App Inventor Beginner Tutorials

THE
MEDIA
LAB

MIT Center for Mobile Learning

Four Simple Tutorials for Getting Started with App Inventor

TalkToMe: Your first App Inventor app

This step-by-step picture tutorial will guide you through making a talking app.

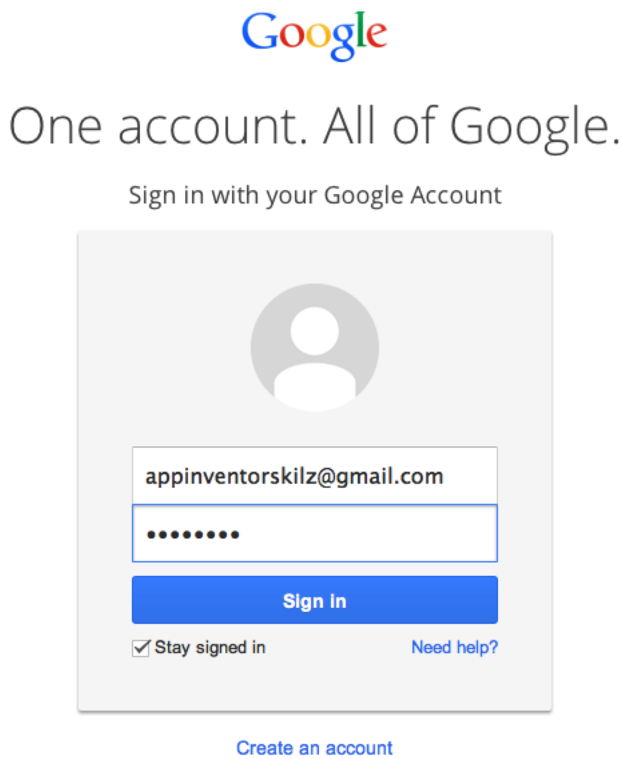
To get started, go to App Inventor on the web.

Go directly to **ai2.appinventor.mit.edu**, or click the orange "Create" button from the App Inventor website.

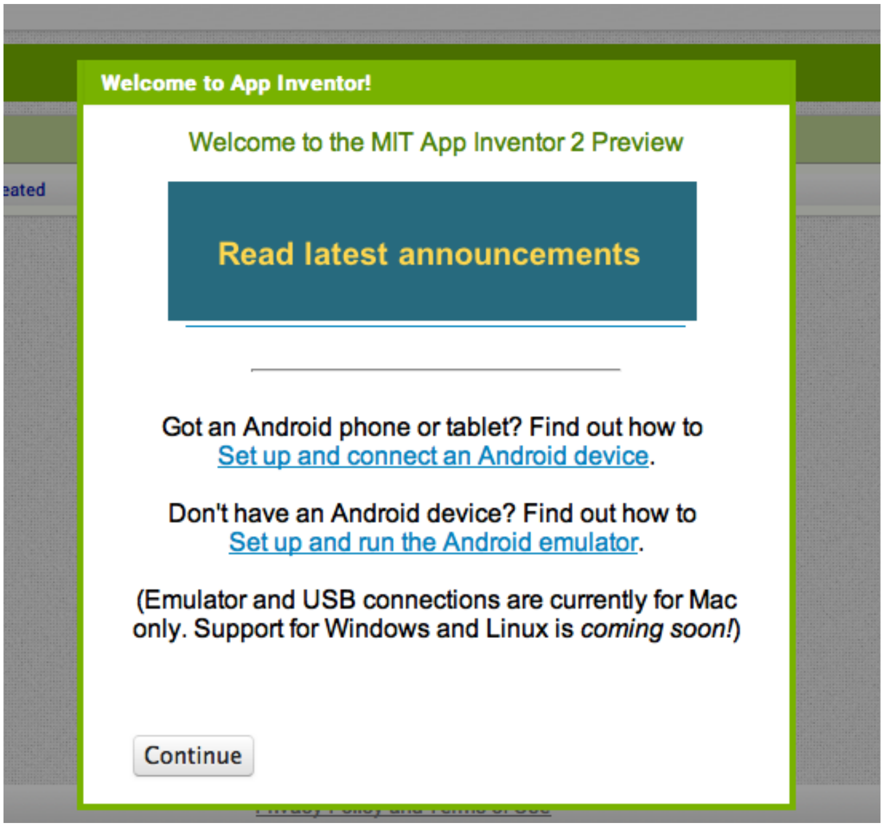
MIT App Inventor
appinventor.mit.edu

Log in to App Inventor with a gmail (or google) user name and password.

Use an existing gmail account or school-based google account to log in to ai2.appinventor.mit.edu
To set up a brand new gmail account, go to accounts.google.com/SignUp

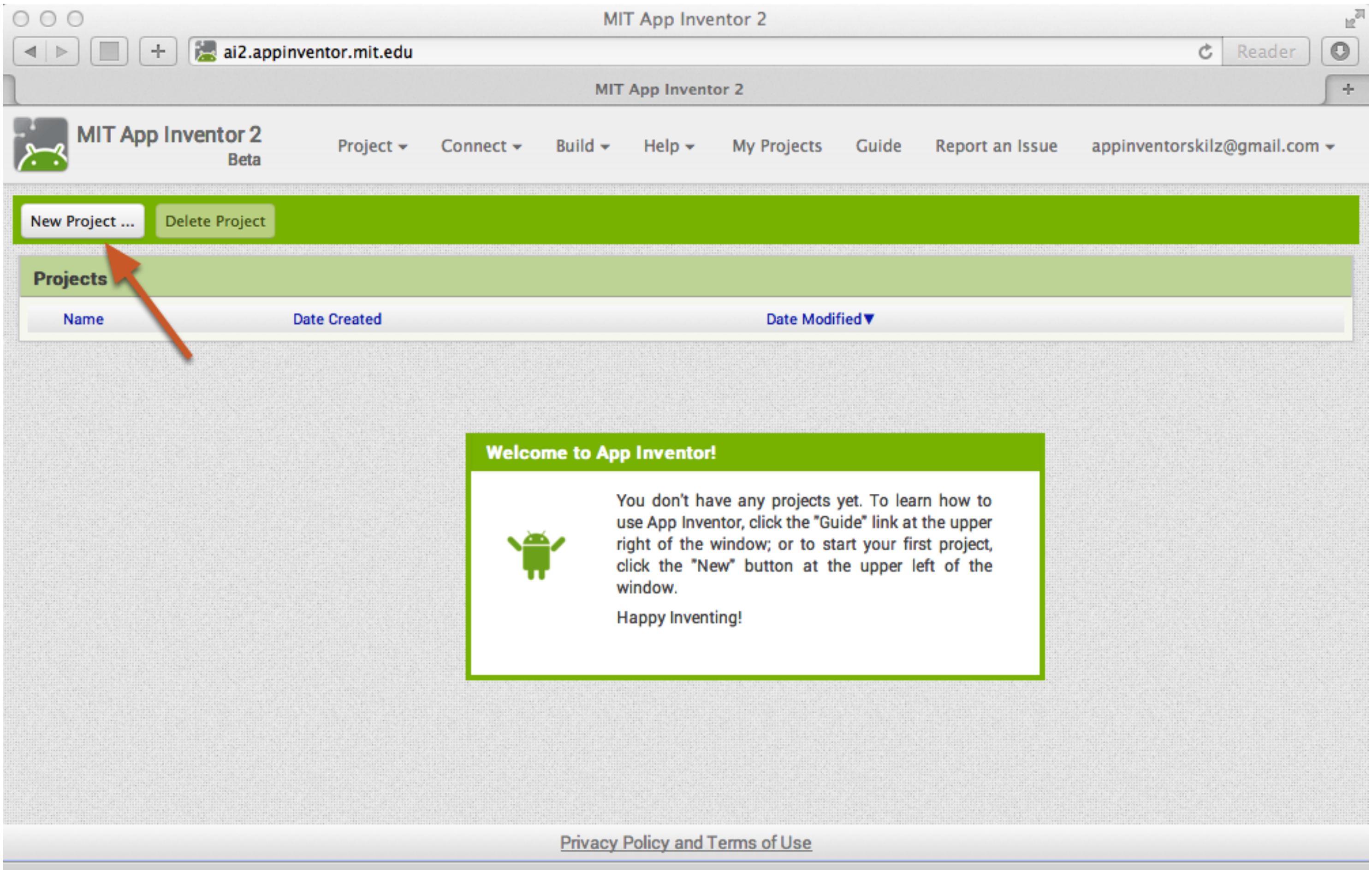


Click "Continue" to dismiss the splash screen.



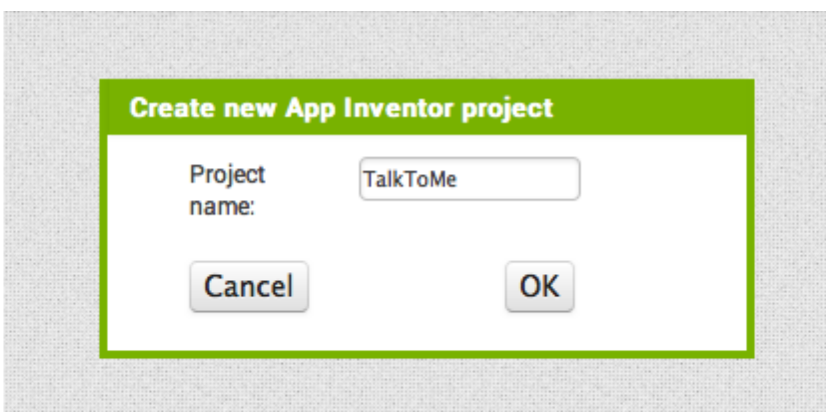
MIT App Inventor
appinventor.mit.edu

Start a new project.



Name the project "TalkToMe" (no spaces!)

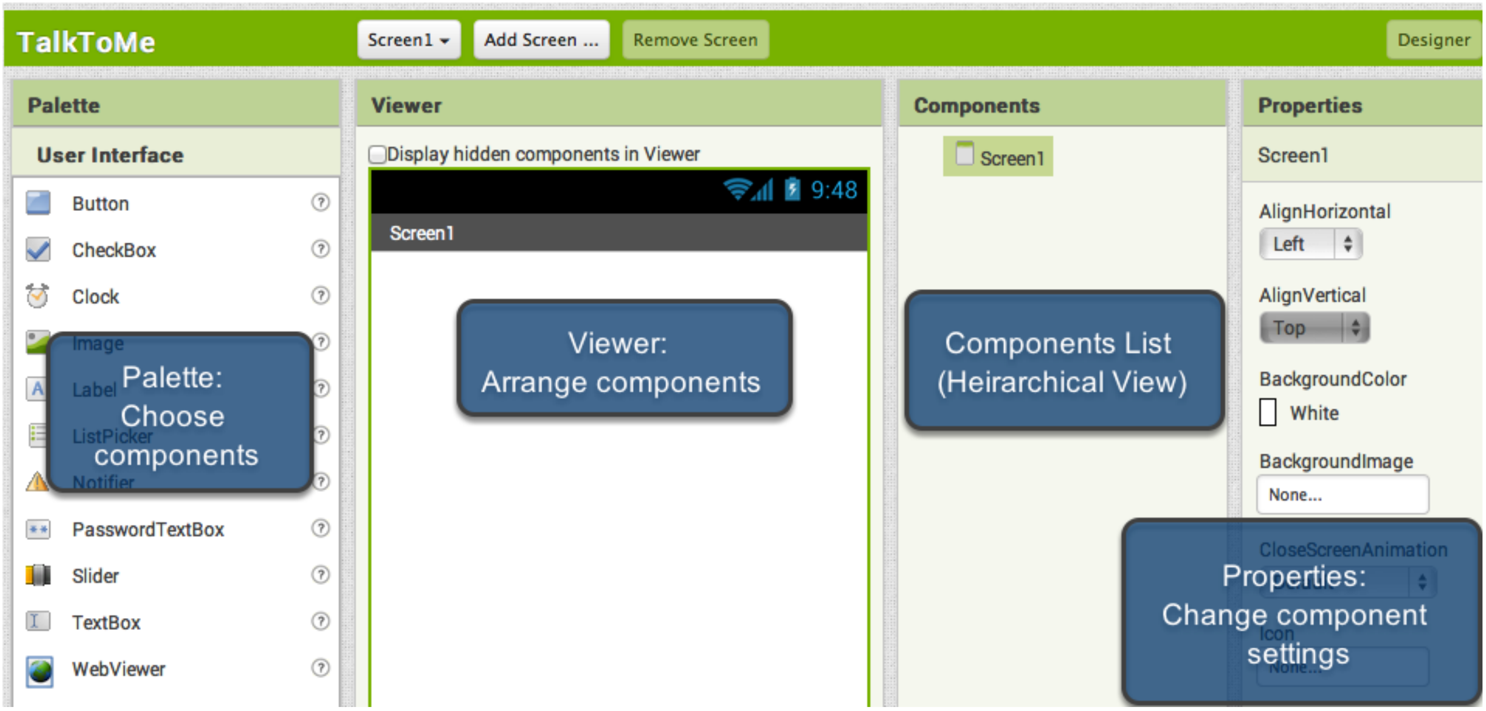
Type in the project name (underscores are allowed, spaces are not) and click OK.



MIT App Inventor
appinventor.mit.edu

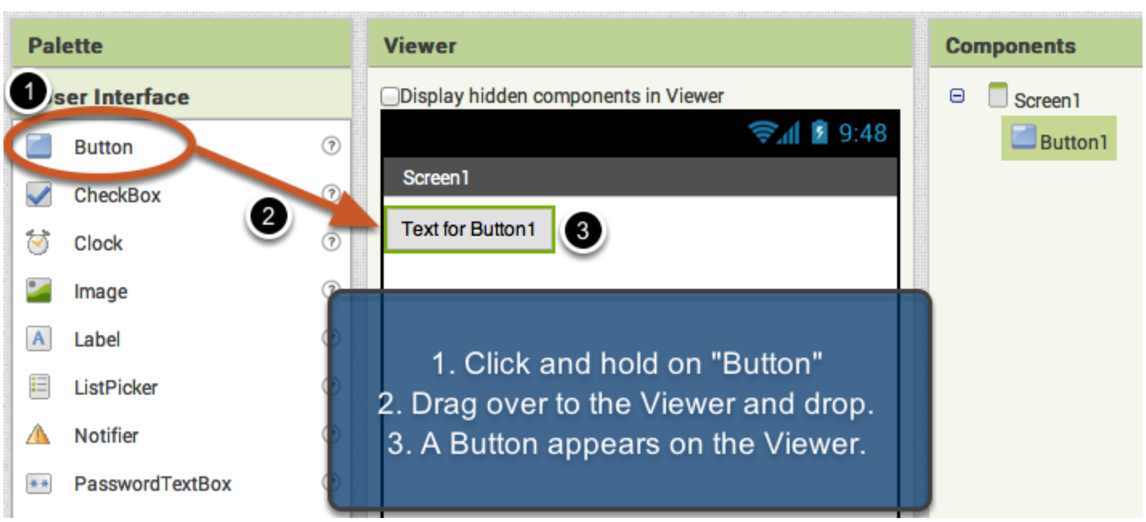
You are now in the Designer, where you lay out the "user interface" of your app.

The Design Window, or simply "Designer" is where you lay out the look and feel of your app, and specify what functionalities it should have. You choose things for the user interface things like Buttons, Images, and Text boxes, and functionalities like Text-to-Speech, Sensors, and GPS.



Add a Button

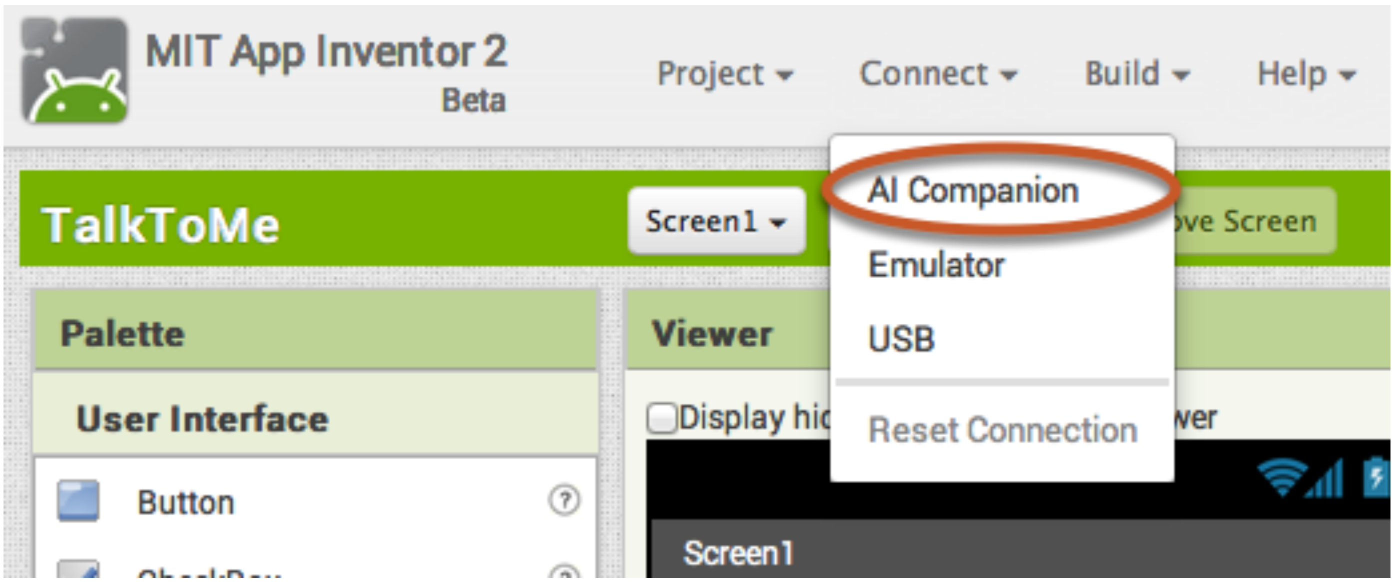
Our project needs a button. ***Click and hold*** on the word "Button" in the palette. ***Drag*** your mouse over to the Viewer. ***Drop*** the button and a new button will appear on the Viewer.



MIT App Inventor
appinventor.mit.edu

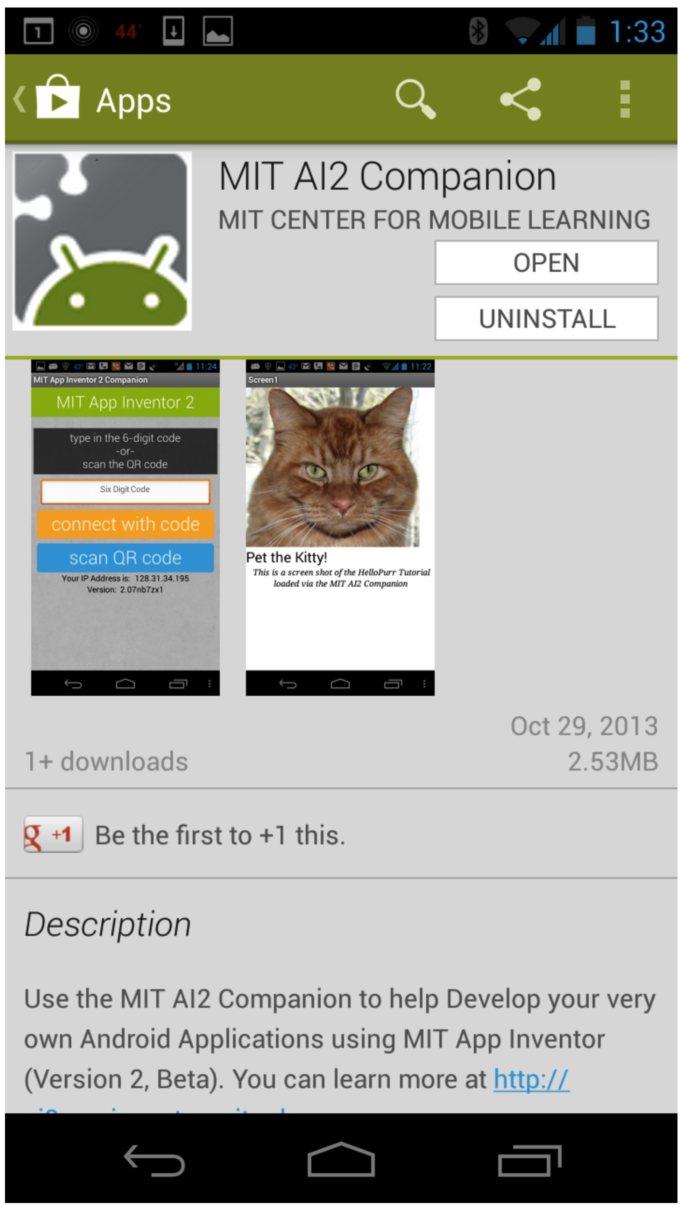
Connect App Inventor to your phone for live testing

One of the neatest things about App Inventor is that you can see and test your app while you're building it, on a connected device. If you have an **Android phone or tablet, follow the steps below.** *If you do not have a device,* then follow the instructions for setting up the on-screen emulator (opens a new page) and then come back to this tutorial once you've gotten the emulator connected to App Inventor.



Get the MIT AI2 Companion from the Play Store and install it on your phone or tablet.

The preferred method for getting the AI2 Companion App is to **download the app from the Play Store by searching for "MIT AI2 Companion".**

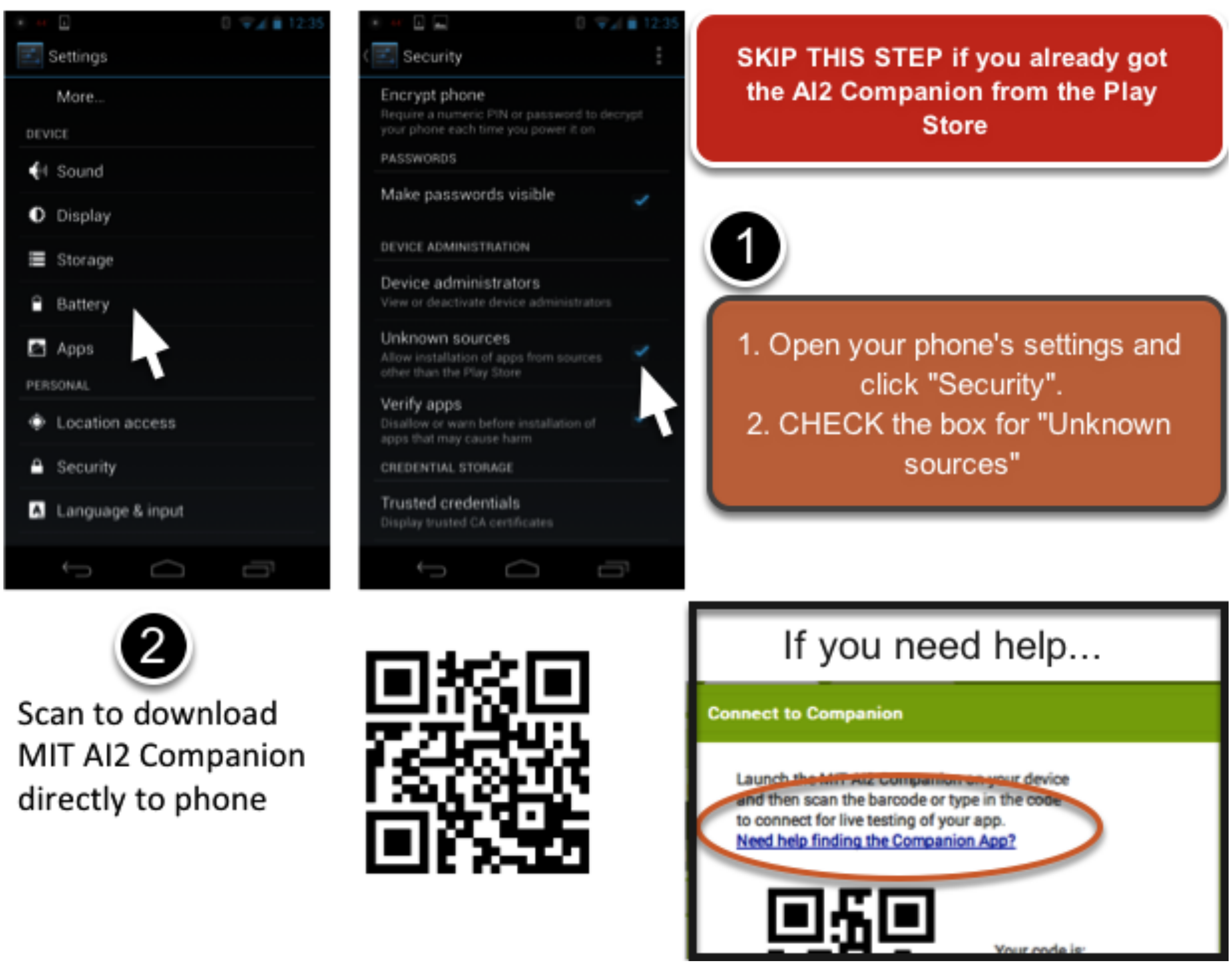


To download the AI2 Companion App to your device directly (SKIP THIS STEP IF YOU already got the app from Play Store)

If for some reason you can not connect to the Google Play store, you can download the AI2 Companion as described here.

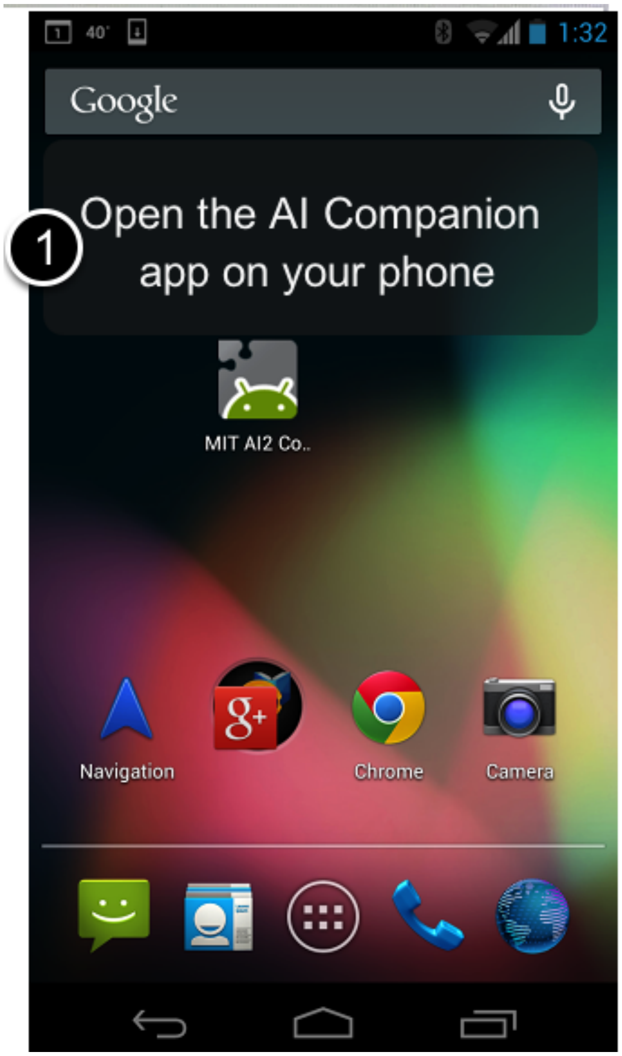
First, you will need to go into your phone's settings (#1), choose "Security", then scroll down to allow "Unknown Sources", which allows apps that are not from the Play Store to be installed on the phone.

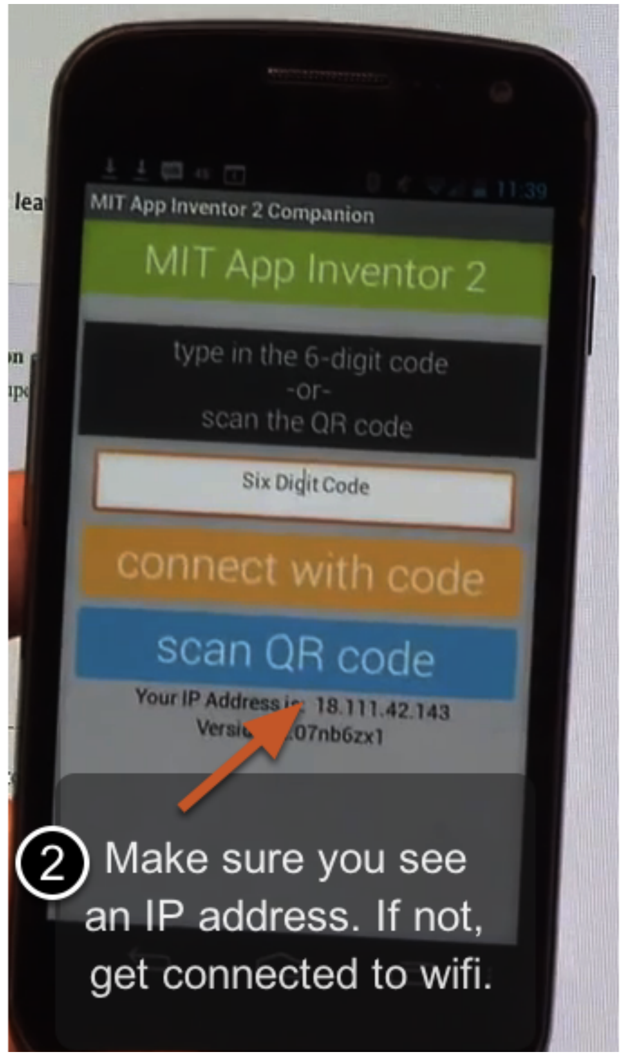
Second, do one of the following:
A) **Scan the QR code above (#2)**
or
B) Click the "Need help finding..." link and you'll be taken to the download page. From there you can download the MITAI2Companion.apk file to your computer and then move it over to your device to install it.



Start the AICompanion on your device

On your phone or tablet, click the icon for the MIT AI Companion to start the app. NOTE: Your **phone and computer must both be on the same wireless network.** Make sure your phone's wifi is on and that you are connected to the local wireless network. If you can not connect over wifi, go to the Setup Instructions on the App Inventor Website to find out how to connect with a USB cable.





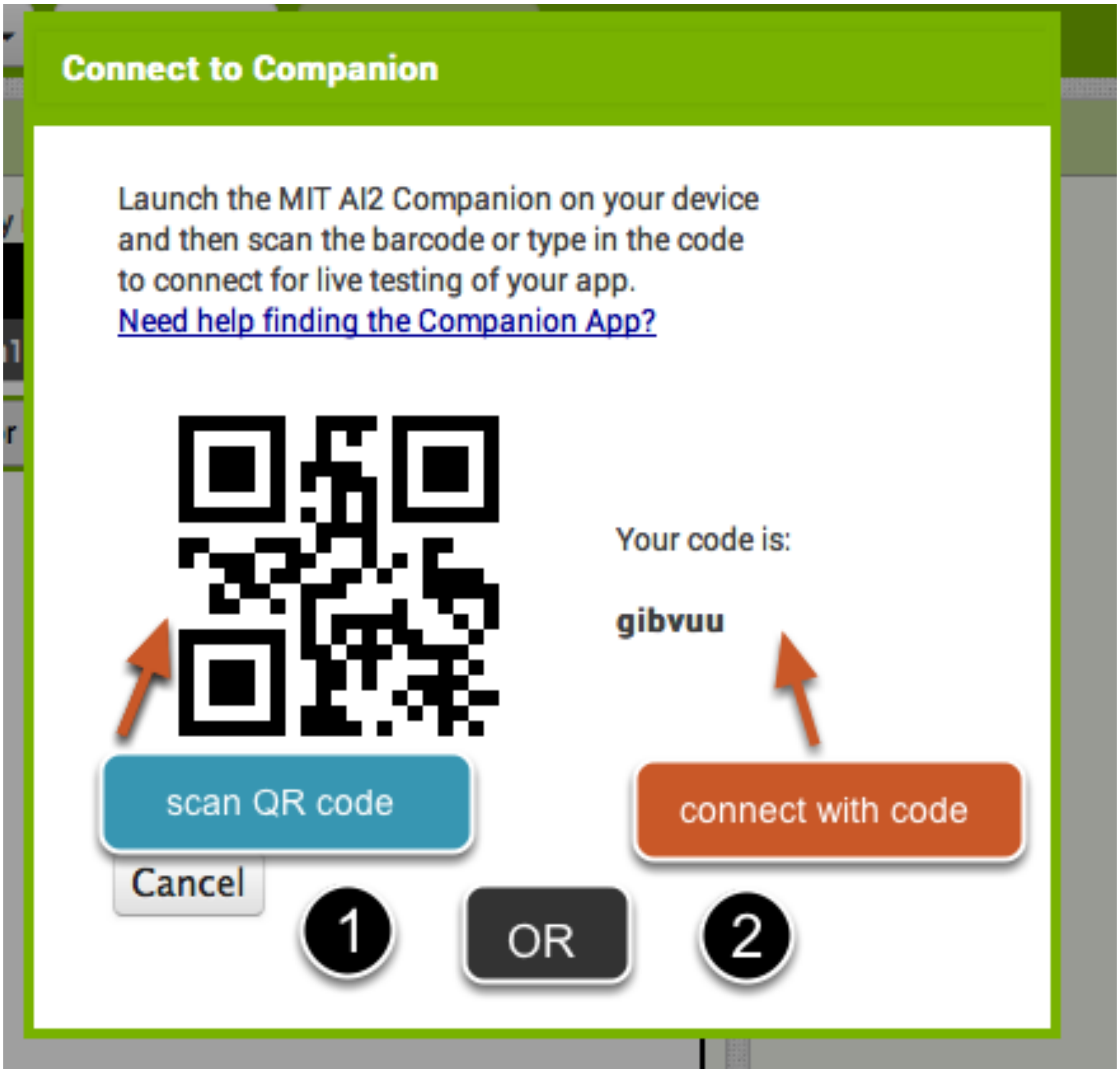
Get the Connection Code from App Inventor and scan or type it into your Companion app

On the Connect menu, choose "AI Companion". You can connect by:

1 - Scanning the QR code by clicking "Scan QR code" (#1).

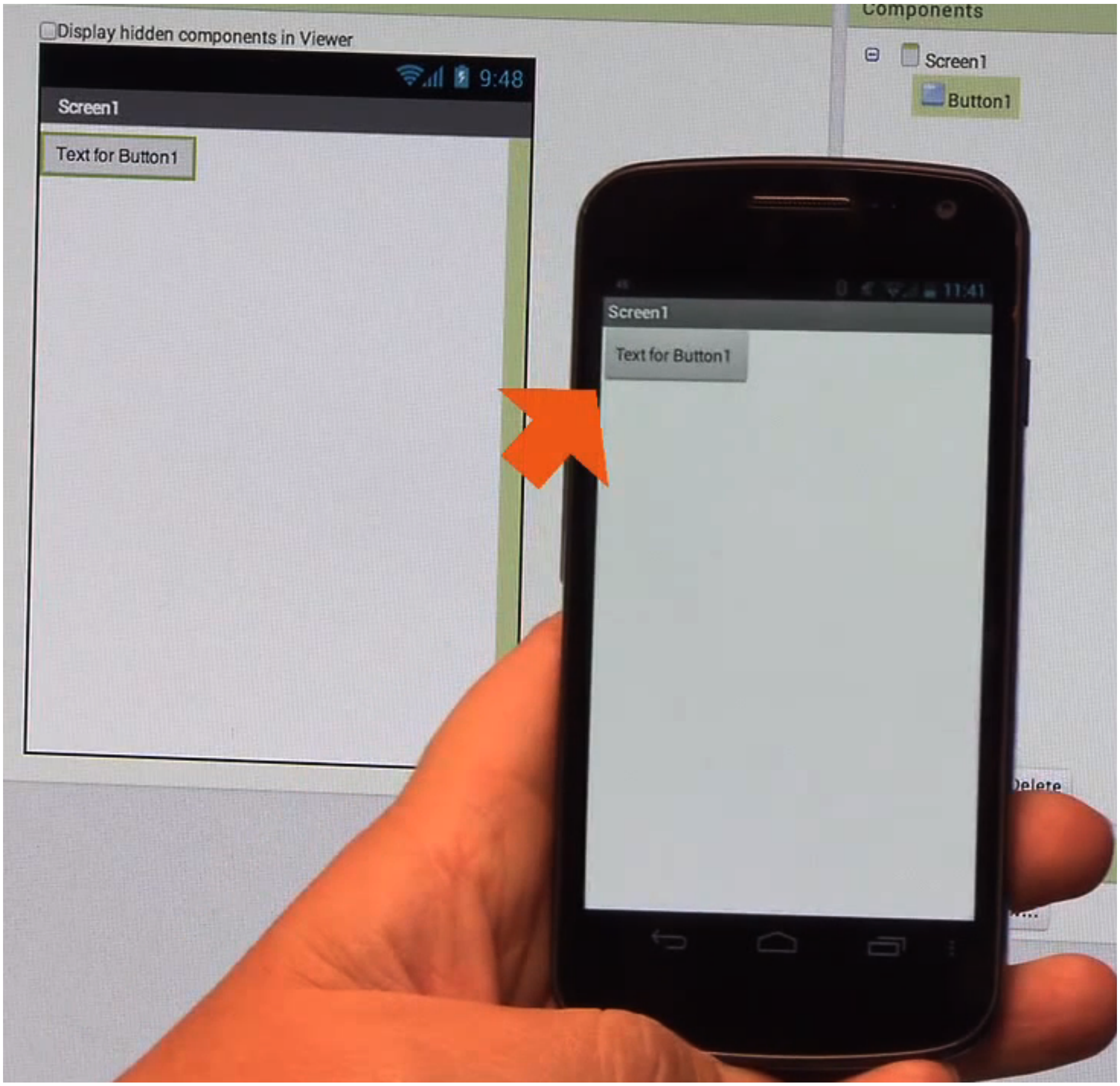
or

2 - Typing the code into the text window and click "Connect with code" (#2).



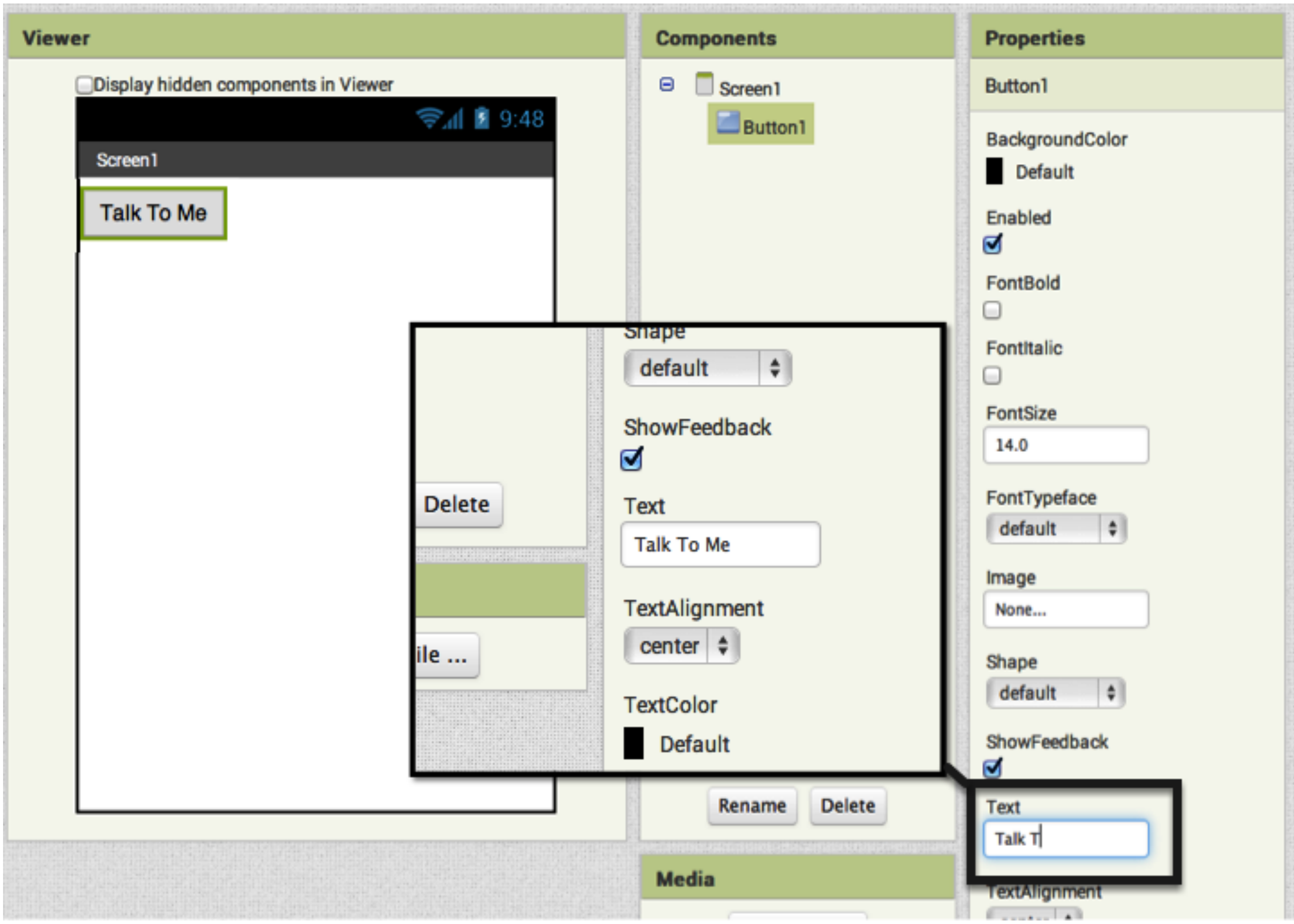
See your app on the connected device

You will know that your connection is successful when you see your app on the connected device. So far our app only has a button, so that is what you will see. As you add more to the project, you will see your app change on your phone.



Change the Text on the Button

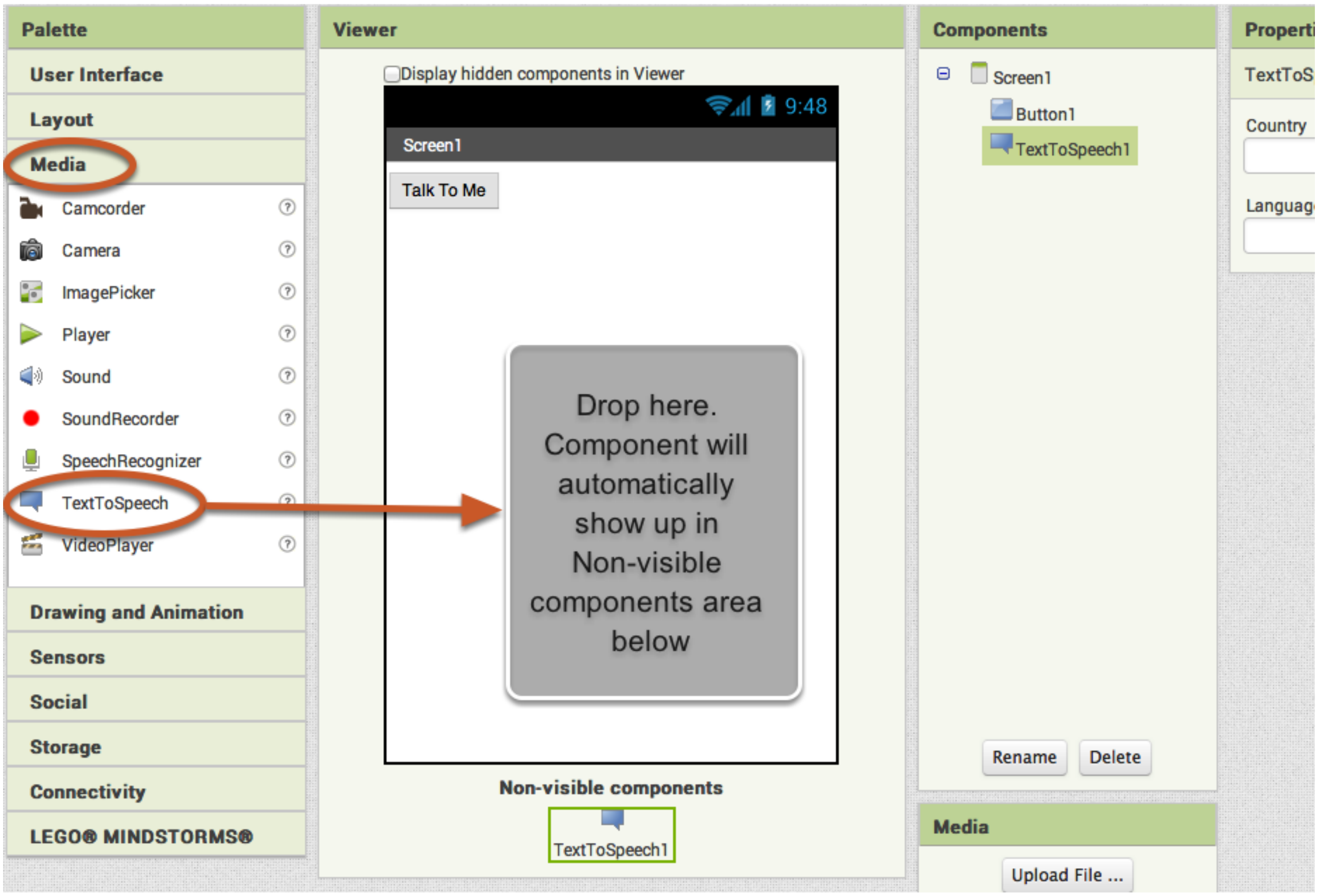
On the properties pane, change the text for the Button. Select the text "Text for Button 1", delete it and type in "Talk To Me". Notice that the text on your app's button changes right away.



MIT App Inventor
appinventor.mit.edu

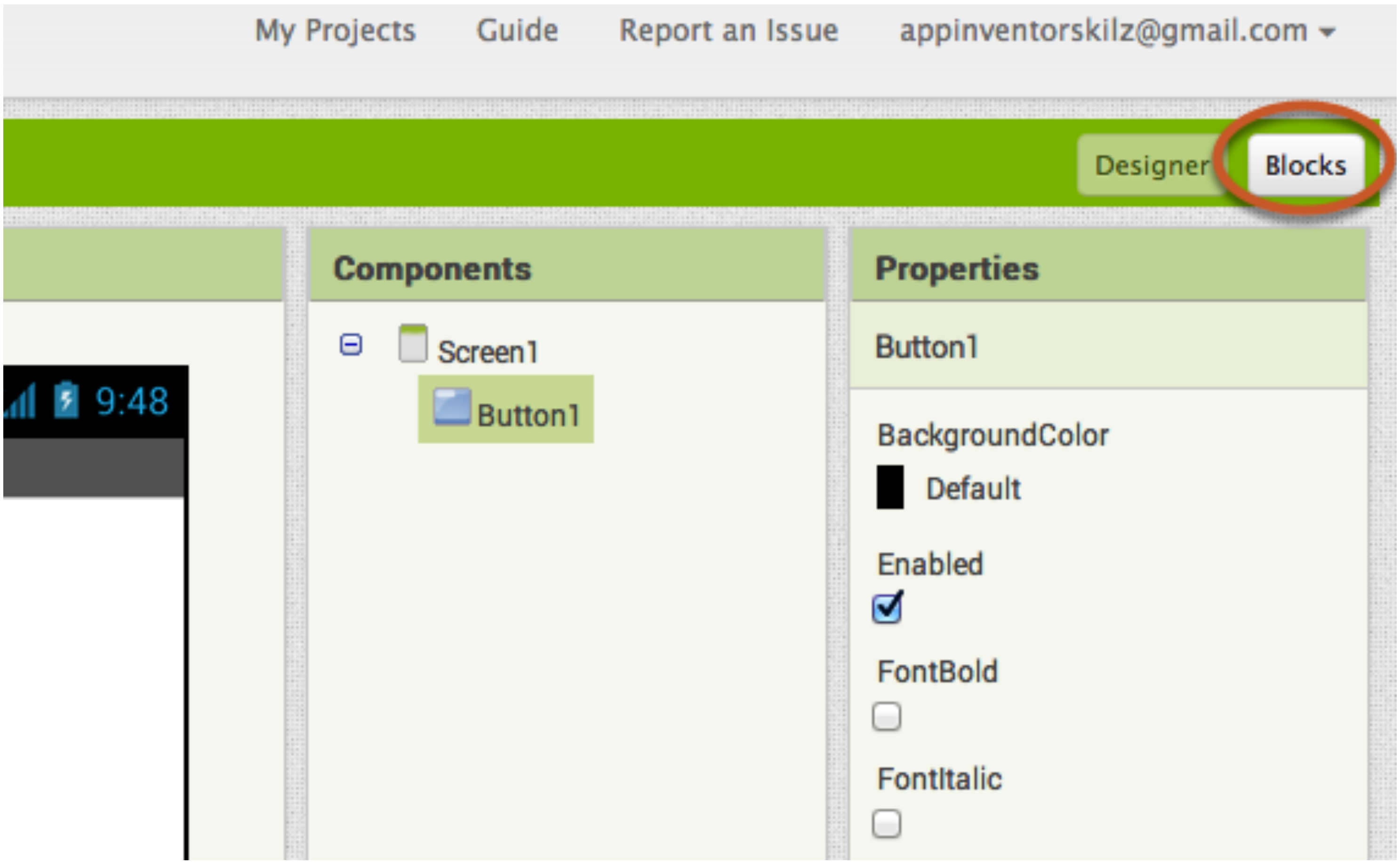
Add a Text-to-Speech component to your app

Go to the Media drawer and drag out a TextToSpeech component. Drop it onto the Viewer. Notice that it drops down under "Non-visible components" because it is not something that will show up on the app's user interface. It's more like a tool that is available to the app.



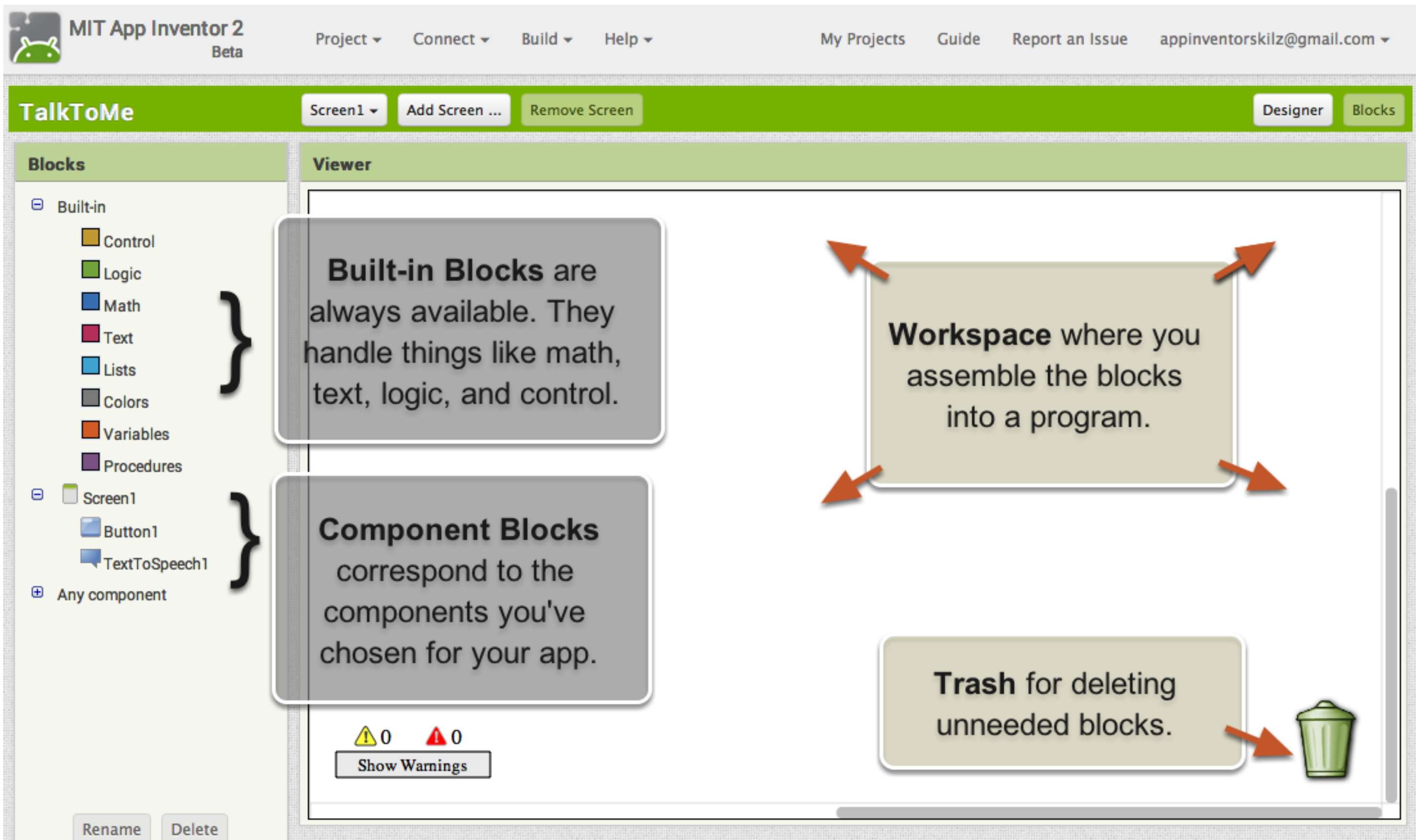
Switch over to the Blocks Editor

It's time to tell your app what to do! Click "Blocks" to move over to the Blocks Editor. Think of the Designer and Blocks buttons like tabs -- you use them to move back and forth between the two areas of App Inventor.



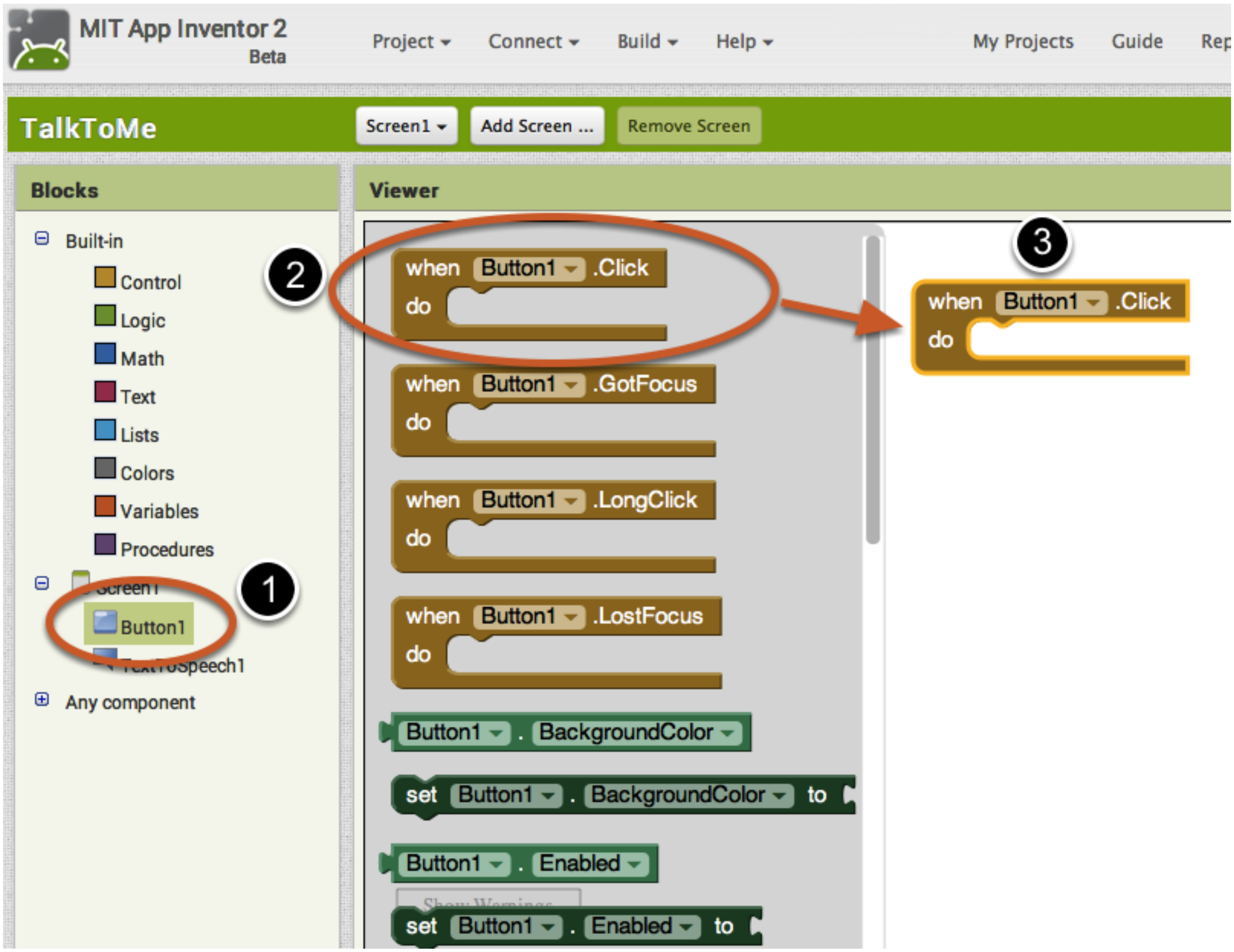
The Blocks Editor

The Blocks Editor is where you program the behavior of your app. There are Built-in blocks that handle things like math, logic, and text. Below that are the blocks that go with each of the components in your app. *In order to get the blocks for a certain component to show up in the Blocks Editor, you first have to add that component to your app through the Designer.*



Make a button click event

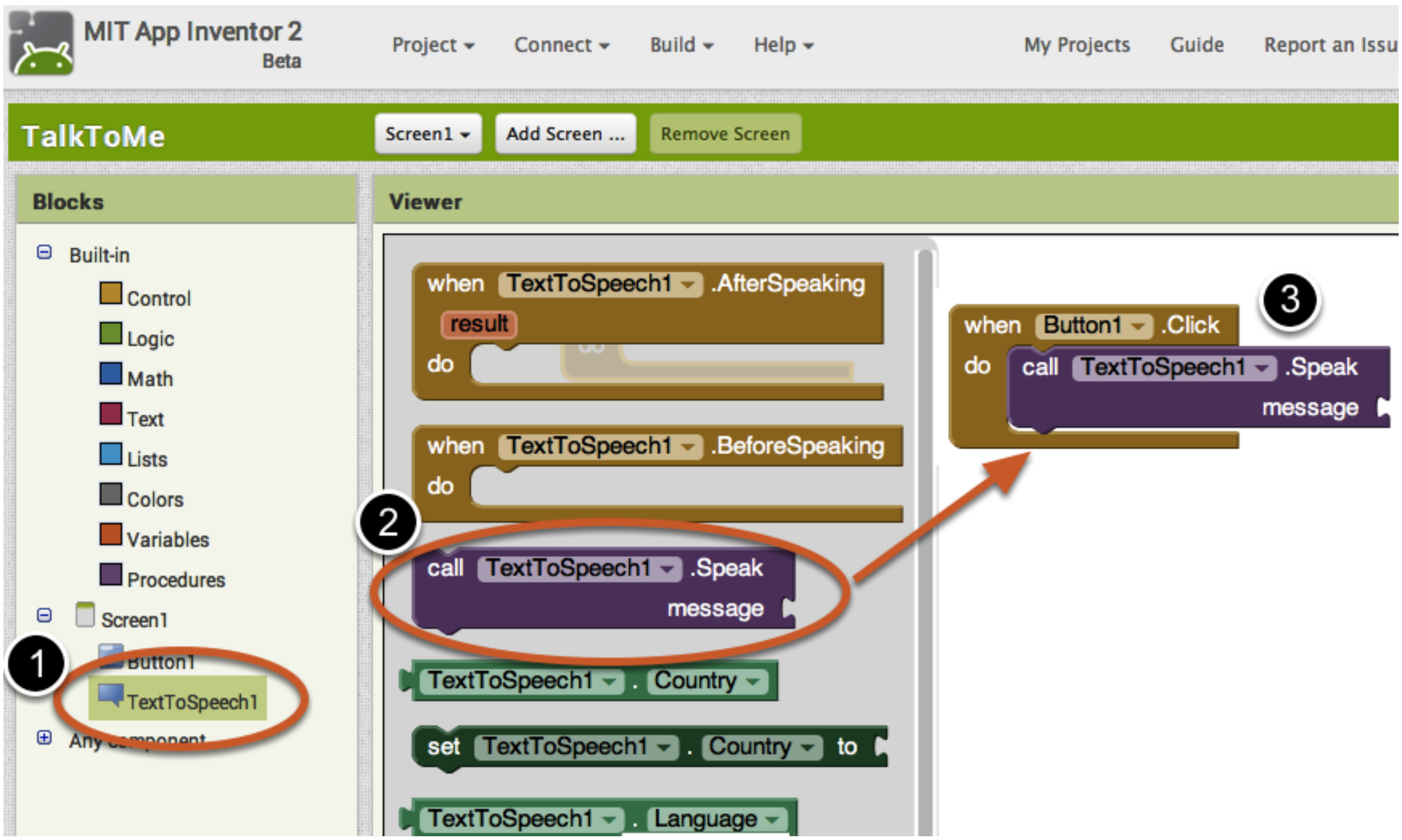
Click on the Button1 drawer. Click and hold the ***when Button1.Click do*** block. Drag it over to the workspace and drop it there. This is the block that will handle what happens when the button on your app is clicked. It is called an "Event Handler".



MIT App Inventor
appinventor.mit.edu

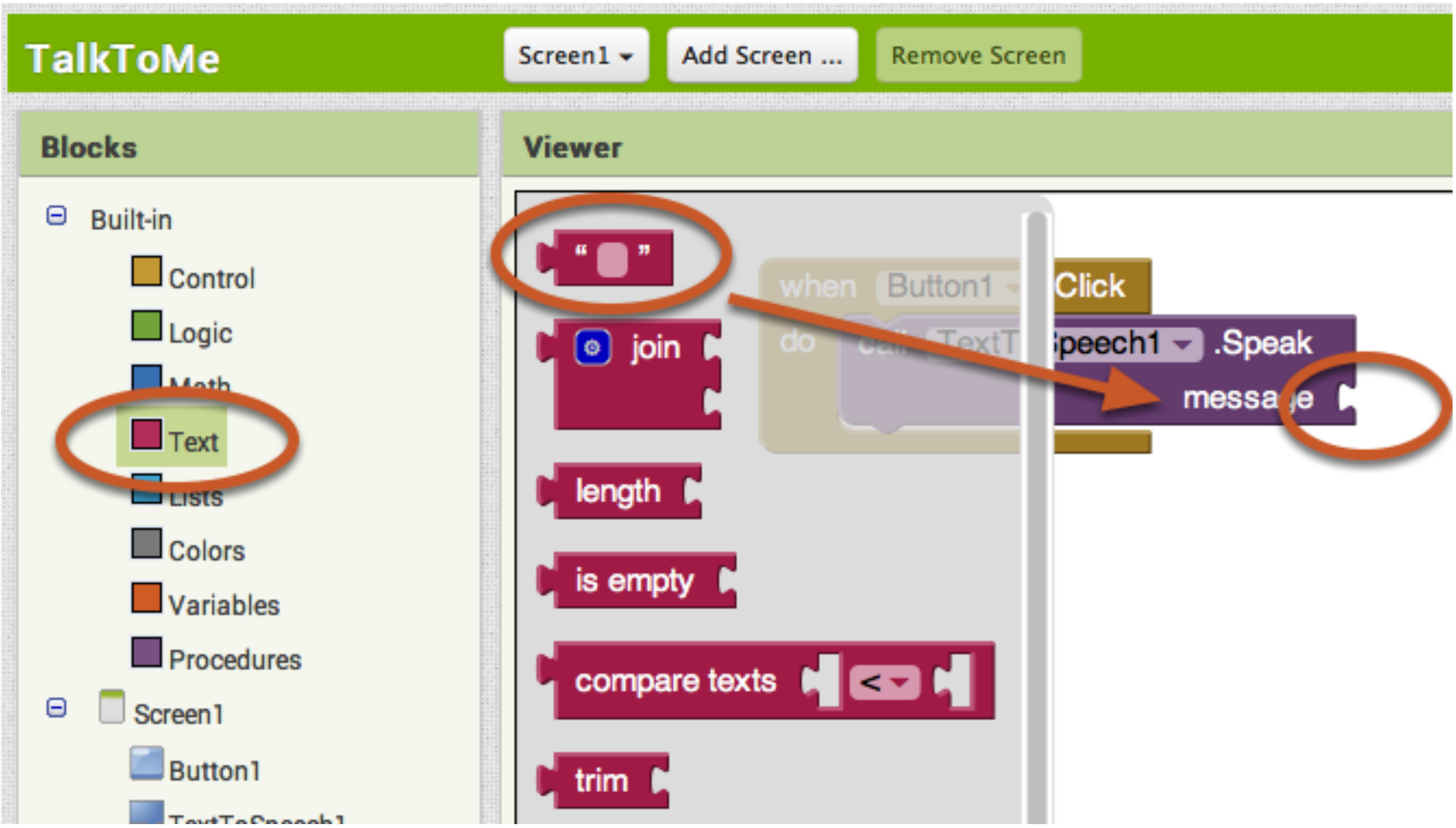
Program the TextToSpeech action

Click on the TextToSpeech drawer. Click and hold the ***call TextToSpeech1.Speak*** block. Drag it over to the workspace and drop it there. This is the block that will make the phone speak. Because it is inside the Button.Click, it will run when the button on your app is clicked.



Fill in the message socket on TextToSpeech.Speak Block

Almost done! Now you just need to tell the TextToSpeech.Speak block what to say. To do that, click on the Text drawer, drag out a ***text*** block and plug it into the socket labeled "message".

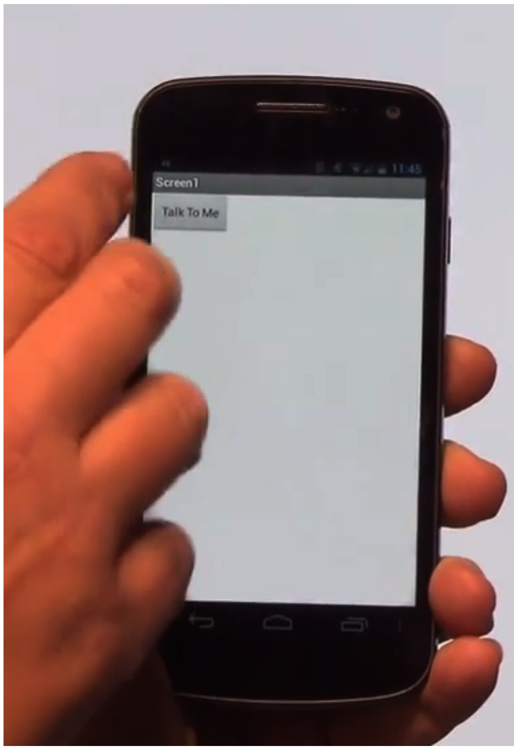


Specify what the app should say when the button is clicked

Click on the text block and type in "Congratulations! You've made your first app." (Feel free to use any phrase you like, this is just a suggestion.)

Now test it out!

Go to your connected device and click the button. Make sure your volume is up! You should hear the phone speak the phrase out loud. (This works even with the emulator.)



Great job!

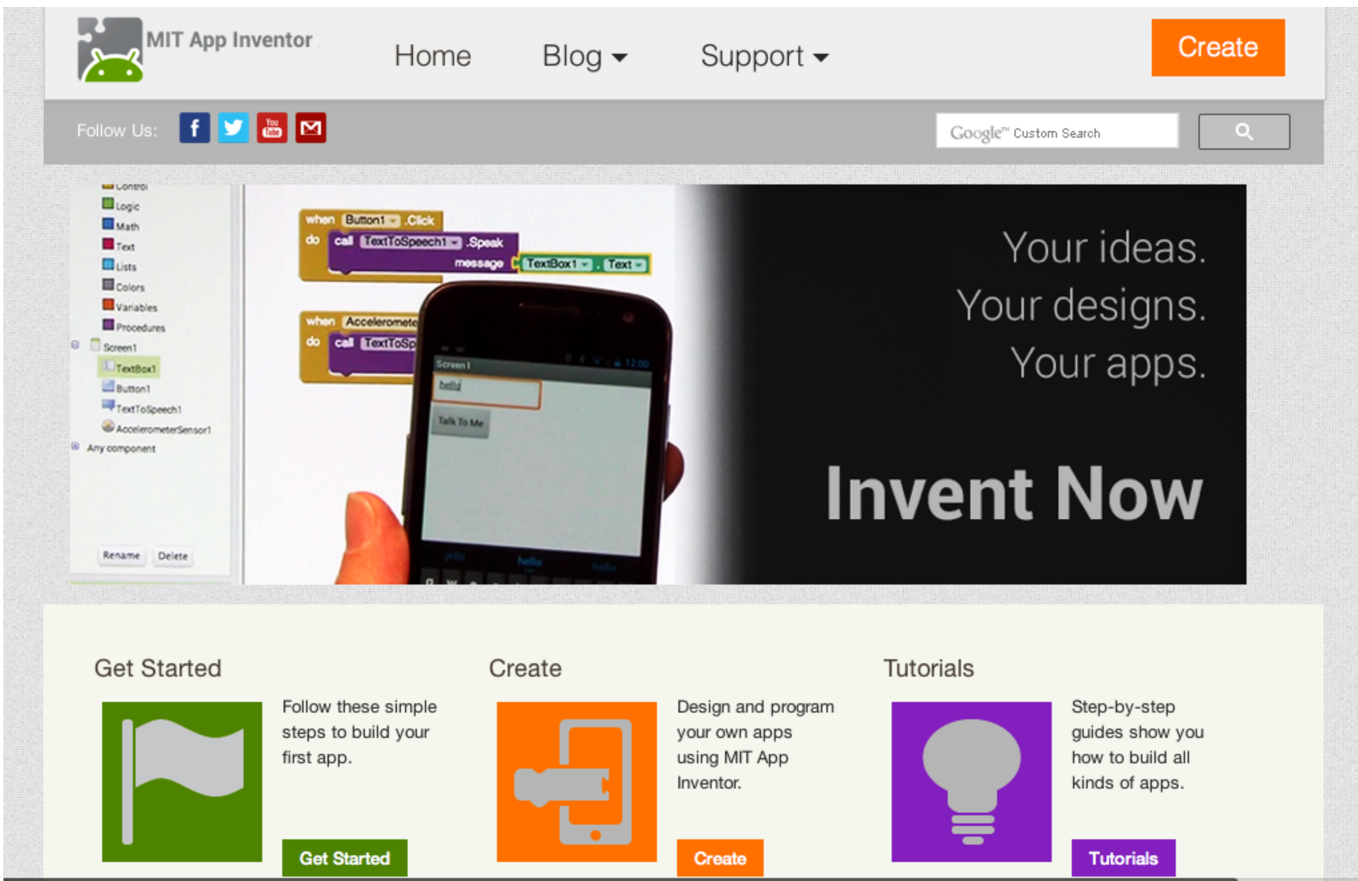
Now move on to TalkToMe Part 2 to make the app respond to shaking and to let users put in whatever phrase they want.

TalkToMe Part 2: Shaking and User Input

This tutorial shows you how to extend the basic TalkToMe app so that it responds to shaking, and so that the user can make the phone say any phrase s/he types in.

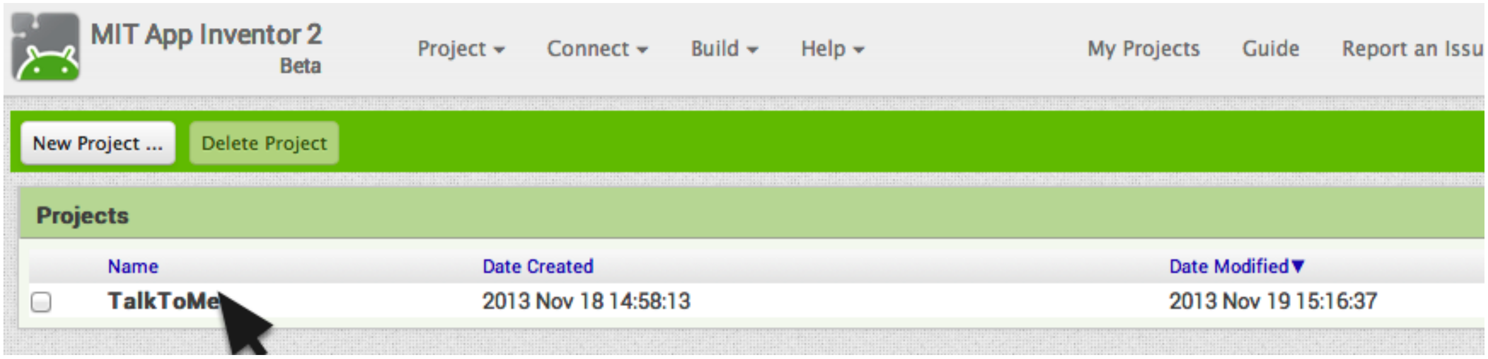
Go to App Inventor on the web and log in.

Go to appinventor.mit.edu and click "Create" or log in directly at ai2.appinventor.mit.edu.



Open the "TalkToMe" project that you worked on in the last tutorial.

App Inventor will always open the last project you worked on, so you may automatically be taken into your TalkToMe app.

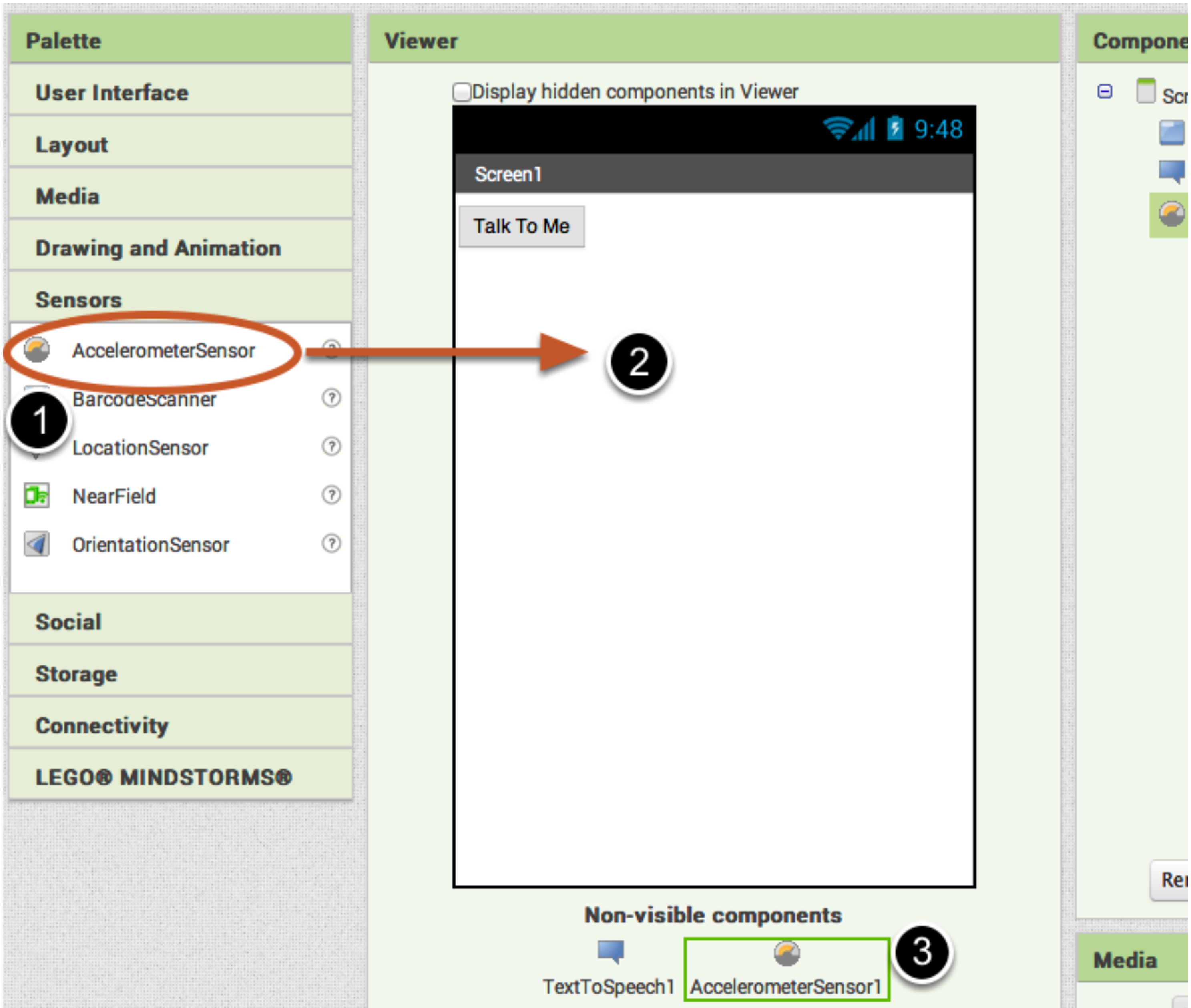


Go to the Designer Tab

Your project may open in the Designer. If it does not, click "Designer" in the upper right.

Add an Accelerometer Sensor

In the **Sensors** drawer, drag out an AccelerometerSensor component and drop it onto the Viewer. (It's a non-visible component, so it drops to the bottom of the screen.) NOTE: emulator users should skip this part and proceed to the next section of this tutorial called "Say Anything". (The emulator can not respond to shaking!)



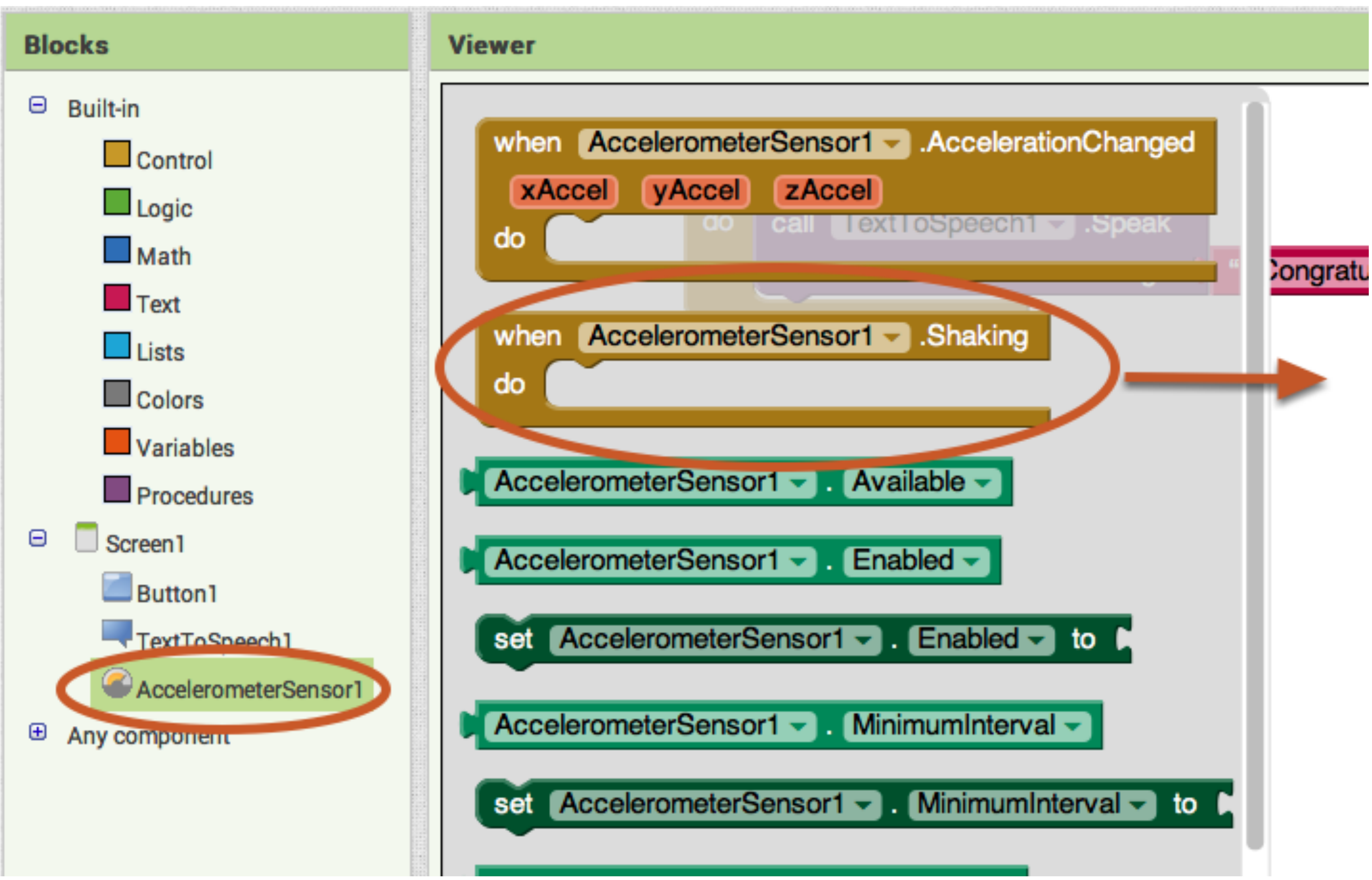
MIT App Inventor
appinventor.mit.edu

Go to the Blocks Editor

Click "Blocks" to program the new Accelerometer Sensor that you just added.

Program the Accelerometer Shaking event

Click the AccelerometerSensor1 drawer to see its blocks. Drag out the ***when AccelerometerSensor1.Shaking do*** block and drop it on the workspace.

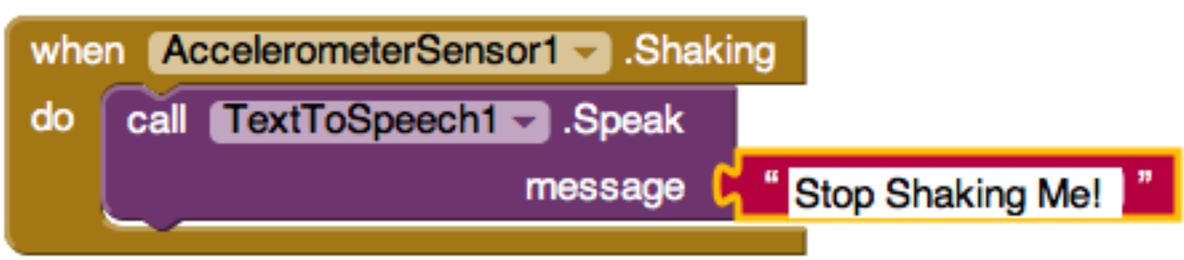


What do we want the app to do when the accelerometer detects shaking?

Copy and paste the blocks that are currently inside the when Button1.Click event handler. You can select the purple block, then hit the key combination on your computer to copy and then to paste. You'll have a second set of blocks to put inside the when Accelerometer.Shaking block.
(Alternatively, you can drag out a new *call TextToSpeech1.Speak block* from the TextToSpeech drawer, and a new pink *text block* from the Text drawer.)

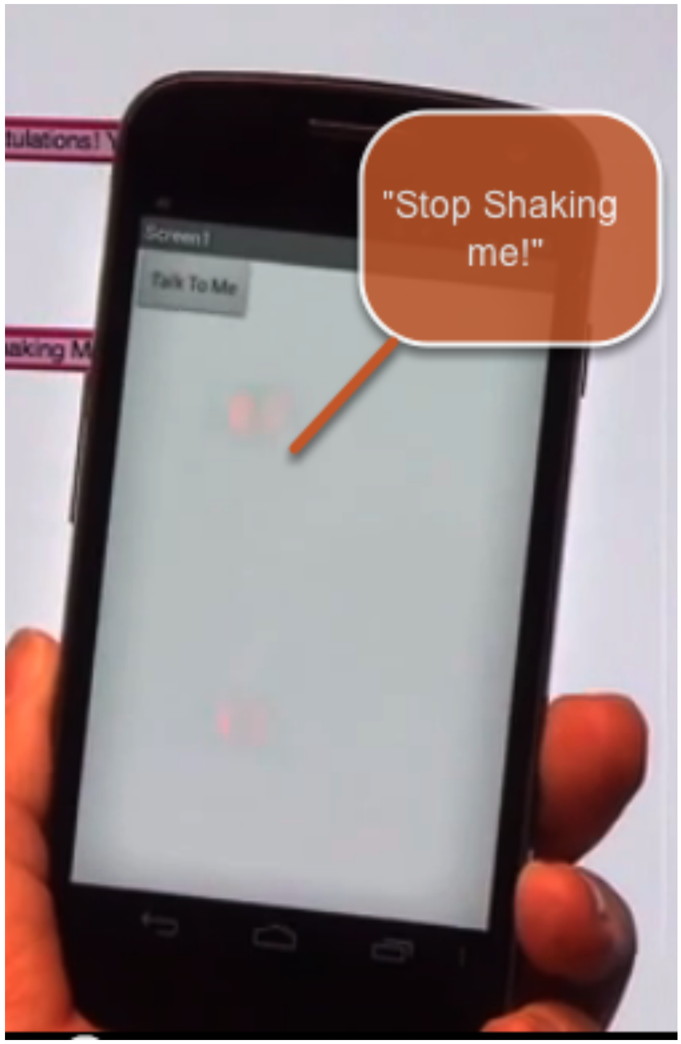
Change the phrase that is spoken when the phone is shaking.

Type in something funny for when the phone responds to shaking.



Test it out!

You can now shake your phone and it should respond by saying "Stop shaking me!" (or whatever phrase you put in.)

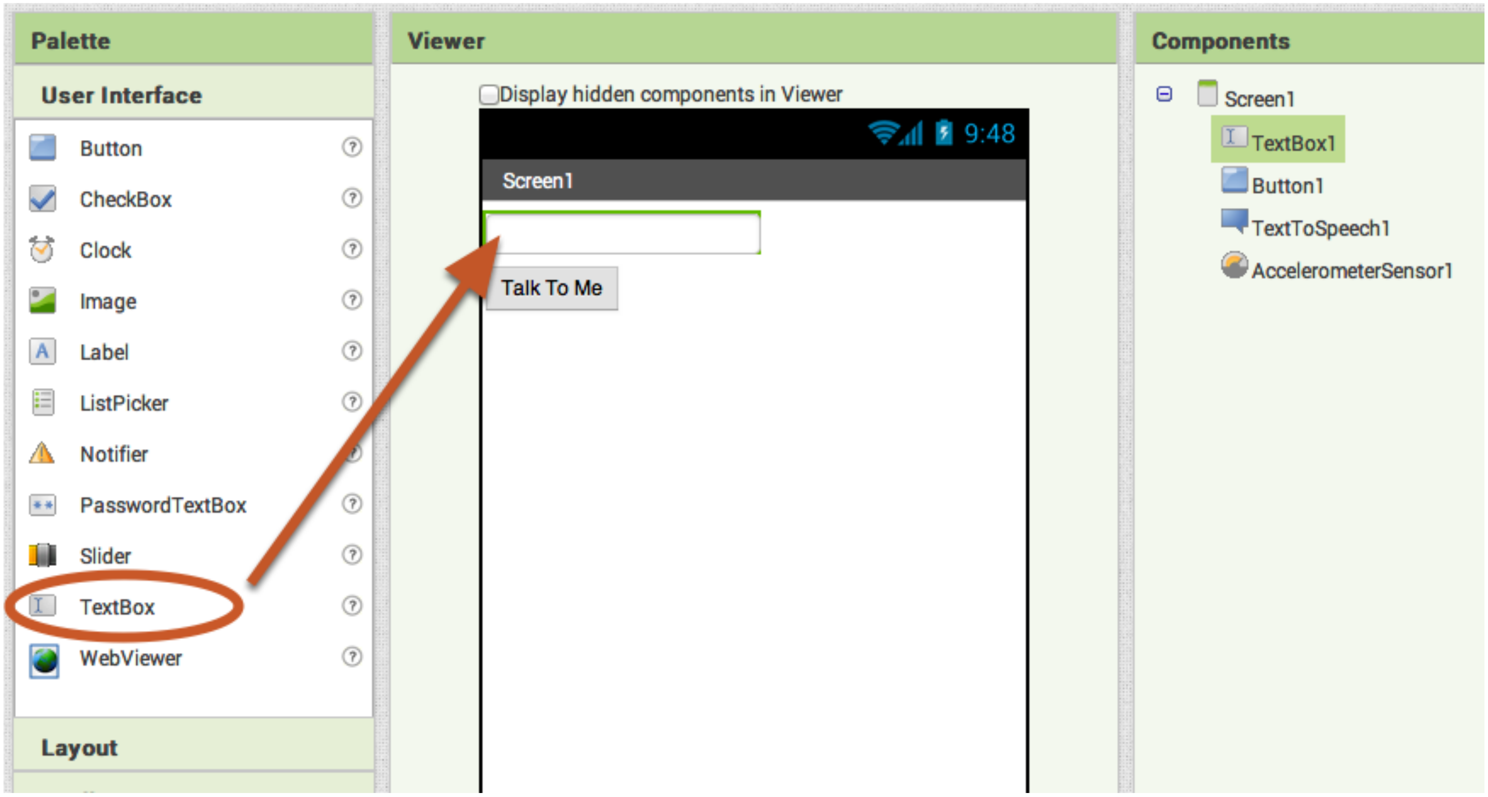


Say Anything

Is your phone talking to you? Cool! Now let's program the button click so that it causes the phone to speak whatever phrase the user put into the text box. Go back to the Designer.

Add a Text Box to your user interface.

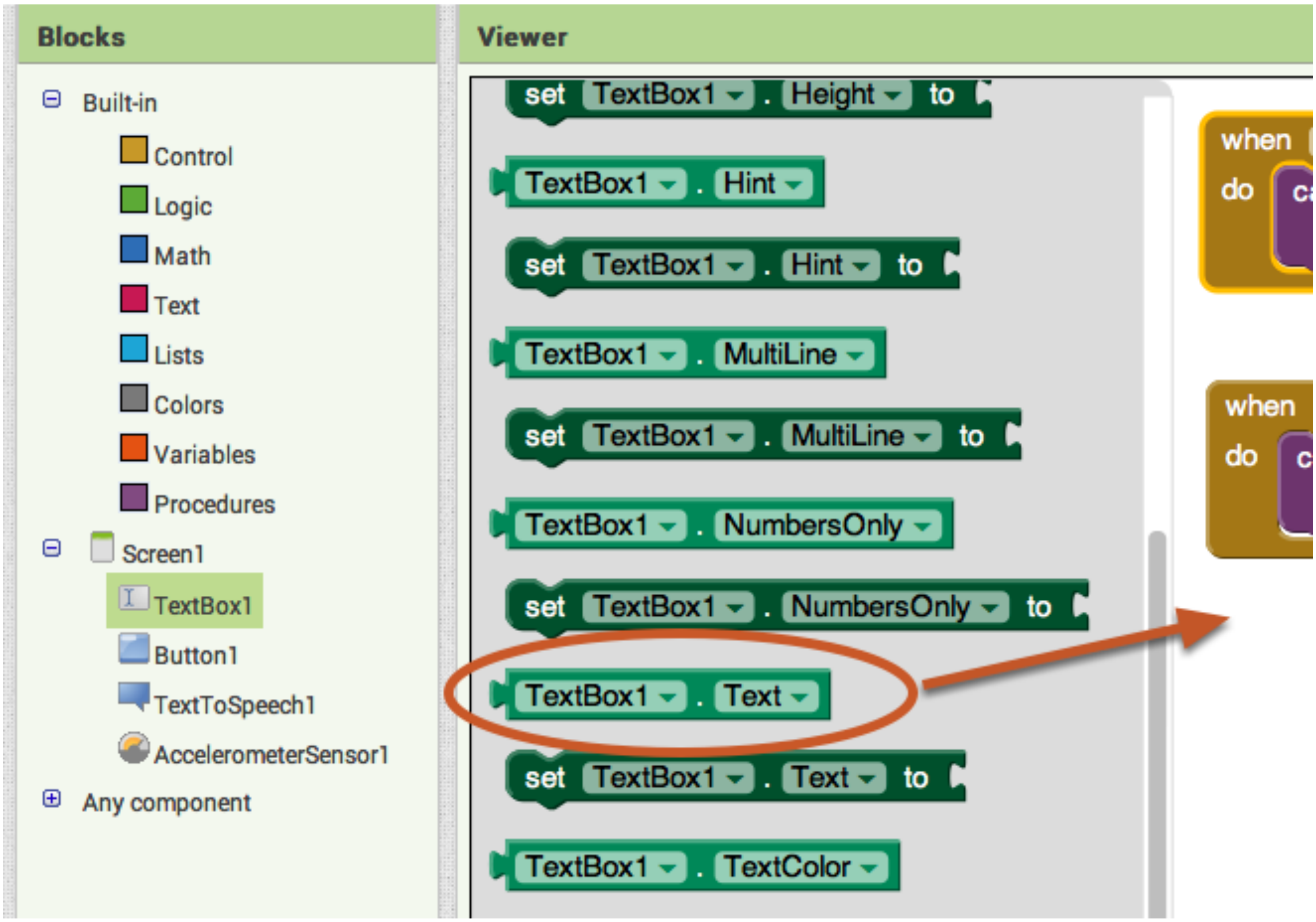
From the User Interface drawer, drag out a TextBox and put it above the Button that is already on the screen.



Back to the Blocks Editor!

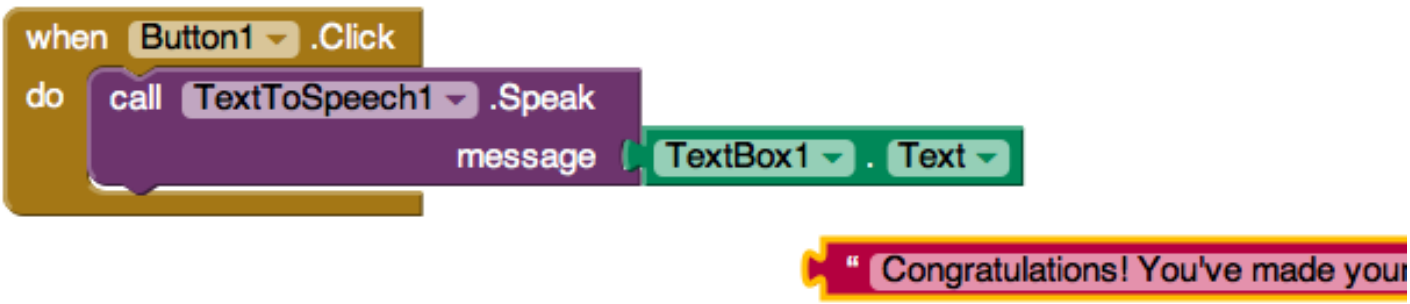
Get the text that is typed into the TextBox.

Get the text property of the TextBox1. The green blocks in the TextBox1 drawer are the "getters" and "setters" for the TextBox1 component. You want your app to speak out loud whatever is currently in the TextBox1 Text property (i.e. whatever is typed into the text box). Drag out the ***TextBox1.Text*** getter block.



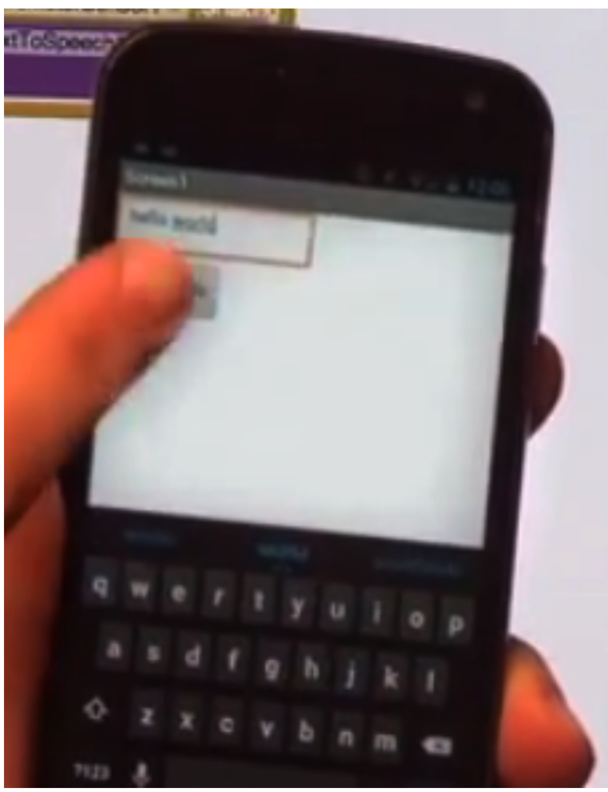
Set the Button Click event to speak the text that is in the Text Box.

Pull out the "congratulations..." text box and plug in the TextBox1.Text block. You can throw the pink text block away by dragging it to the trash in the lower right corner of the workspace.



Test your app!

Now your app has two behaviors: When the button is clicked, it will speak out loud whatever words are currently in the Text Box on the screen. (if nothing is there, it will say nothing.)
The app will also say "Stop Shaking Me" when the phone is shaken.



Congrats! You've built a real app!

Give some thought to what else this app could do. Here are some ideas for extensions:

- Random phrase generator
- Mad Libs - player chooses noun, verb, adjective, adverb, person and it picks one from a list that you program.
- Magic 8 Ball App
- Name picker - useful for teachers to call on a student

You could also play around with Speech-To-Text. Have fun!

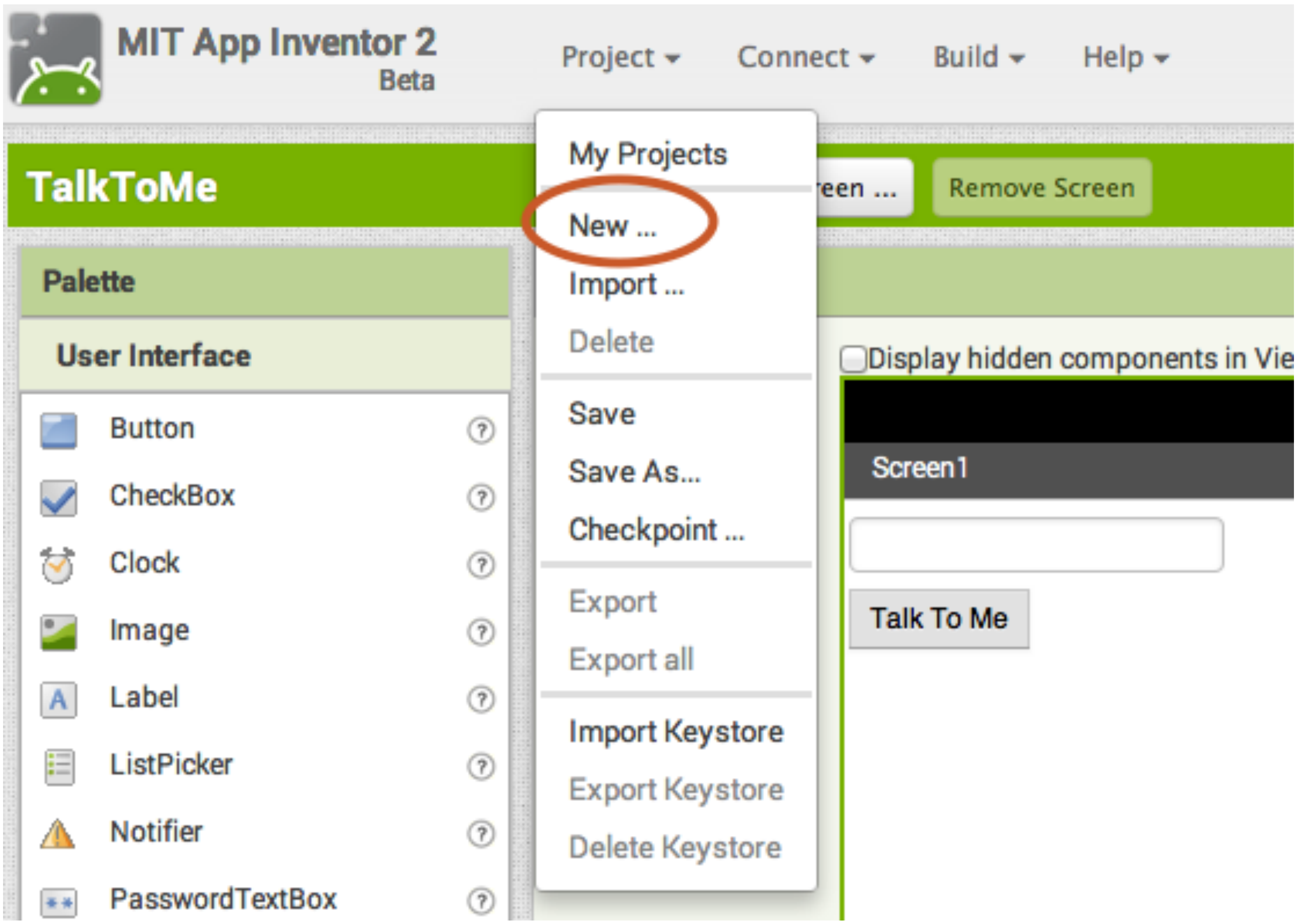
MIT App Inventor
appinventor.mit.edu

BallBounce: A simple game app

In this tutorial, you will learn about animation in App Inventor by making a Ball (a sprite) bounce around on the screen (on a Canvas).

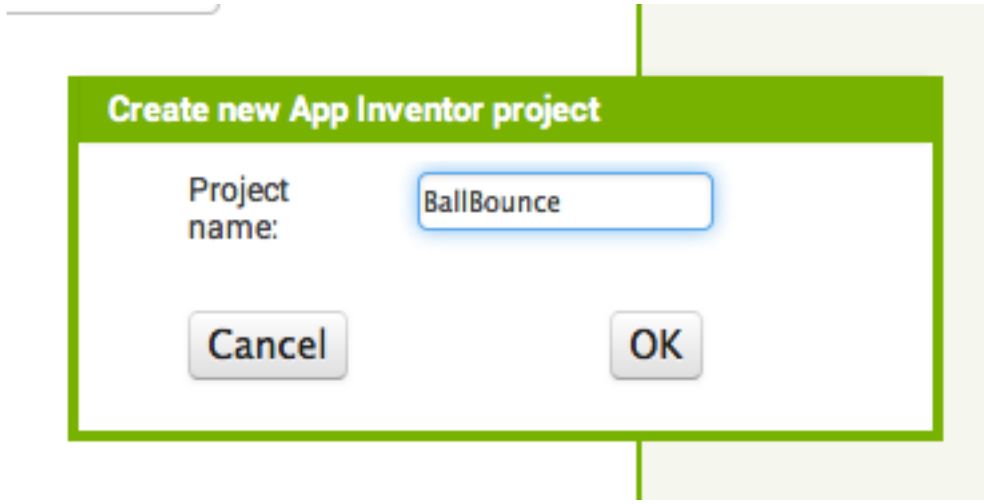
Start a New Project

If you have another project open, go to My Projects menu and choose New Project.



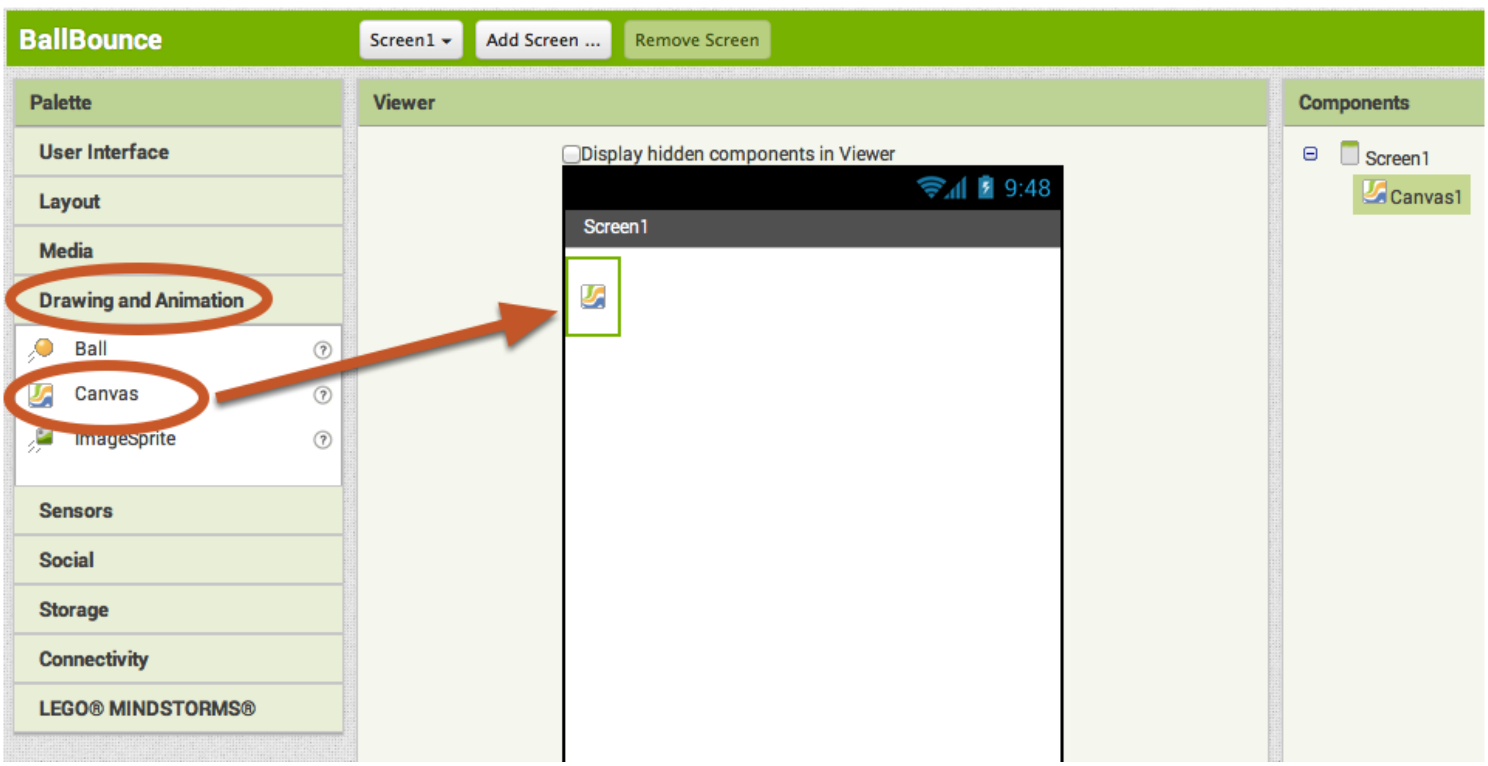
Name the Project

Call it something like "BallBounce". Remember, no spaces. But underscores are OK.



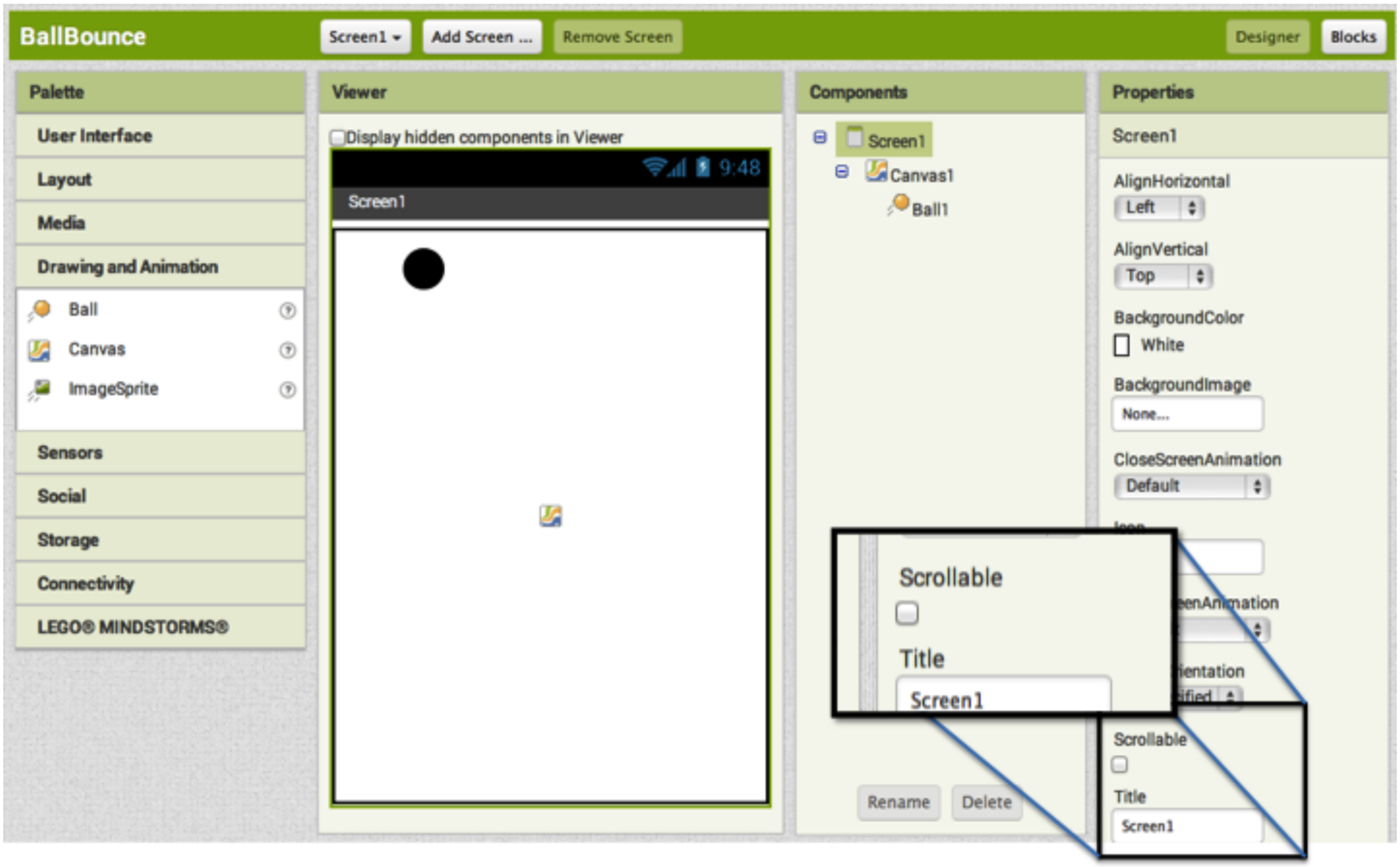
Add a Canvas

From the **Drawing and Animation drawer,** drag out a **Canvas component** and drop it onto the viewer.



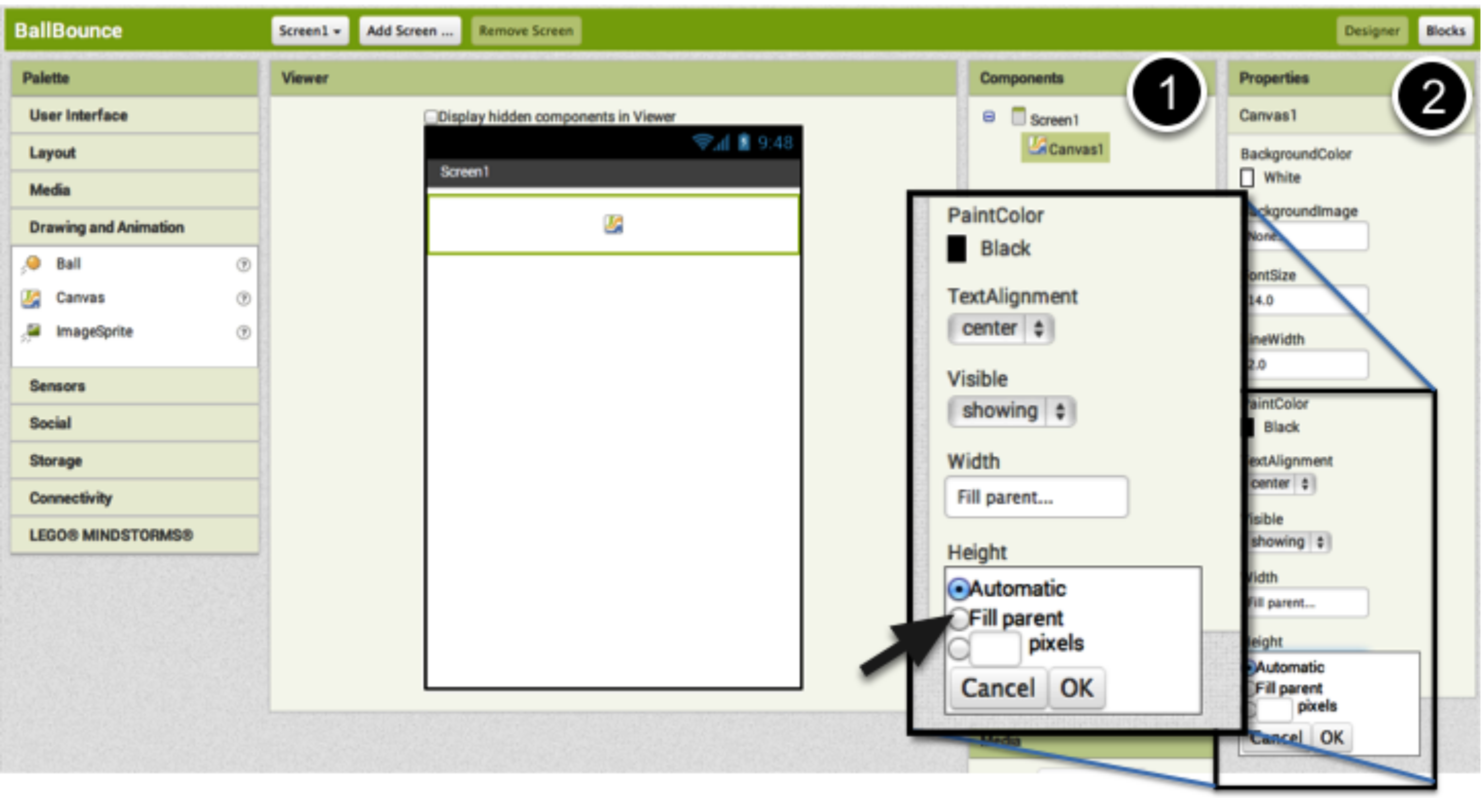
Set the Screen so that it does not scroll

The default setting for App Inventor is that the screen of your app will be "scrollable", which means that the user interface can go beyond the limit of the screen and the user can scroll down by swiping their finger (like scrolling on a web page). When you are using a Canvas, you have to **turn off the "Scrollable" setting** (UNCHECK THE BOX) so that the screen does not scroll. This will allow you to make the Canvas to fill up the whole screen.



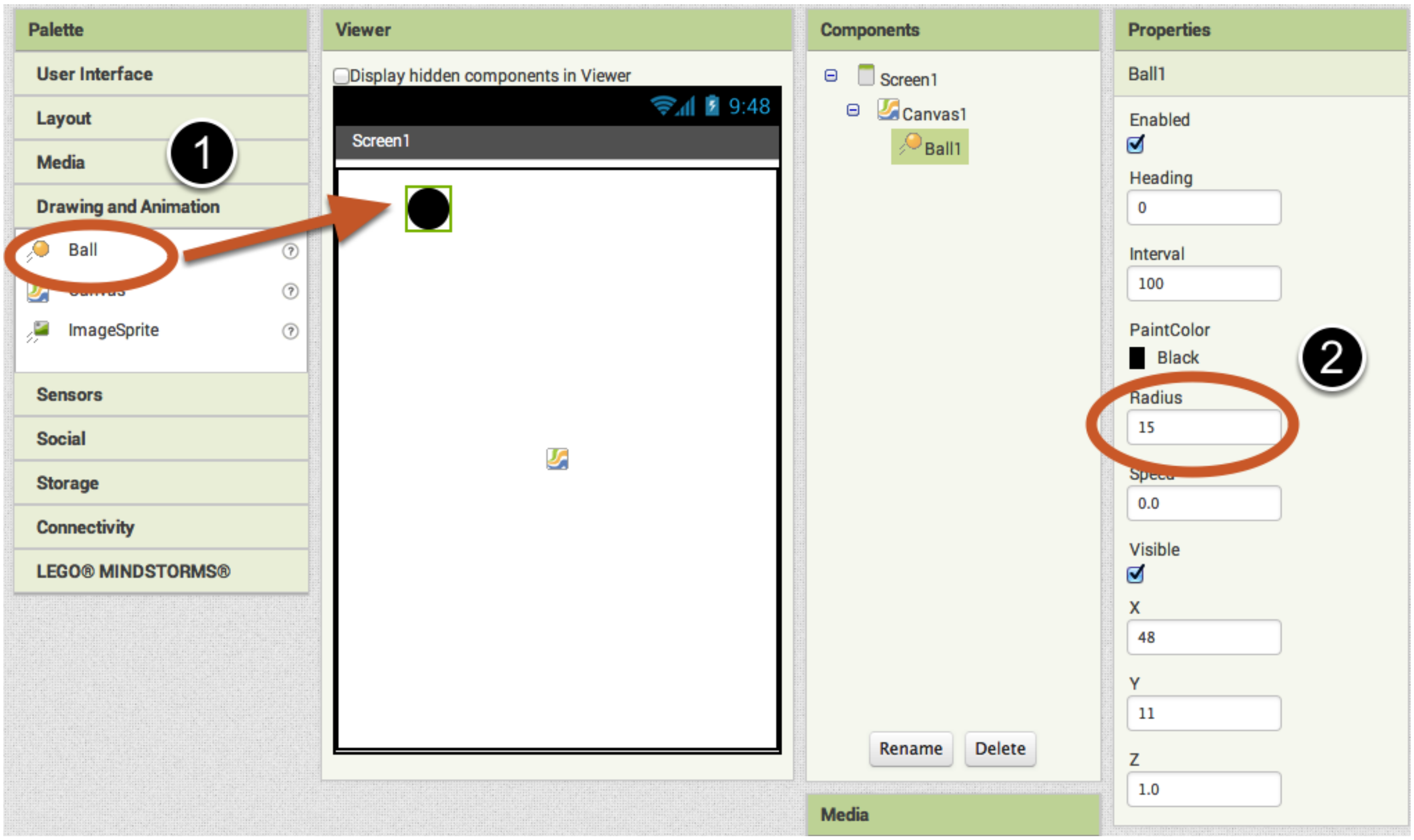
Change the Height and Width of the Canvas to Fill Parent

Make sure the Canvas component is selected (#1) so that its properties show up in the Properties Pane (#2). Down at the bottom, **set the Height property to "Fill Parent".** Do the **same with the Width property.**

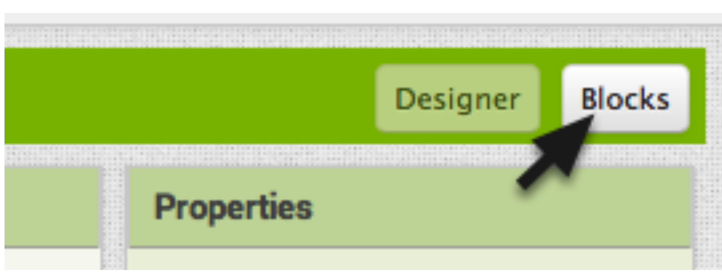


Add a Ball

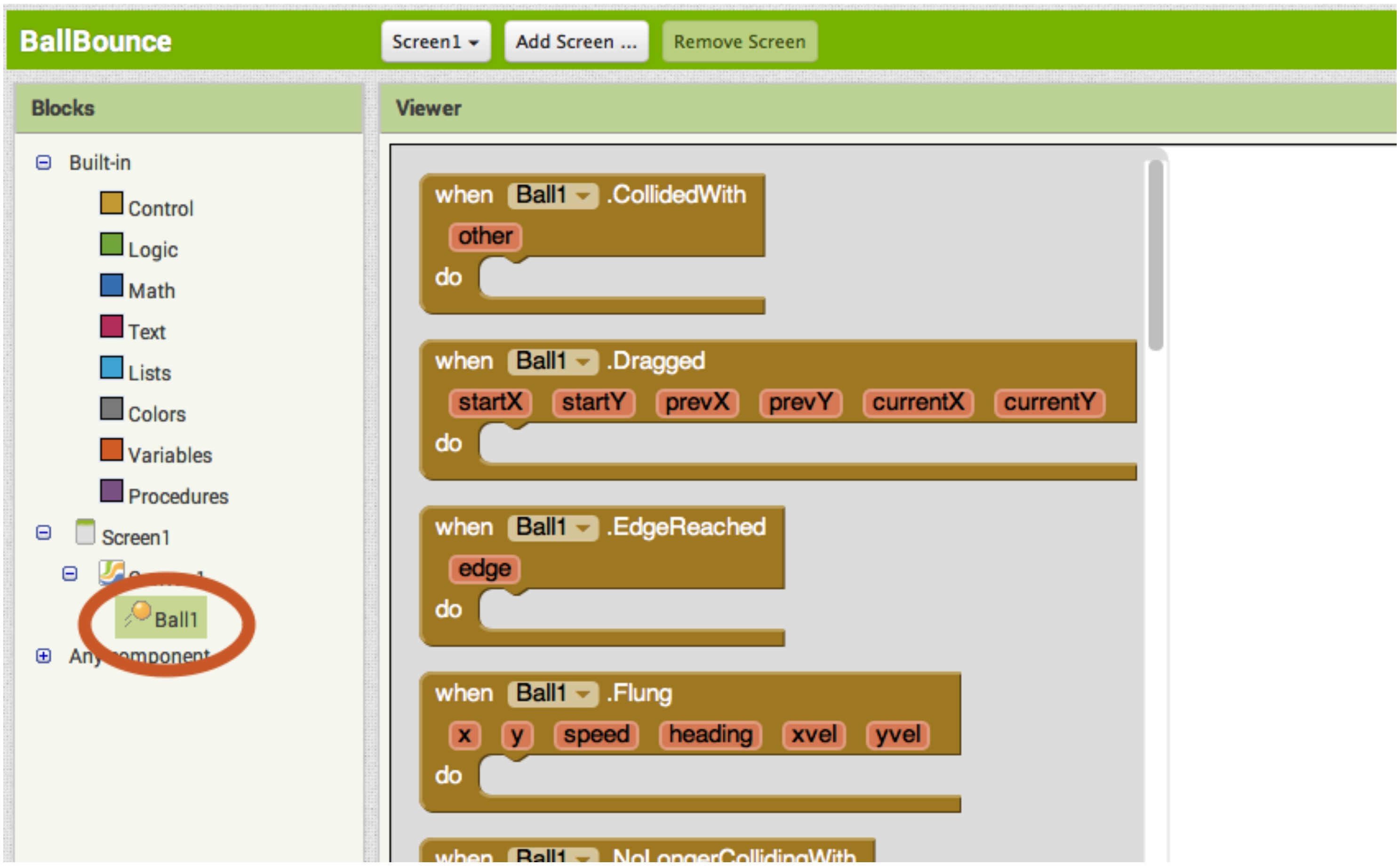
Now that we have a Canvas in place, we can add a Ball Sprite. This can also be found in the **Drawing and Animation drawer**. Drag out a **Ball component** and drop it onto the Canvas (#1). If you'd like the ball to show up better, you can change its **Radius property** in the Properties pane (#2).



Open the Blocks Editor.

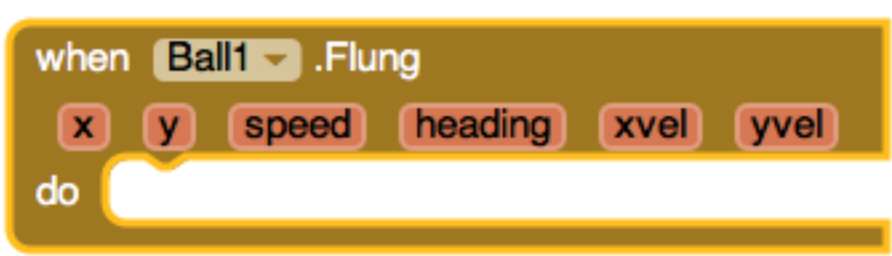


Open the Ball1 Drawer to view the Ball's blocks.



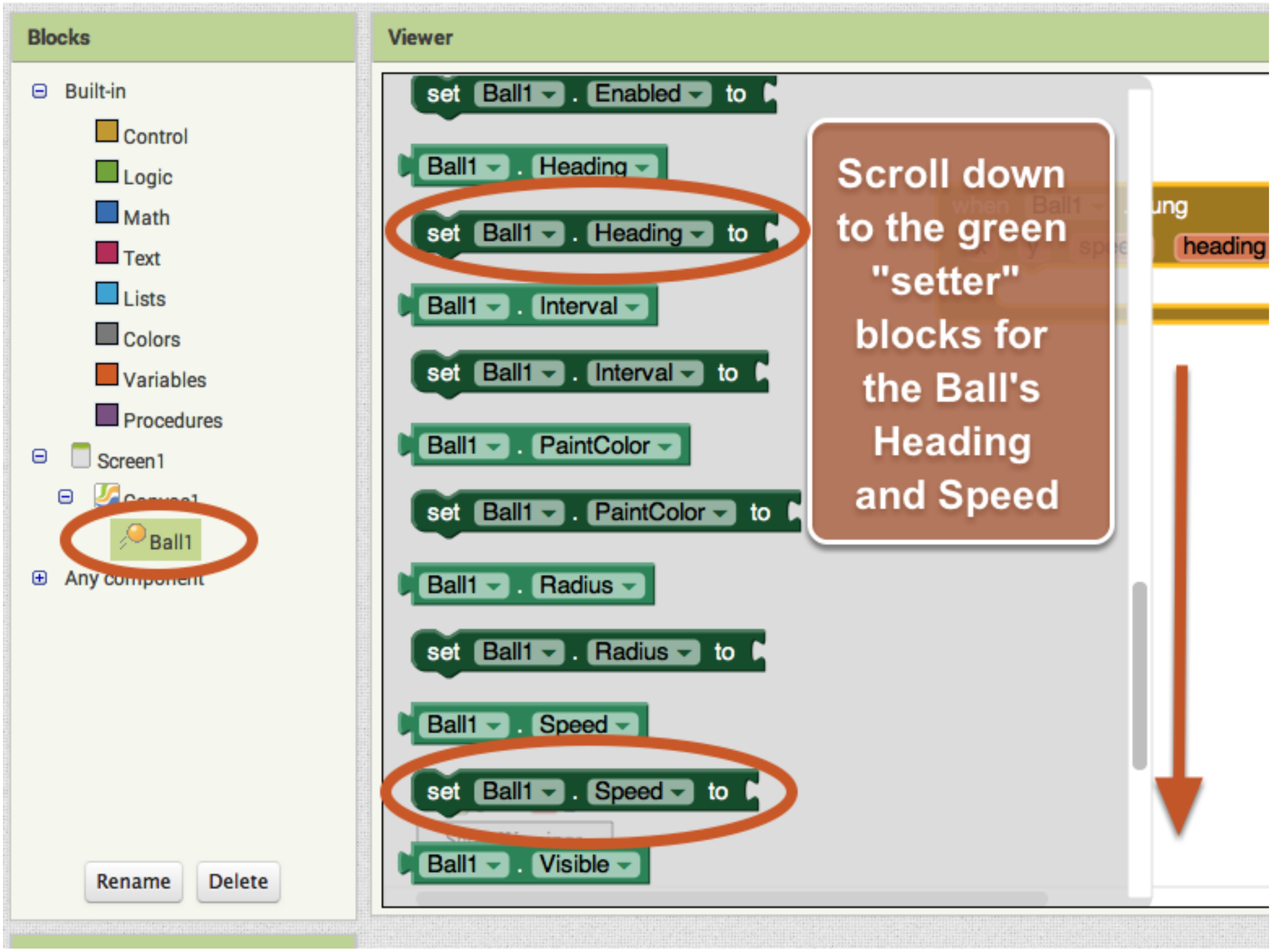
Drag out the Flung Event Handler

Choose the block **when Ball1.Flung** and drag-and-drop it onto the workspace. Flung refers to the user making a "Fling gesture" with his/her finger to "fling" the ball. Fling is a gesture like what a golf club does, not like how you launch Angry Birds! In App Inventor, the event handler for that type of gesture is called *when Flung*.



Set the Ball's Heading and Speed. First get the setter blocks.

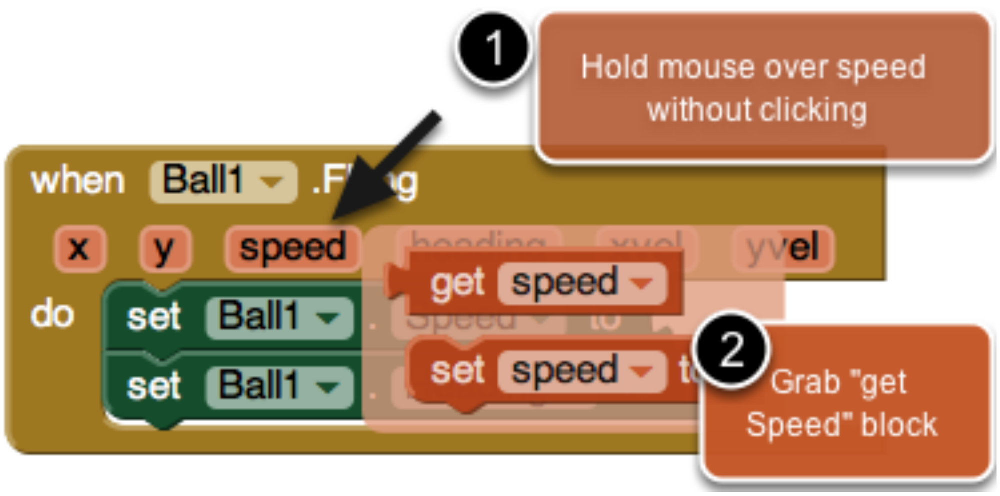
Open the Ball drawer and scroll down in the list of blocks to get the **set Ball1.Heading** and **set Ball1.Speed** blocks



Plug the set Ball1.Speed and set Ball1.Heading into the Fling event handler

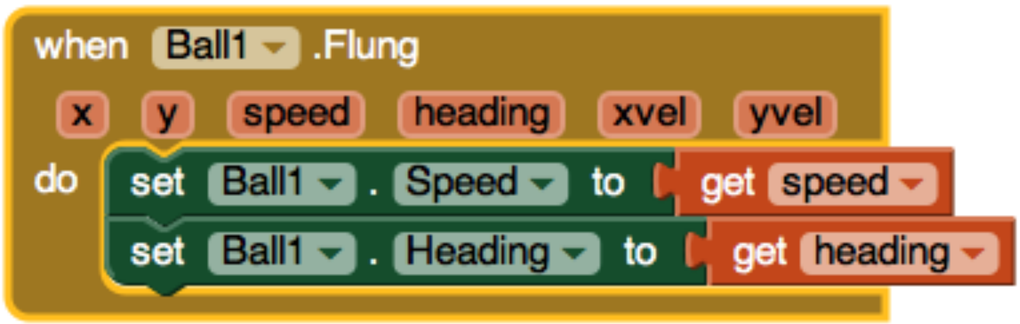
Set the Ball's speed to be the same as the Fling gesture's speed

Mouse over the "speed" parameter of the **when Ball1.Flung** event handler. The get and set blocks for the speed of the fling will pop up. Grab the **get speed** block and plug that into the **set Ball1.Speed** block.



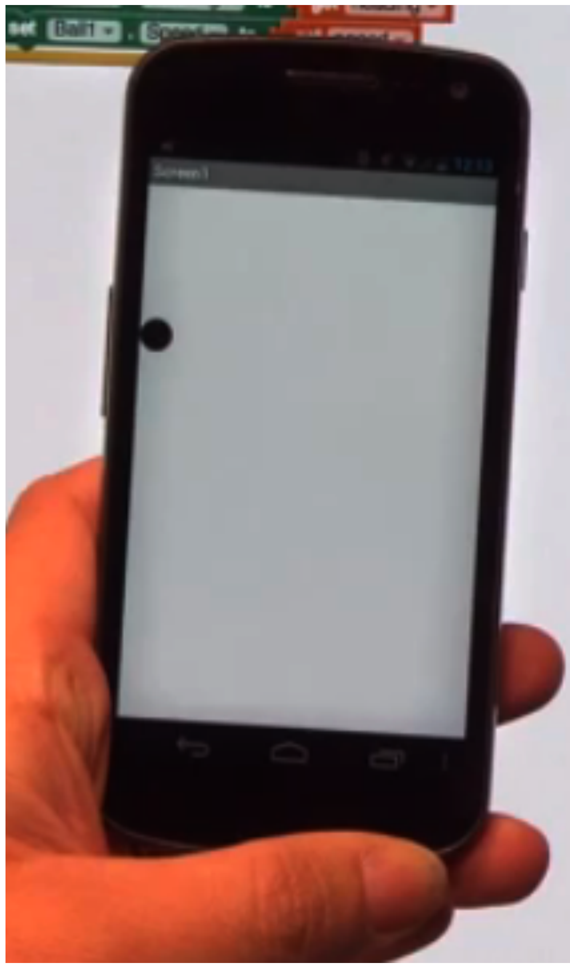
Set the Ball's heading to be the same as the Fling gesture's heading

Do the same for the Ball's heading. Mouse over the **heading** parameter and you'll see the **get heading** block appear. Grab that block, and click it into the **set Ball1.Heading** block.



Test it out

A good habit while building apps is to test while you build. App Inventor lets you do this easily because you can have a live connection between your phone (or emulator) and the App Inventor development environment. If you don't have a phone (or emulator) connected, go to the connection instructions and then come back to this tutorial. (Connection instructions are in Tutorial #1 or on the website under "Getting Started".)

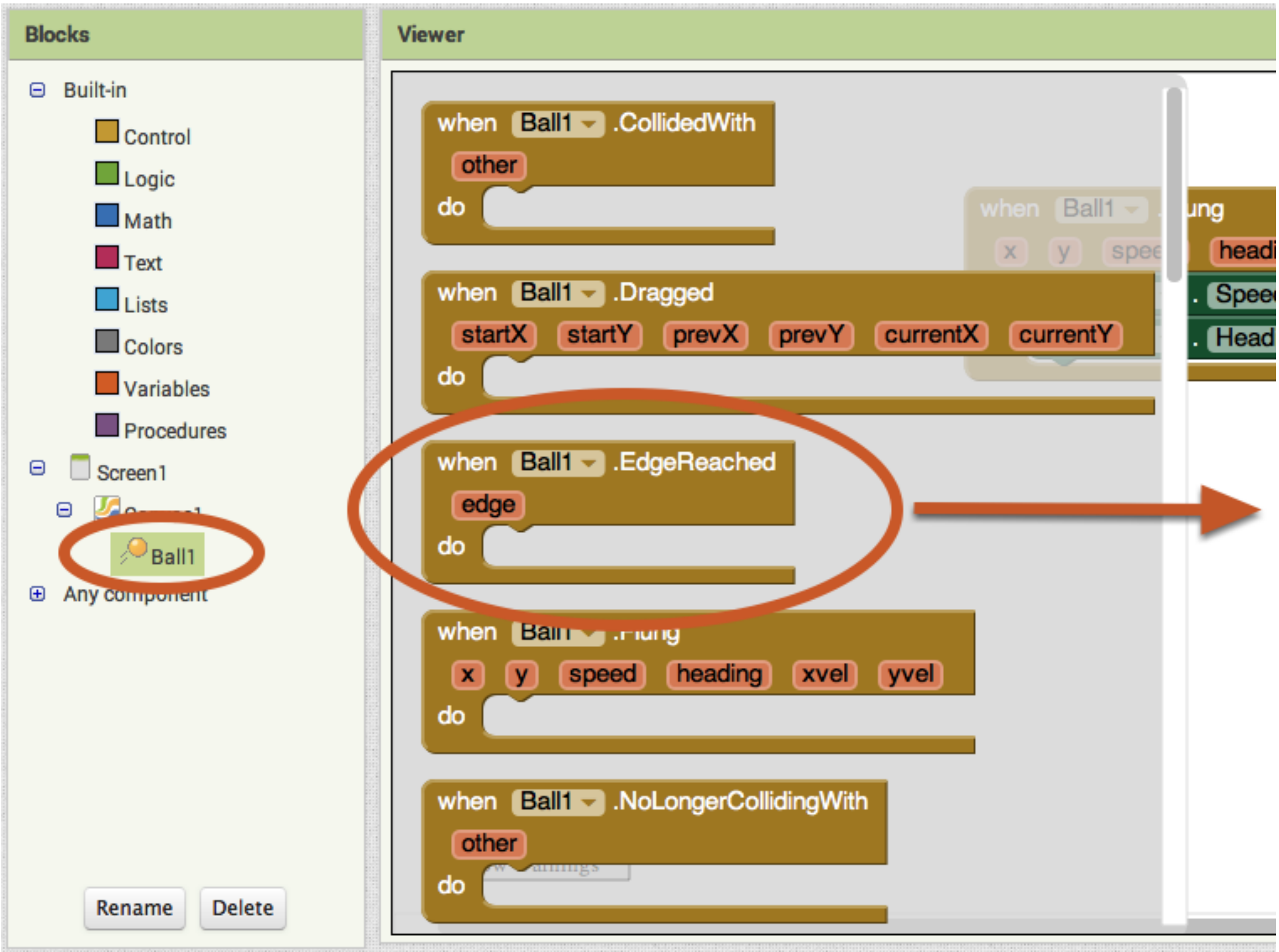


Why does the Ball get stuck on the side of the screen?!

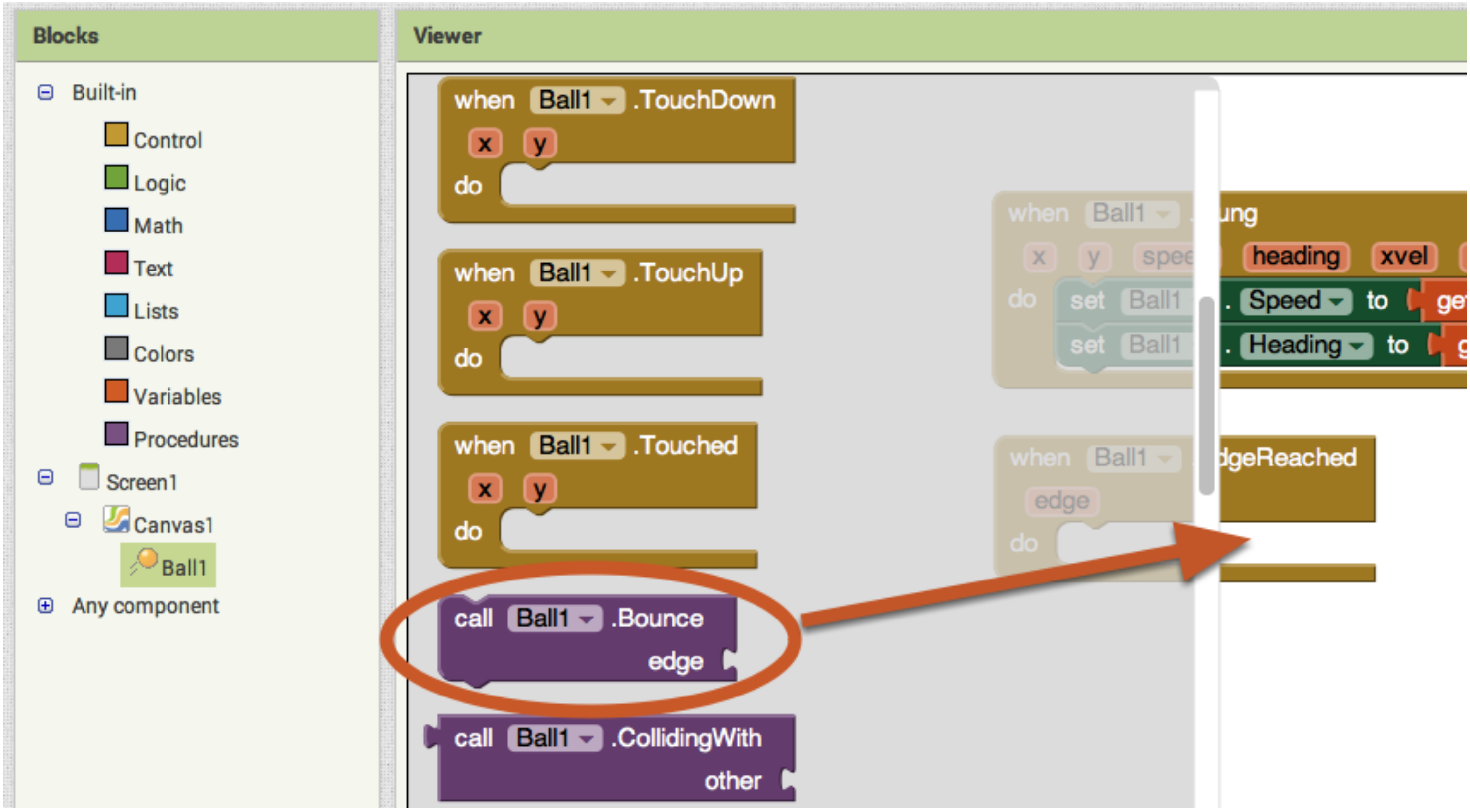
After flinging your ball across the screen, you probably noticed that it got stuck on the side. This is because the ball's heading has not changed even though it hit the side of the canvas. To make the ball "bounce" of the edge of the screen, we can program in a new event handler called "When Edge Reached".

Add an Edge Reached Event

Go into the Ball1 drawer and pull out a **when Ball1.EdgeReached do** event.

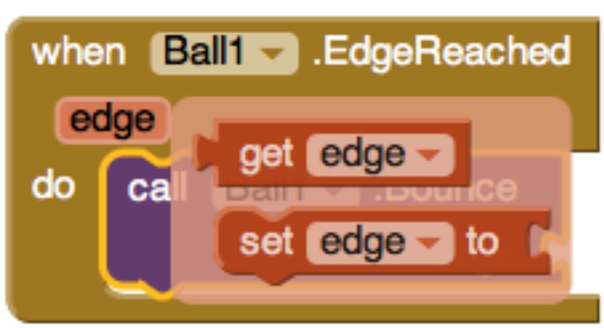


Go back into the Ball1 drawer and pull out a Ball.Bounce block.

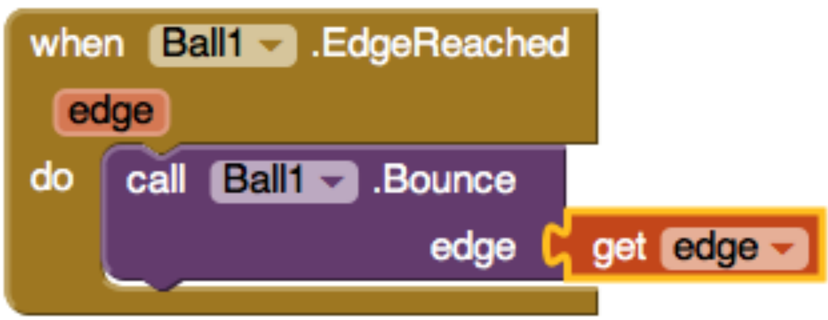


Add the edge value for the Ball.Bounce block

The **Ball.Bounce** method needs an edge argument. Notice that the **Ball1.EdgeReached** event has an "edge" as a parameter. We can take the **get edge** block from that argument and plug it into the call **Ball1.Bounce** method. Grab the **get edge** block by mousing over (hover your mouse pointer over) the "edge" parameter on the w**hen Ball1.EdgeReached** block.

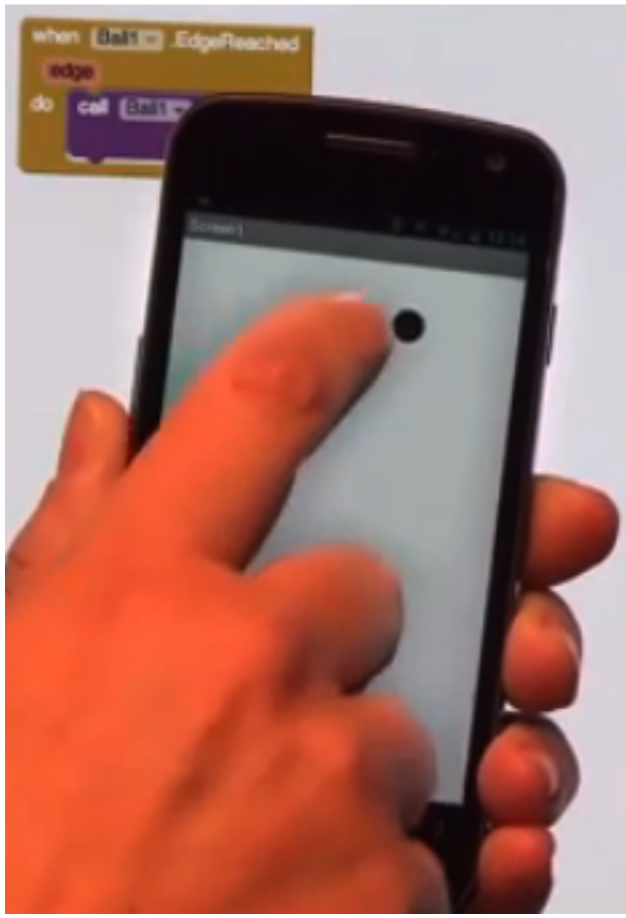


Your final blocks should look like this. Now test it out!



Test it out!

Now, when you fling the ball, it should bounce off the edges of the canvas. Great job!



There are many ways to extend this app.

Here are some ideas... but the possibilities are endless!
- Change the color of the ball based on how fast it is moving or which edge it reaches.
- Scale the speed of the ball so that it slows down and stops after it gets flung.
- Give the ball obstacles or targets to hit
- Introduce a paddle for intercepting the ball, like a Pong game

Visit the App Inventor website to find tutorials that help you extend this app, particularly the Mini Golf tutorial.

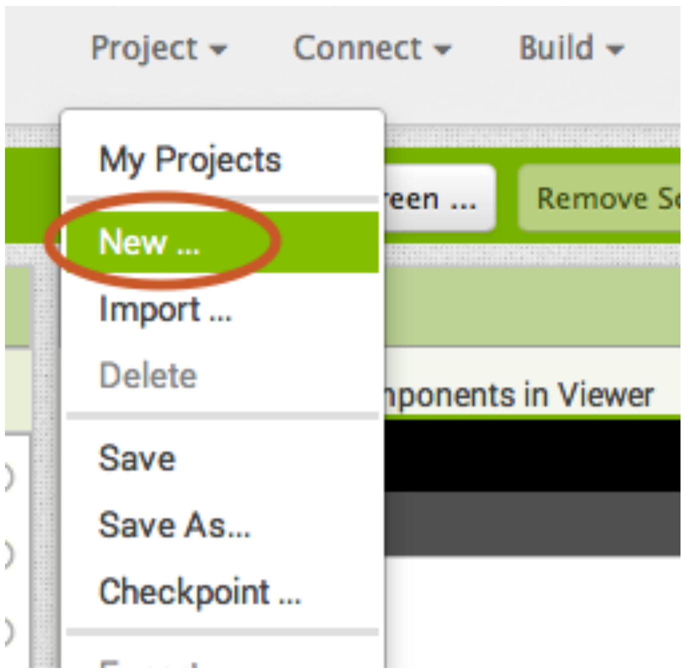
Have fun with these extensions, or others that you think up!

DigitalDoodle: Drawing App

This tutorial will show you how to draw a line on the screen as the user drags a finger around.

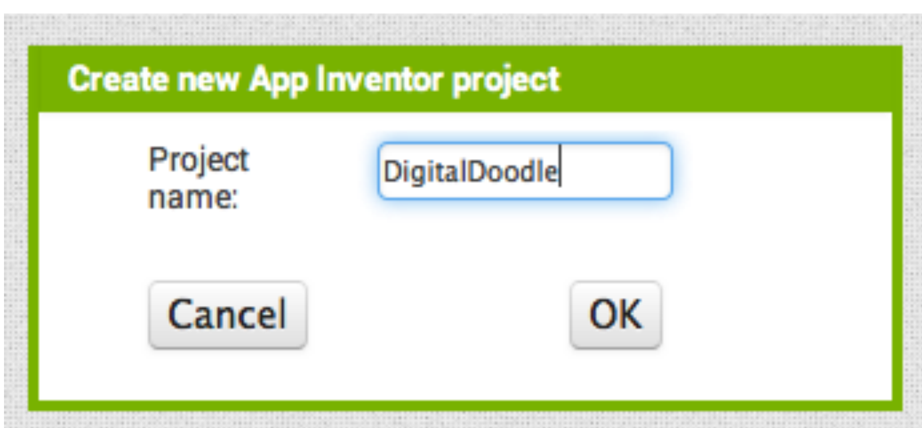
Start a New Project

From the My Projects page, click New Project. If you have another project open, go to My Projects menu and choose New Project.



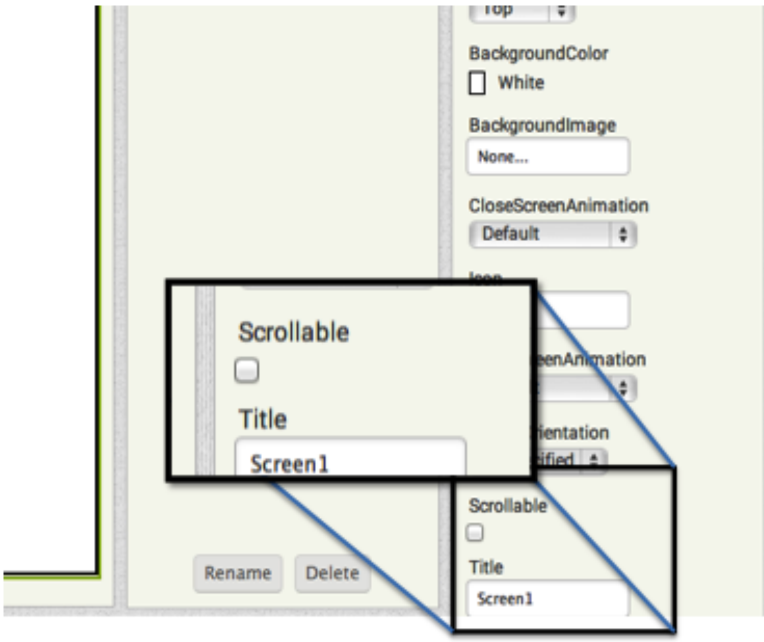
Name the Project

Call this project DigitalDoodle, or create your own name for this drawing app.



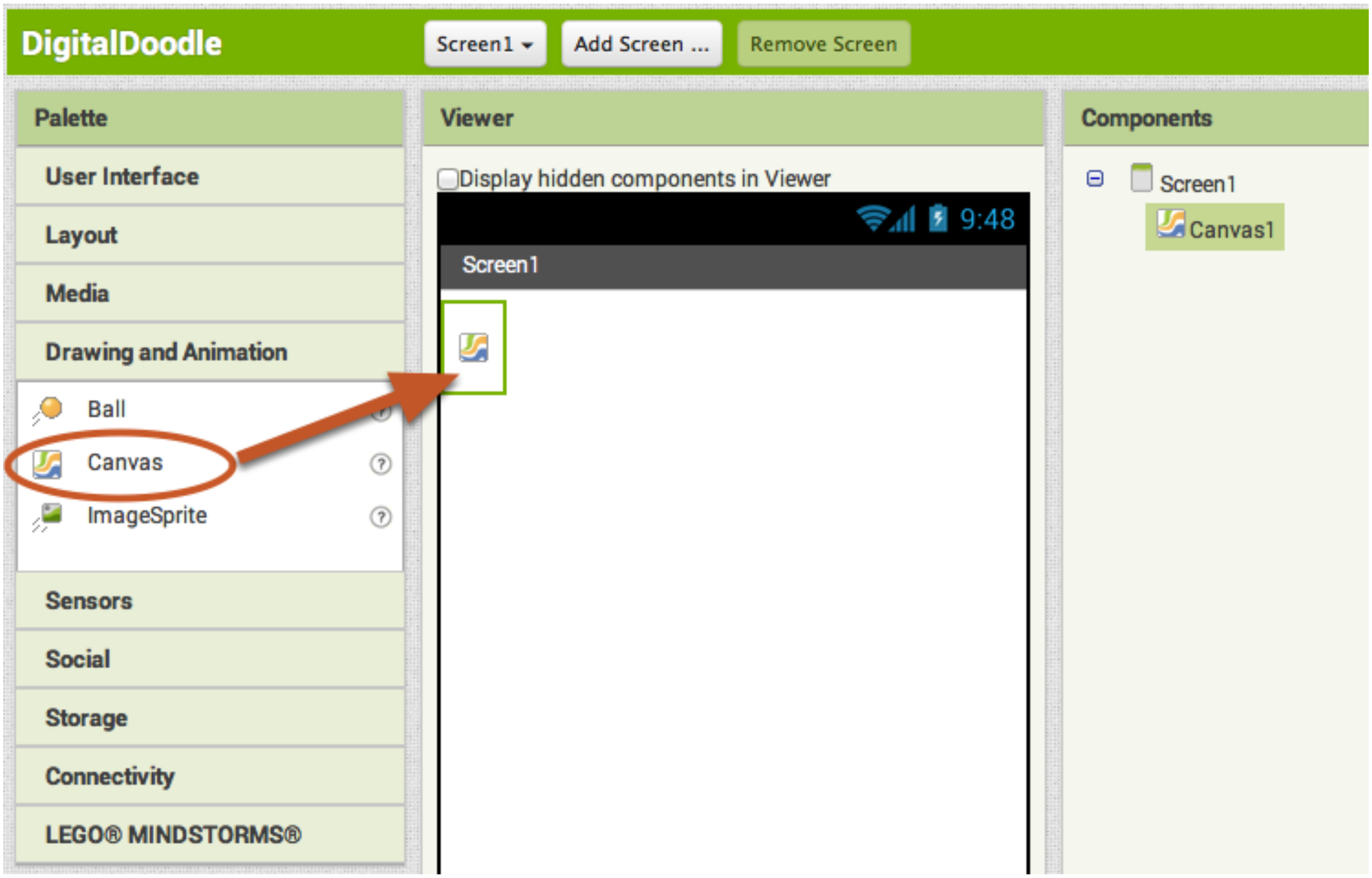
Set the Screen so that it does not scroll

The default setting for App Inventor is that the screen of your app will be "scrollable", which means that the user interface can go beyond the limit of the screen and the user can scroll down by swiping their finger (like scrolling on a web page). When you are using a Canvas, you have to **turn off the "Scrollable" setting** (UNCHECK THE BOX) so that the screen does not scroll. This will allow you to make the Canvas to fill up the whole screen.



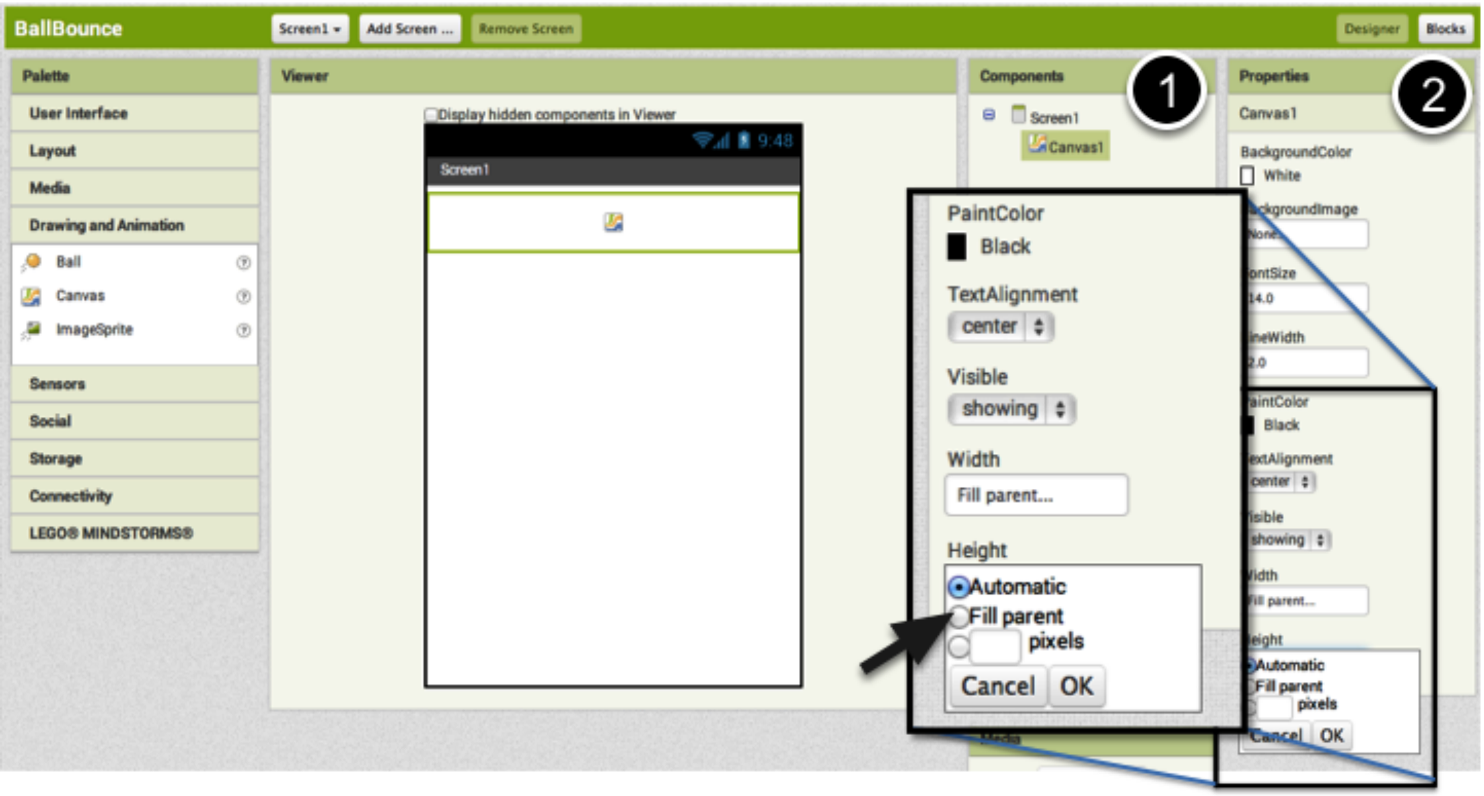
Add a Canvas

From the Drawing and Animation drawer, drag out a Canvas component.



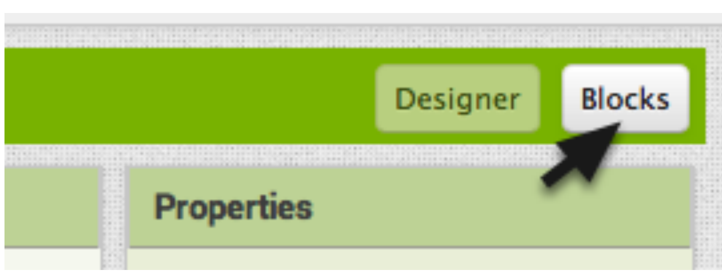
Change the Height and Width of the Canvas to Fill Parent

Make sure the Canvas component is selected (#1) so that its properties show up in the Properties Pane (#2). Down at the bottom, **set the Height property to "Fill Parent".** Do the **same with the Width property.**



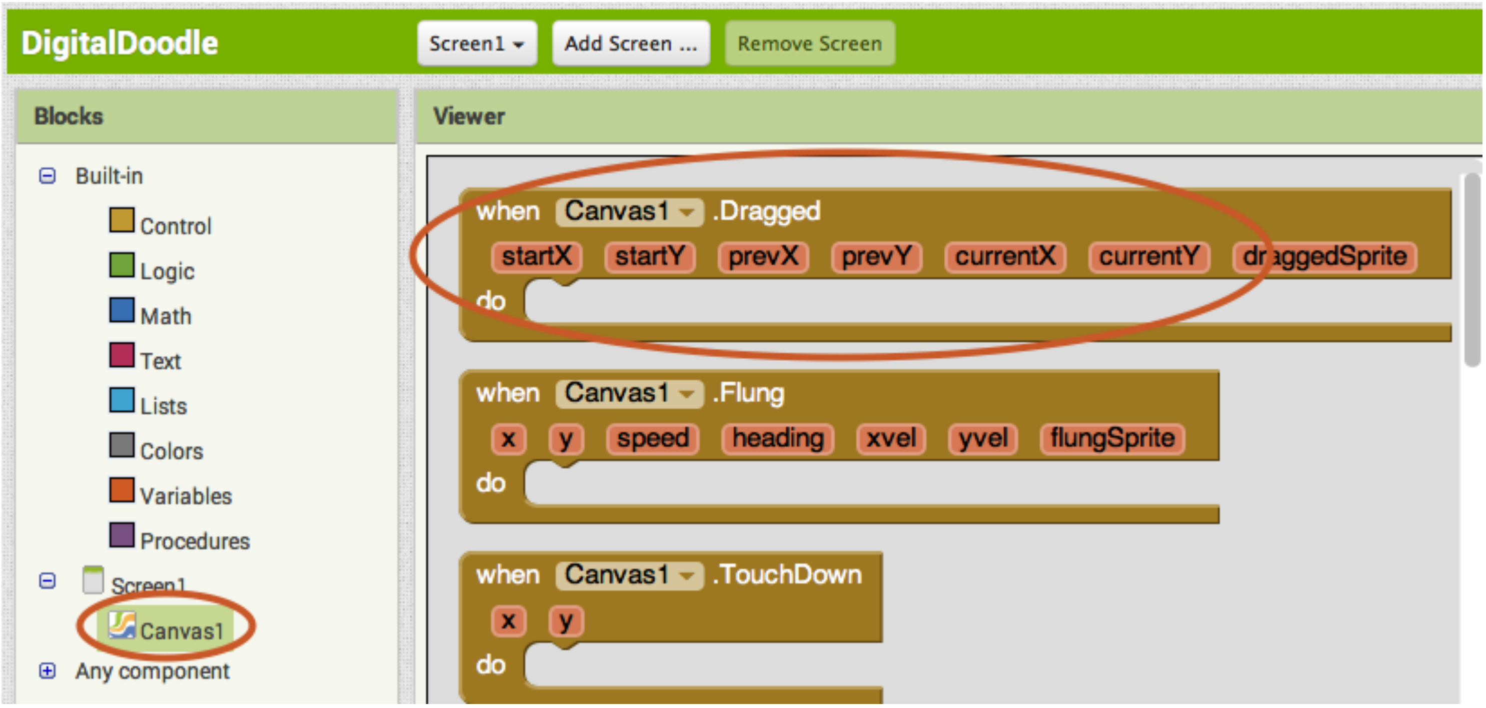
That's all for the Designer! Go over to the Blocks.

Believe it or not, for now this app only needs a Canvas. Go into the Blocks Editor to program the app.



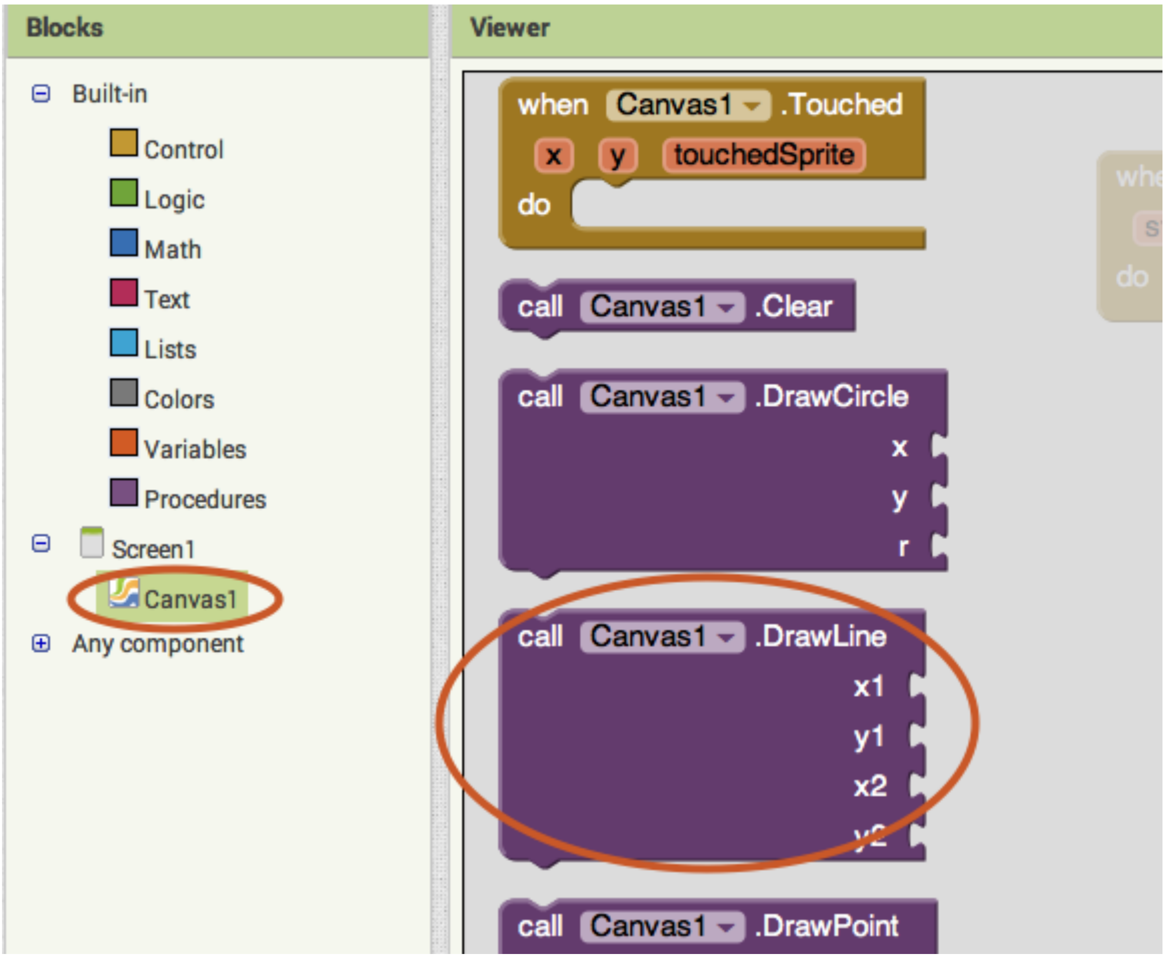
Get a Canvas.Dragged event block

In the Canvas1 drawer, pull out the **when Canvas1.Dragged** event.



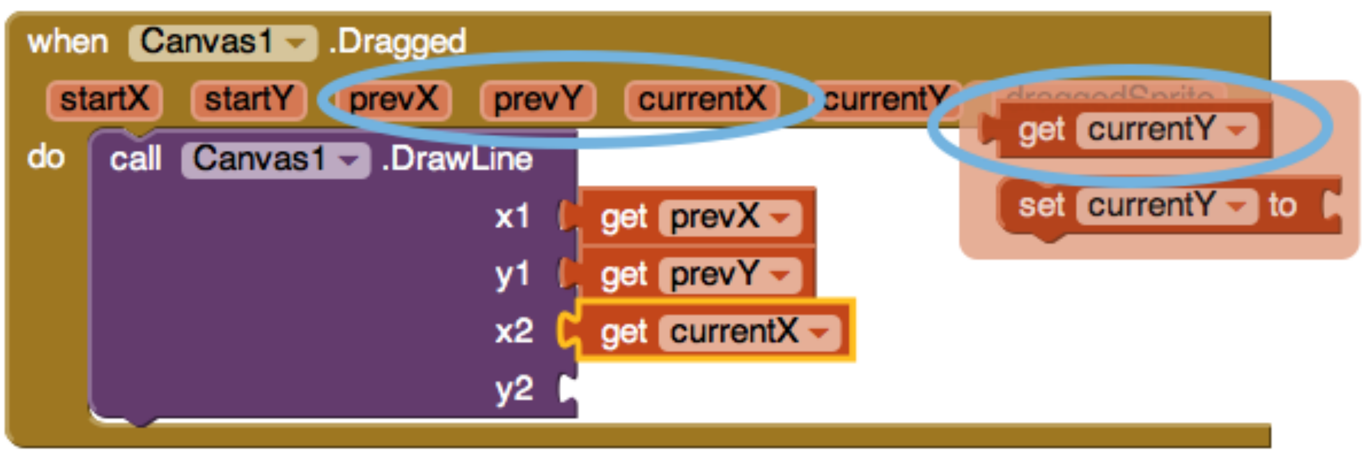
Get a Canvas.DrawLine call block

In the Canvas1 drawer, pull out the **when Canvas1.DrawLine** method block..



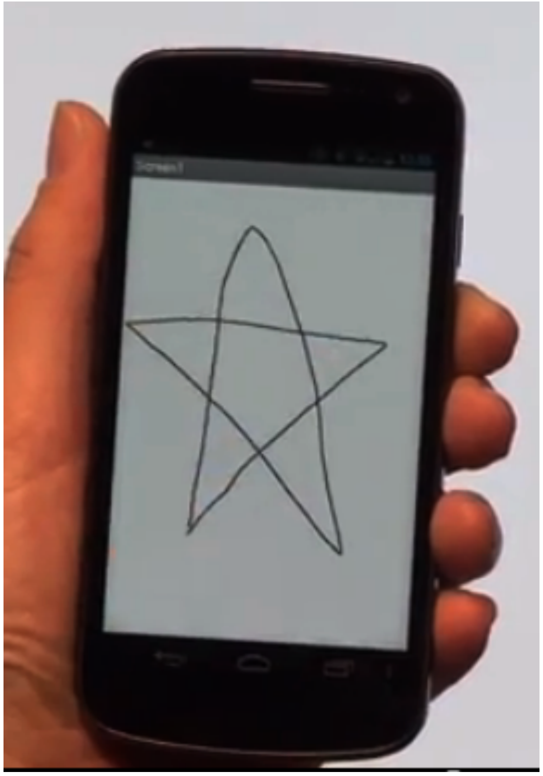
Use the get and set blocks from the Dragged block to fill in the values for the Draw Line block.

The Canvas Dragged event will happen over and over again very rapidly while the user drags a finger on the screen. Each time that Dragged event block is called, it will draw a small line between the previous location (prevX, prevY) of the finger to the new location (currentX, currentY). Mouse over the parameters of the Canvas1.Dragged block to pull out the get blocks that you need. (Mouse over them, don't click on them.)



Test it out!

Go to your connected device and drag your finger around the screen. Do you see a line?



Great work! Now extend this app

Here are some ideas for extending this app. You can probably think of many more!
- Change the color of the ink (and let the user pick from a selection of colors). See Paint Pot tutorial.
- Change the background to a photograph or picture.
- Let the user draw dots as well as lines (hint: Use DrawCircle block).
- Add a button that turns on the camera and lets the user take a picture and then doodle on it.