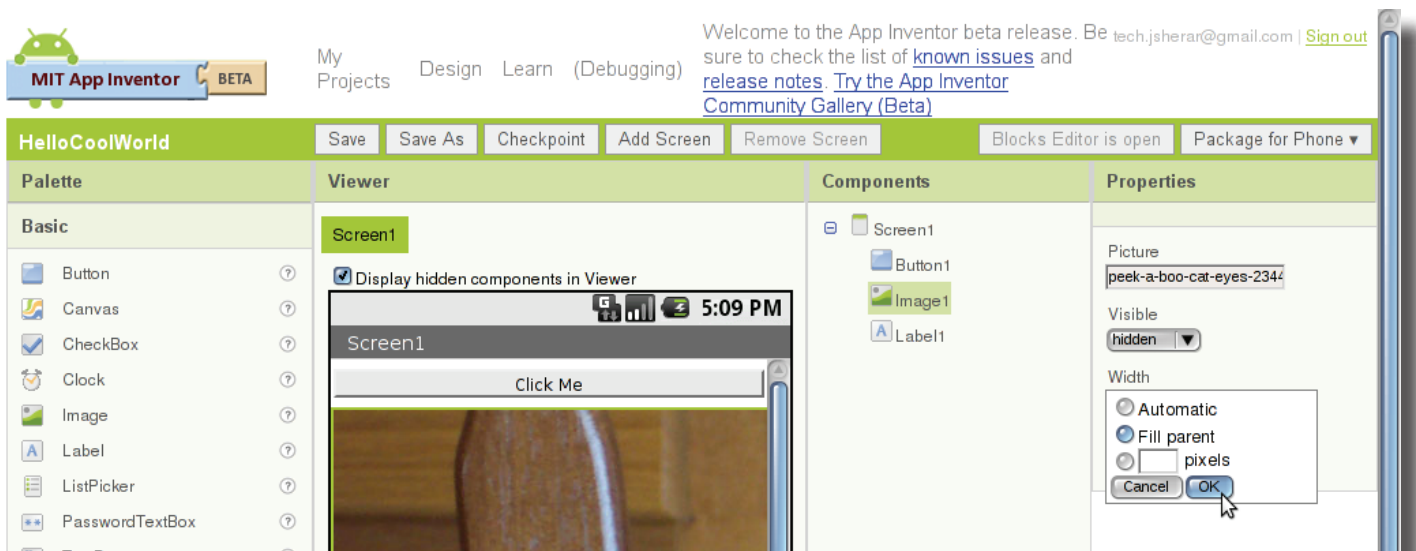


Next, of course, we want something to happen once the user clicks the button. So let's add an Image object and a Label object to Screen1 in the same manner as we did with the button – by drag-and-drop. For Image1, we are able to set the picture by clicking in the text entry area of Image1's Picture property and clicking on Upload new....

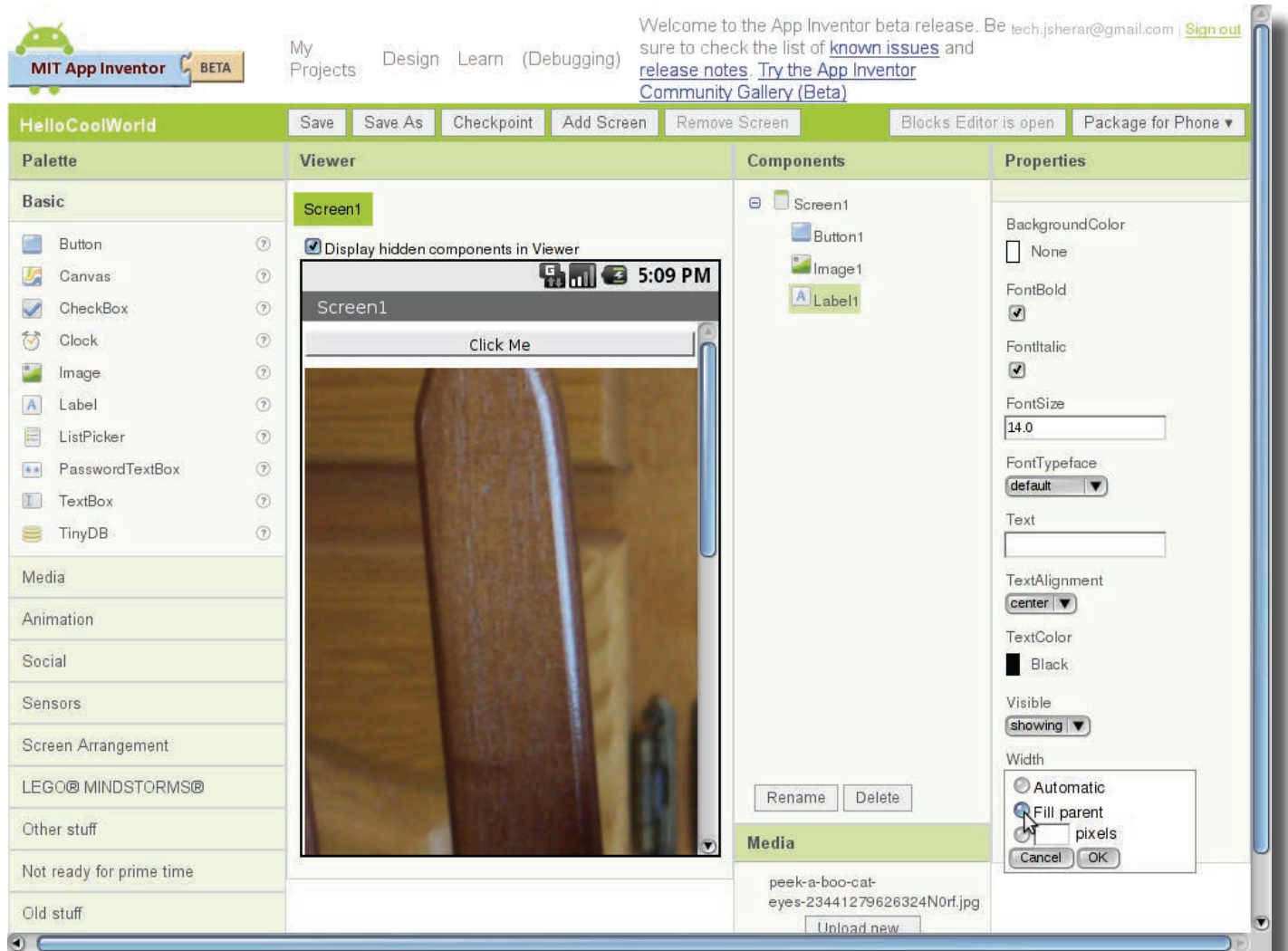


From the dialogue box that opens, click Browse... choose the peek-a-boo-cat-eyes pic retrieved earlier.

Set the Visible property to hidden, and Width to Fill parent.



In the Components panel, select Label1. Set its Text property to be blank, TextAlignment to center and Width to Fill Parent....



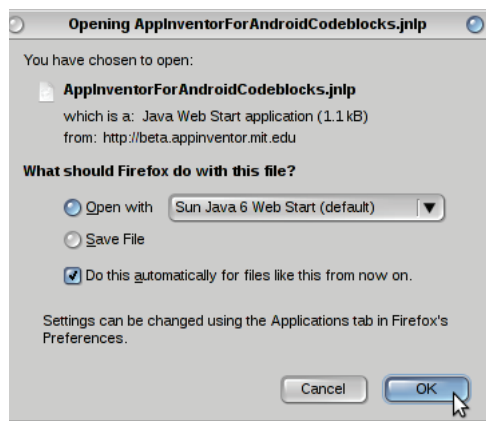
Now, we move on to adding functionality to these objects with the Blocks Editor which is a Java application.

## 3. Coding And Testing

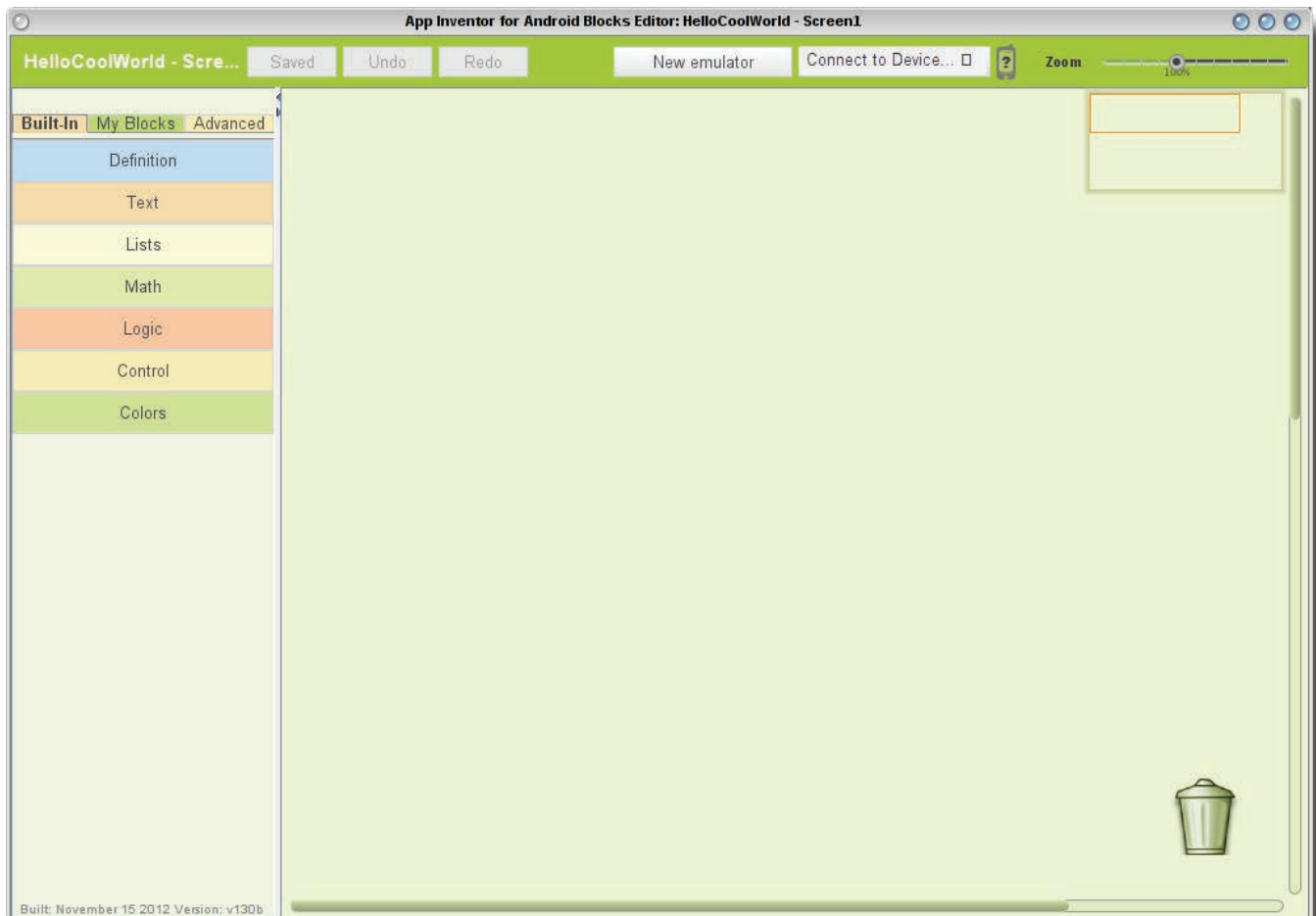
Here is where the App Inventor departs from traditional programming. There is a minimal amount of actual coding required, and it is done in conjunction with live testing with the emulator (or Android device). The beauty of the Blocks Editor is that pretty much all of the “coding” is achieved through the visual process of connecting various programming elements – objects, methods, and properties – like jigsaw puzzle prefabricated pieces of code. This affords the system a measure of self-validation since only certain elements marry up. This is one hallmark of the project that allows it to be highly accessible.

### 3.1 – Accessing The Blocks Editor

Click on the “Open the Blocks Editor” button near the top right corner of the Design window. If a dialogue window opens asking what to do, be sure to run it rather than save it.

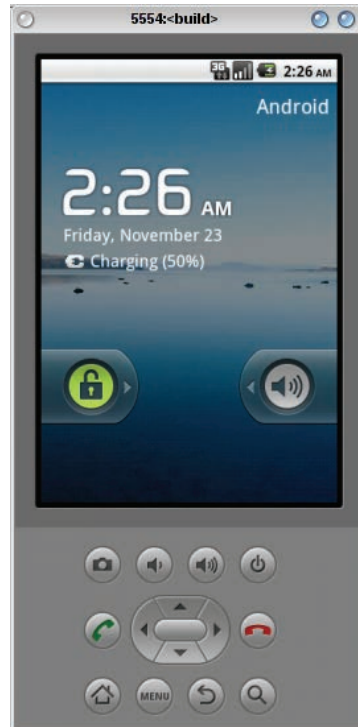


The Java application that launches is the App Inventor for Android Blocks Editor, or Blocks Editor for short. This grants control over the functionality of each of your app’s components.

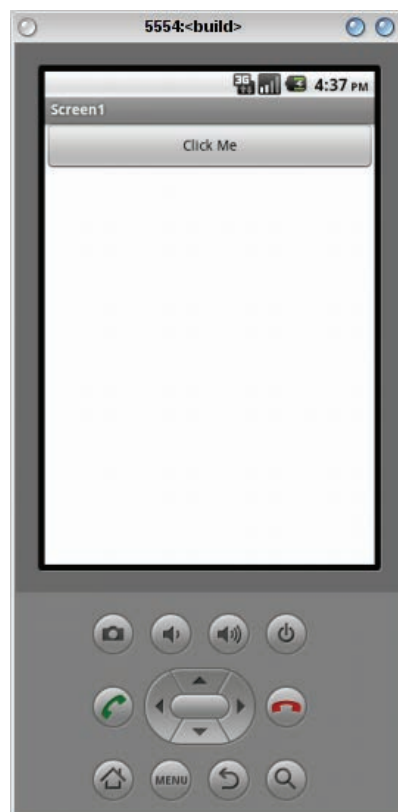


## 3.2 – Accessing The Emulator

Click on the New emulator button at the top of the Blocks Editor window. This will open an emulation of an Android device. This could take a few minutes. The time it takes to load will be proportionate to the speed of your system. A window titled, “Starting the emulator. Please be patient.” will pop up with some useful information that you can peruse while you wait. Click okay when you’re ready. Once fully loaded, you’ll be presented with a window that looks like a mobile phone. Click and drag the green lock to the right to unlock the phone if necessary.

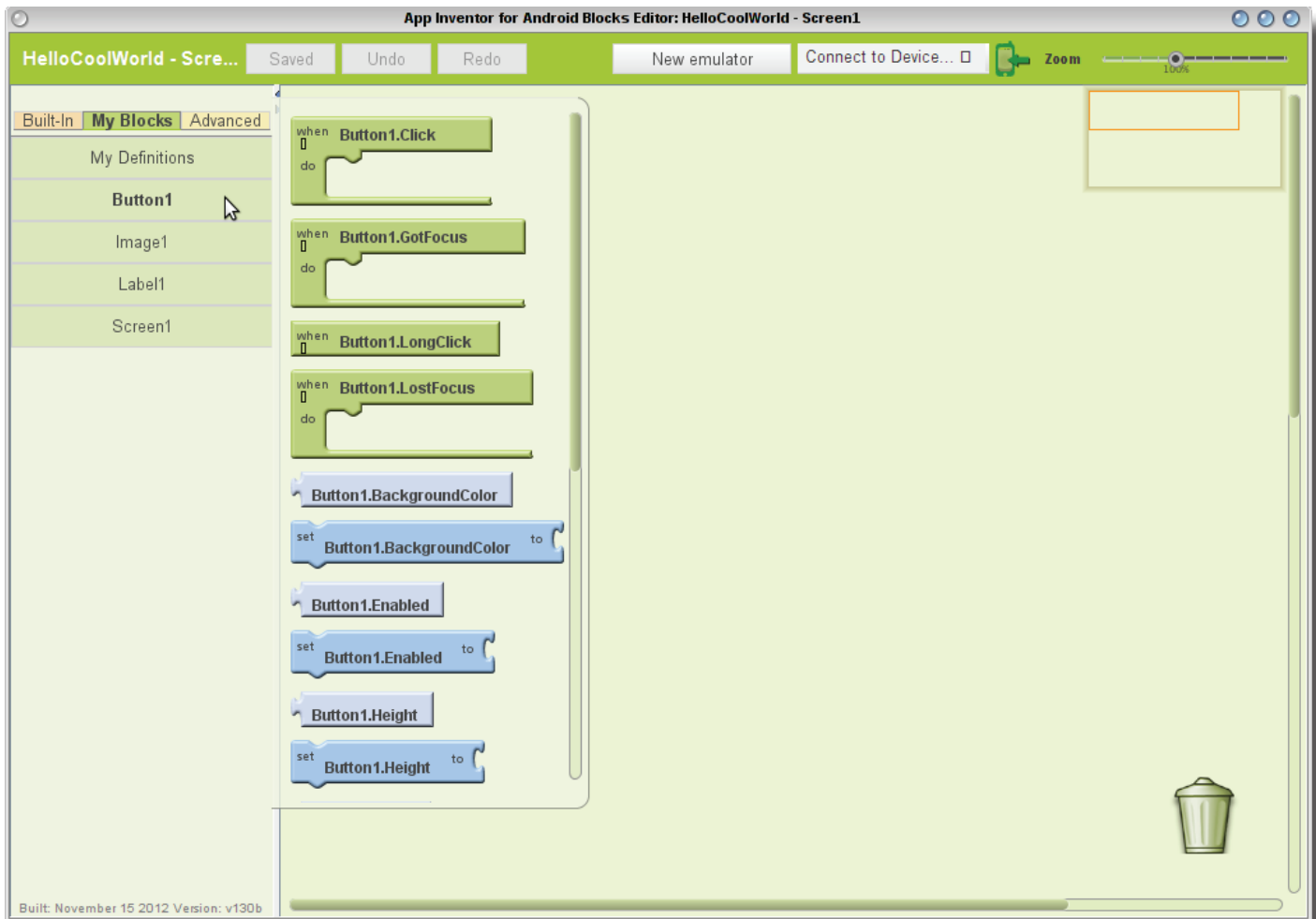


Back in the Blocks Editor, click on the Connect to Device... button and select the emulator from the drop-down list. Again, it could take a while for your computer to establish the connection. One indication that the connection was successful is that now there is a little green icon that looks like a mobile phone at the top of the Blocks Editor. You’ll also be able to see your project in the emulator.

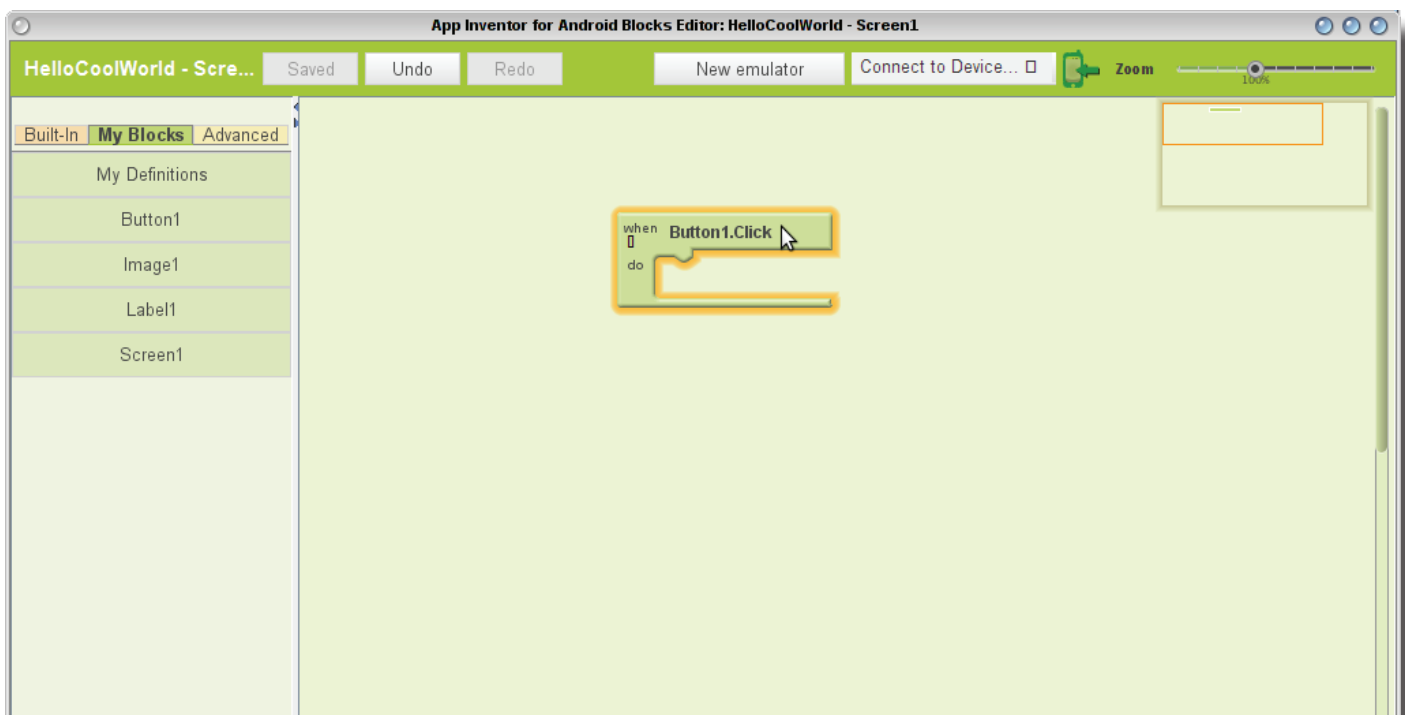


### 3.3 – Coding And testing With The Blocks Editor And The Emulator

Still in the Blocks Editor, select the My Blocks tab and click on Button1. The drawer that slides open, presents you with an object's methods, and properties that can be assembled and manipulated in order to produce the desired functionality.

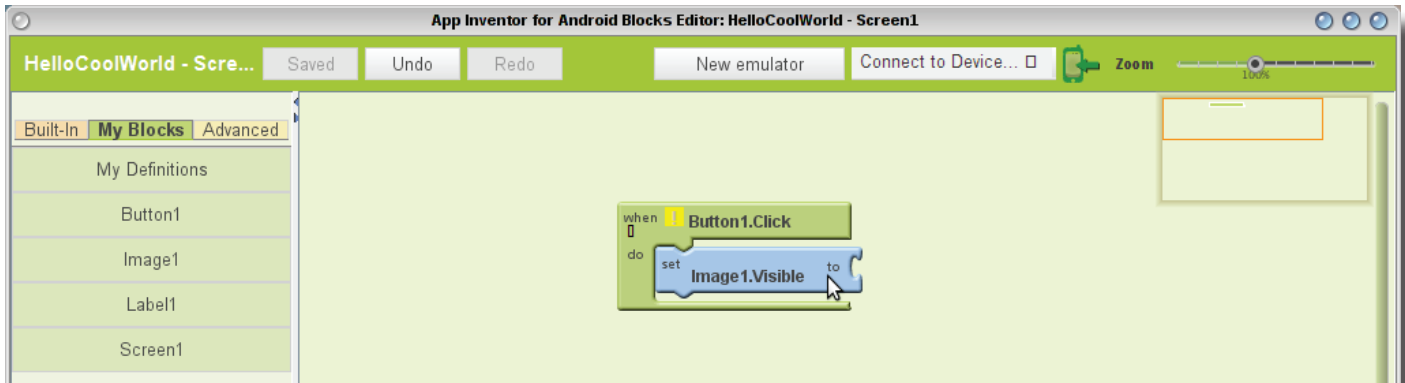


For example, click and drag the Button1.Click method from the drawer onto the work area to the right.

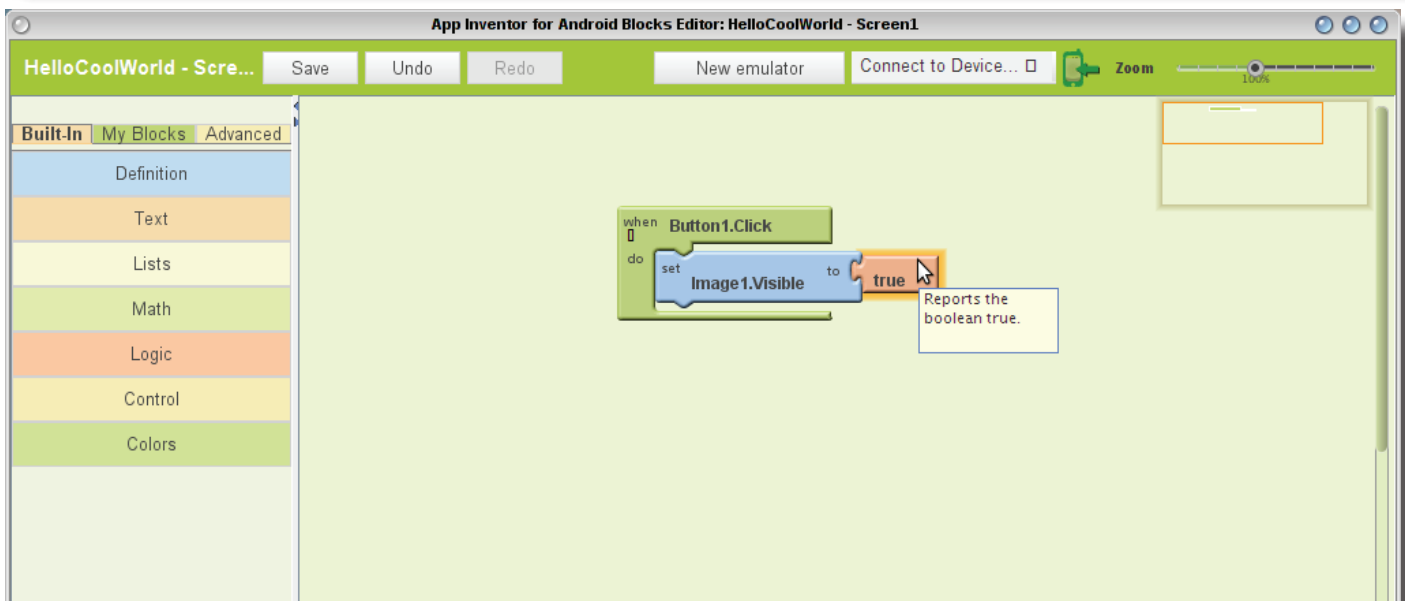
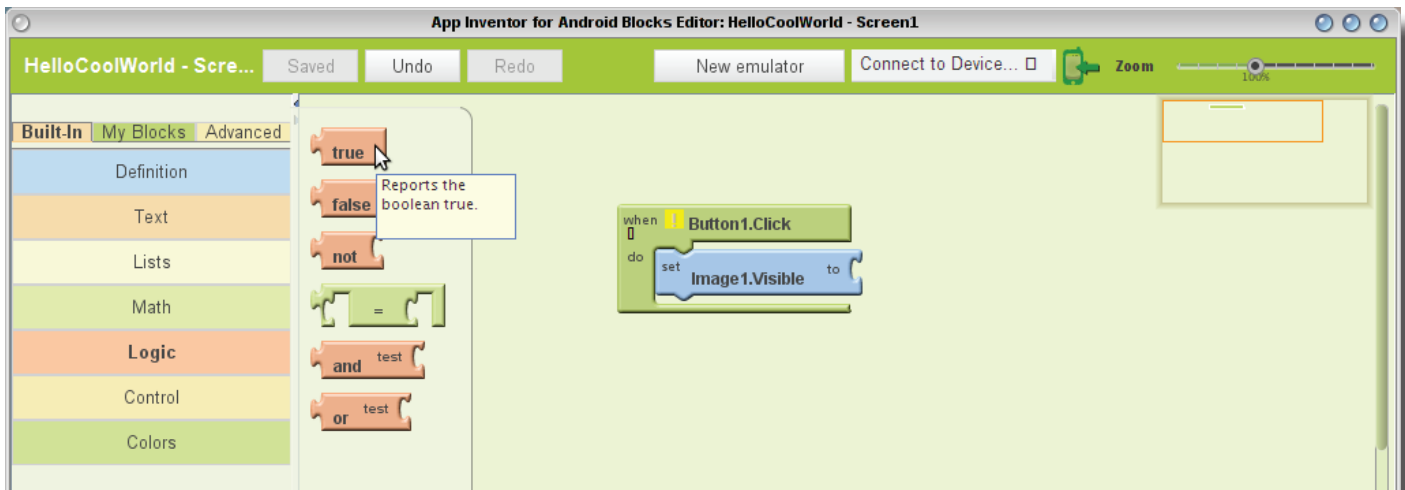


### 3.3.1 – Adding A picture

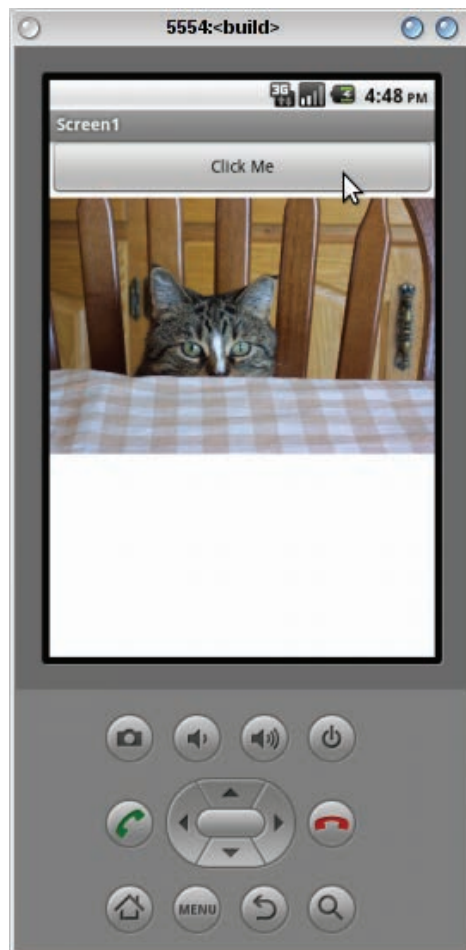
What we would like to happen once the user presses the button in our app, is for the picture of the cute cat to appear, along with the message, “Hello Cool World!” that we set up as a label. So let’s drag-and-drop the set Image1.Visible to method from the Image1 drawer, and fit it inside the Button1.click piece.



You’ll notice that now we have an empty socket that we have to attach something to. So, return to the Built-In tab at the top left of the screen, and from the Logic drawer, select the True piece and attach it to the Image1.Visible socket by drag-and-drop.

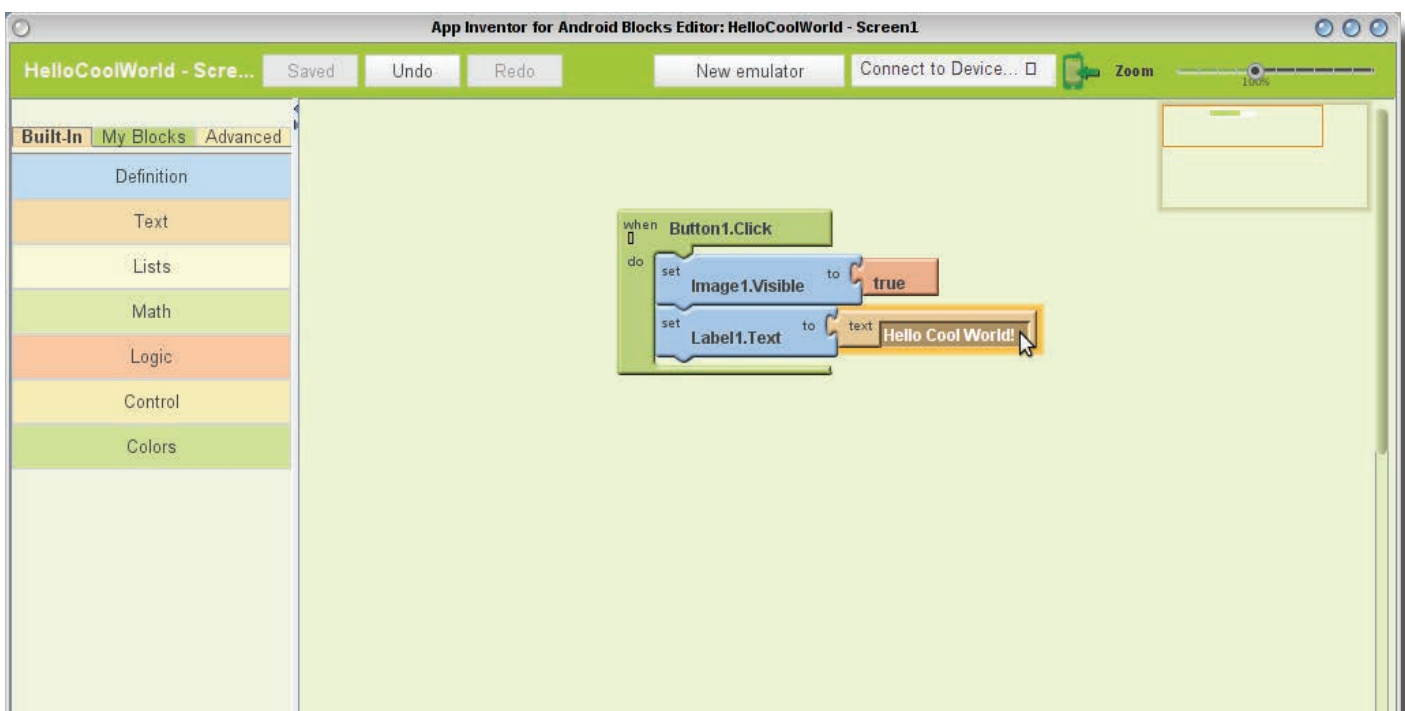


From here, we can test our app by switching to the emulator and clicking on the Click Me button.

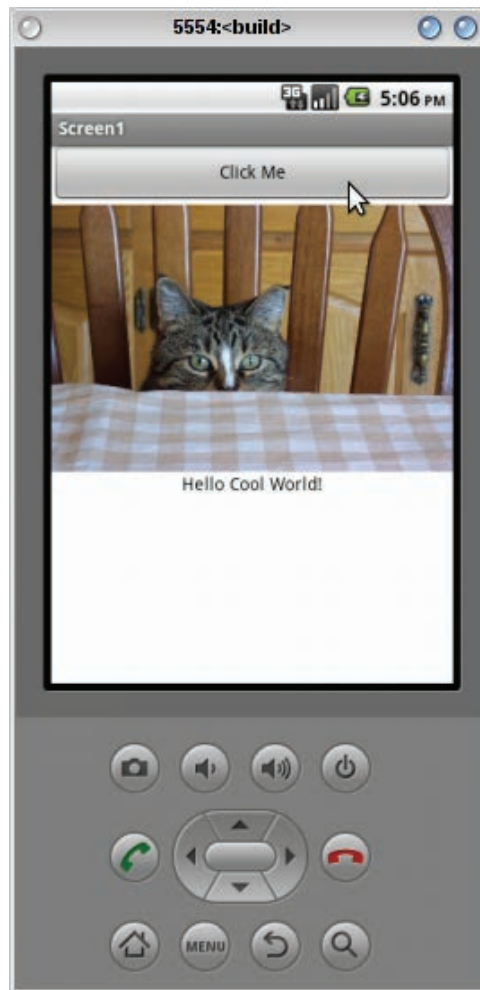


### 3.3.2 – Adding a Message

We also want our message to appear beneath the picture. Returning to the My Blocks tab in the Block Editor, as previously with the image, we'll drag-and-drop the set Label1.Text to method from the drawer and fit it into the Button1.Click object. Returning to the Built-In tab, open the Text drawer and drag-and-drop the Text piece into the Button1.Click object. Click the text that appears in bold face in the piece to set it to read "Hello Cool World!".



Returning to the emulator, we test what we've just done. Clicking on the button now displays our message.



This, of course, is cause for celebration because you have just built your first mobile app. It's simple, but while building it you learned your way around the basics of a system that you can use to eventually build something more complex.

Keep exploring, and enjoy!

## 4. Testing

There's not too much to cover here since the Java based emulator included in the software allowed us to develop and test the app as we went along; a process termed "live testing". As you can tell, this is particularly useful way of doing things because you can immediately see the effects of your changes, therefore making it easy to see your mistakes and realize where you went wrong, as well as your progress and successes.

## 5. Sharing And Marketing

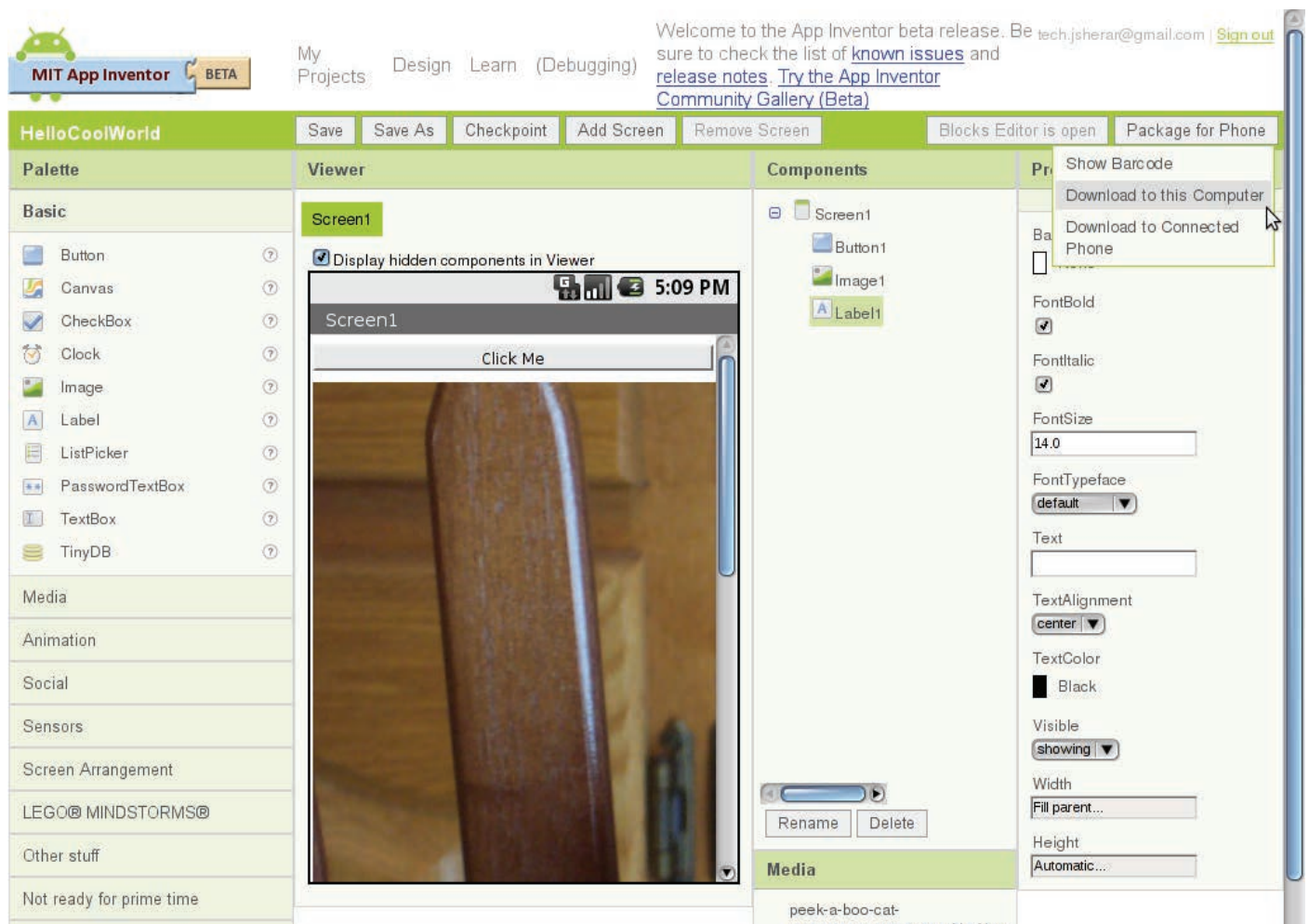
Bringing your app to market involves what is known as **packaging**. What this is just a process whereby your app is assembled into the Android Package format with an .apk extension, that is at once machine readable and easily and widely distributable.

### 5.1 – Versioning

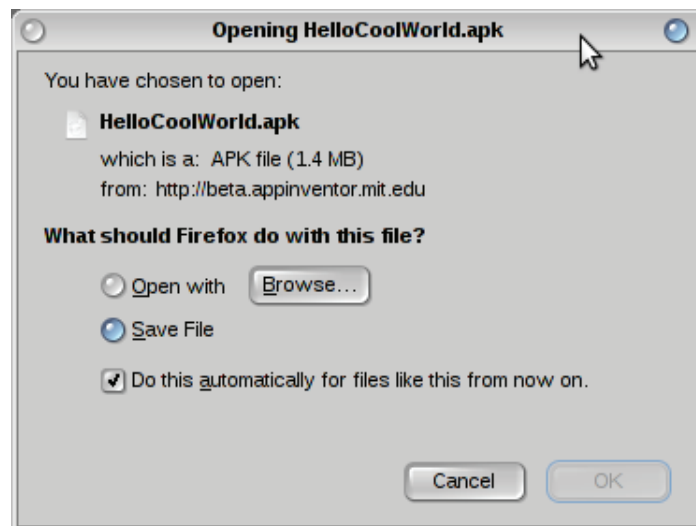
Proper versioning is an important step if you wish to distribute your app commercially. This is done on the Design page via **Screen1's** **VersionCode** and **VersionName** properties. **VersionCode** is an integer value and should be incremented with every new major or minor version of your app. The **VersionName** can be anything you like, however it traditionally includes the name and a decimal number with the whole number representing the major version, and the fractional part representing the number of any minor revision.

### 5.1 – Sharing

In order to share your app with other Android users we first create the .apk file by clicking the **Package for Phone** button in the **Design** window.



Once your package is ready, you'll be prompted to save it to your computer.



Once you have the app, you can distribute it via email or by uploading it to a website as you like. Other people can install it on their phones by opening their email from their device.

## 5.2 – Google Play – The Android Market

In order to place your app on Google Play, first make sure it has a proper version number and is properly named as set out in section 5.1. Once it is properly versioned and you have downloaded your app to your computer as set out in section 5.1, you'll be set to upload it to Google Play... Well, almost.

You must also register as a developer with Google Play and pay a small fee in order to publish. Just follow the instructions from <https://play.google.com/apps/publish/signup>, and you'll be on your way to becoming a bona fide app developer.

## 6. Conclusions

---

Overall, the learning curve is moderate, making introductory computer programming extremely accessible making App Inventor a success even in these early stages of its development. On the other hand some drawbacks at the time of this writing was that you could not include multiple screens for your app, and that it lacked the primitives required to gain full access to the file system. This should be no cause to relegate the system as a toy technology, however, since the platform is still in beta and it has potential for also being an excellent tool, now and into the future, for professional RAD prototyping. For these reasons, App Inventor appears to be poised to become a popular tool not only for non-programmers, but for seasoned developers alike.

### (Endnotes)

1 License: This image is public domain. You may use this picture for any purpose, including commercial. If you do use it, please consider linking back to us. If you are going to redistribute this image online, a hyperlink to this particular page is mandatory. - <http://www.publicdomain-pictures.net/view-image.php?image=7770&picture=peek-a-boo-cats-eyes>





Did you like this PDF Guide? Then why not visit [MakeUseOf.com](http://www.makeuseof.com) for daily posts on cool websites, free software and internet tips?

If you want more great guides like this, why not subscribe to [MakeUseOf](http://www.makeuseof.com) and receive instant access to 50+ PDF Guides like this one covering wide range of topics. Moreover, you will be able to download free Cheat Sheets, Free Giveaways and other cool things.

Home:	<a href="http://www.makeuseof.com">http://www.makeuseof.com</a>
MakeUseOf Directory:	<a href="http://www.makeuseof.com/dir">http://www.makeuseof.com/dir</a>
MakeUseOf Answers:	<a href="http://www.makeuseof.com/answers">http://www.makeuseof.com/answers</a>
Geeky Fun:	<a href="http://www.makeuseof.com/tech-fun">http://www.makeuseof.com/tech-fun</a>
PDF Guides:	<a href="http://www.makeuseof.com/pages/">http://www.makeuseof.com/pages/</a>
Tech Deals:	<a href="http://www.makeuseof.com/pages/hot-tech-deals">http://www.makeuseof.com/pages/hot-tech-deals</a>

Follow [MakeUseOf](http://www.makeuseof.com):

RSS Feed:	<a href="http://feedproxy.google.com/Makeuseof">http://feedproxy.google.com/Makeuseof</a>
Newsletter:	<a href="http://www.makeuseof.com/join">http://www.makeuseof.com/join</a>
Facebook:	<a href="http://www.facebook.com/makeuseof">http://www.facebook.com/makeuseof</a>
Twitter:	<a href="http://www.twitter.com/Makeuseof">http://www.twitter.com/Makeuseof</a>

Think you've got what it takes to write a manual for [MakeUseOf.com](http://www.makeuseof.com)? We're always willing to hear a pitch! Send your ideas to [justinpot@makeuseof.com](mailto:justinpot@makeuseof.com); you might earn up to \$400.

## Download Other MakeUseOf PDF Guides!

**<http://makeuseof.com/pages>**

