TalkToMe: Your first App Inventor app

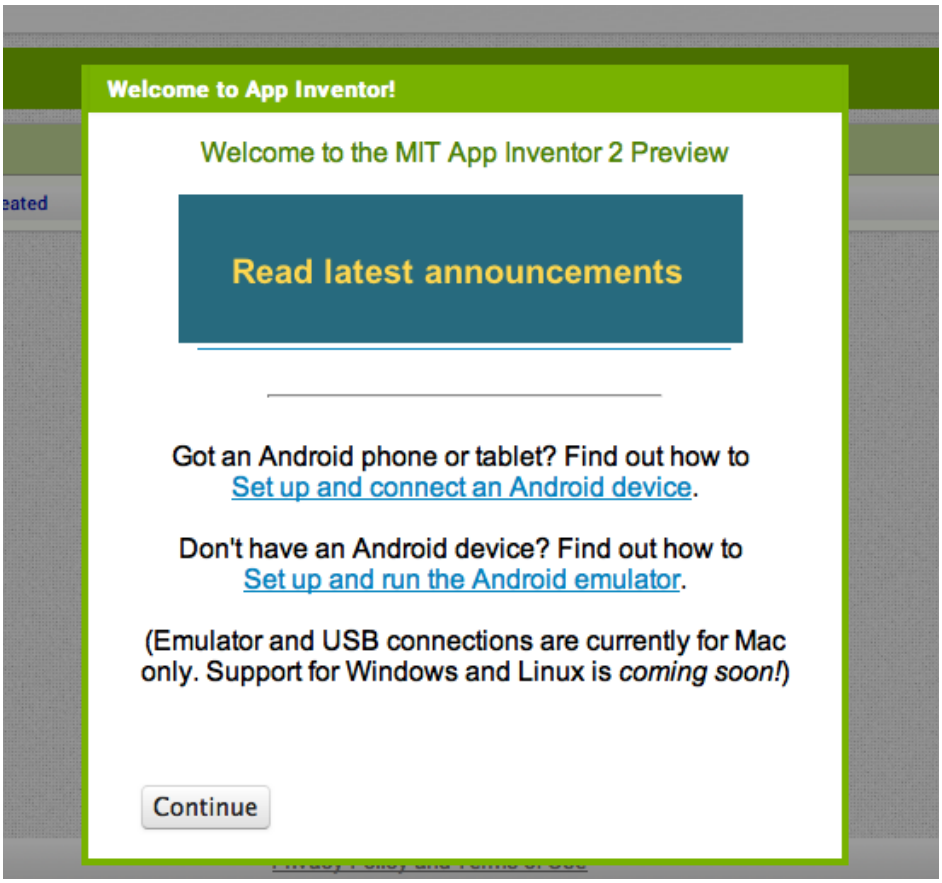
This tutorial will guide you through making a talking app that can save phrases on demand.

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

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

Use an existing gmail account or school-based google account to log in to ai2.appinventor.mit.edu

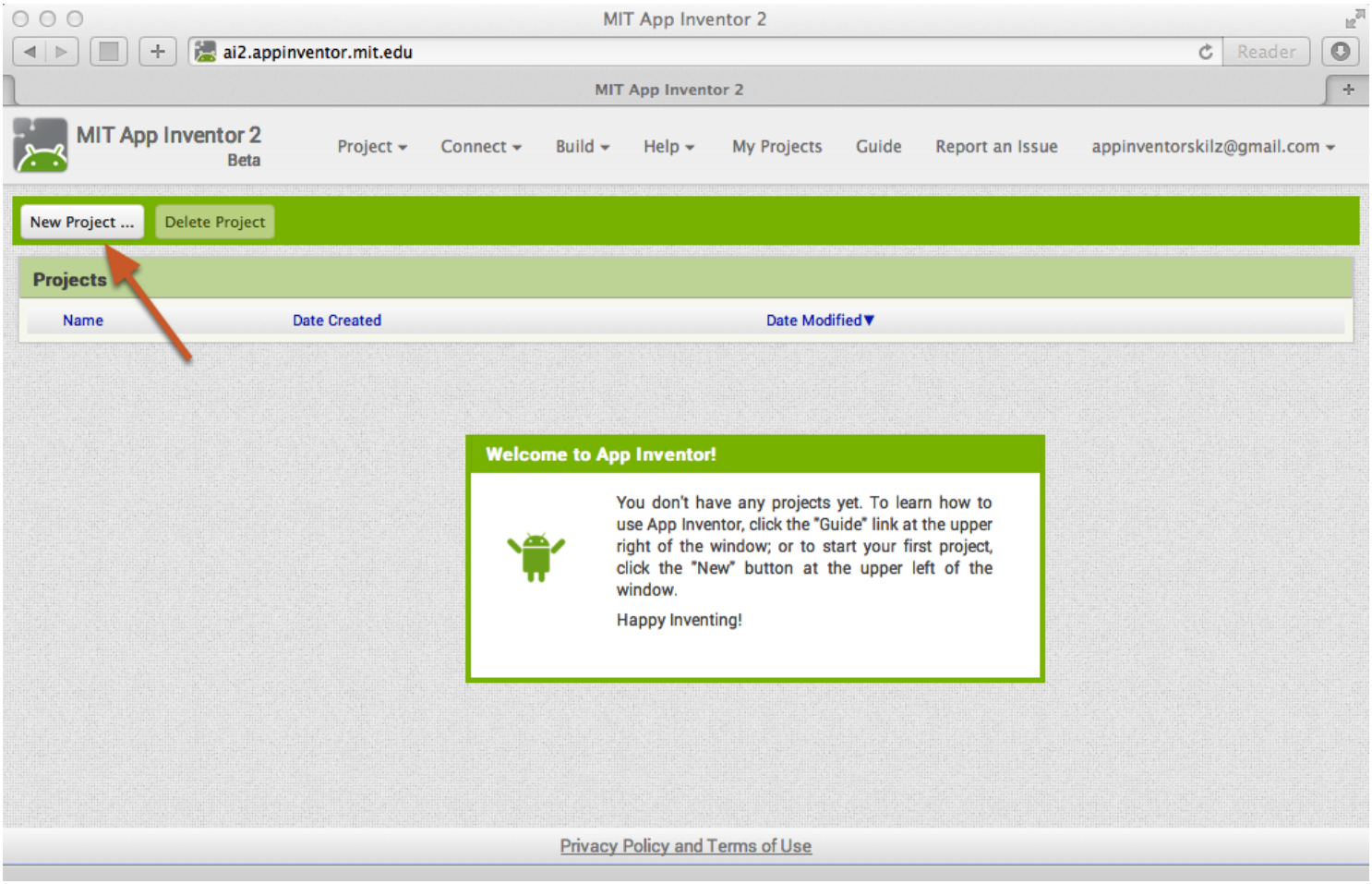
Click "Continue" to dismiss the splash screen.



Start a new project.

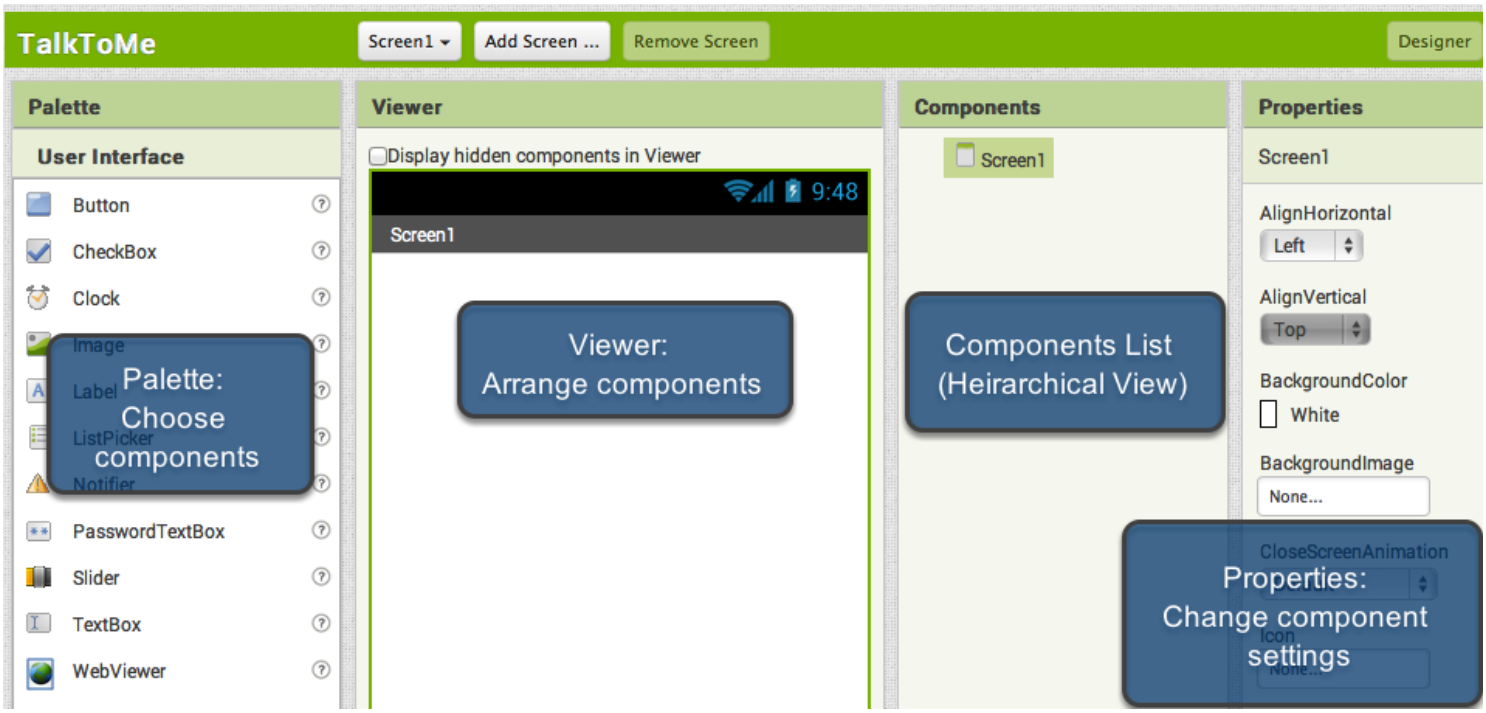
Name the project "TalkToMe" (no spaces!)

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



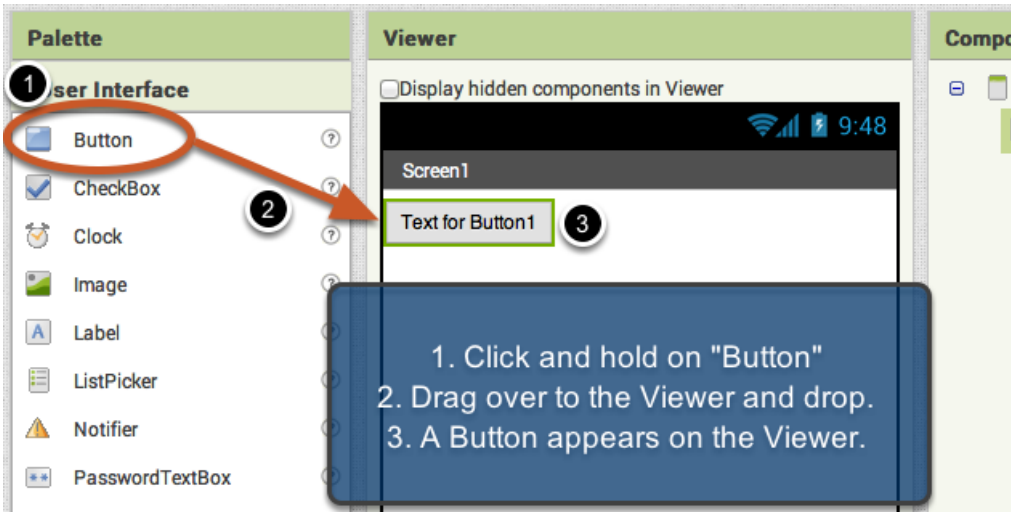
You are now in the Designer, where you lay out the "user interface" of your app.

The Design Window, or simply "Designer" is where you lay out the look and feel of your app, and specify what components 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.



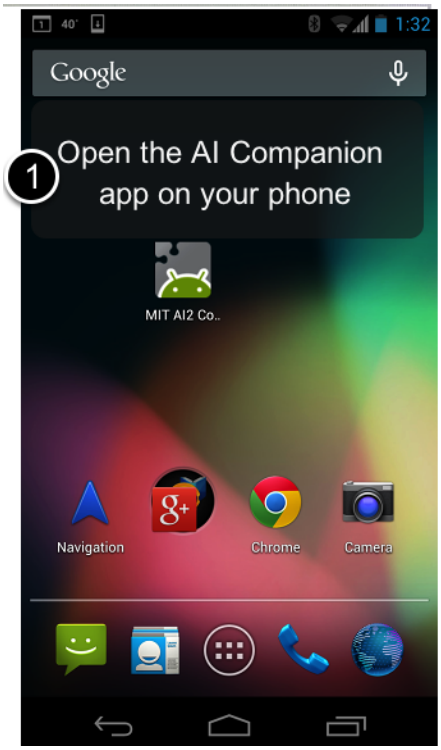
Add a Button

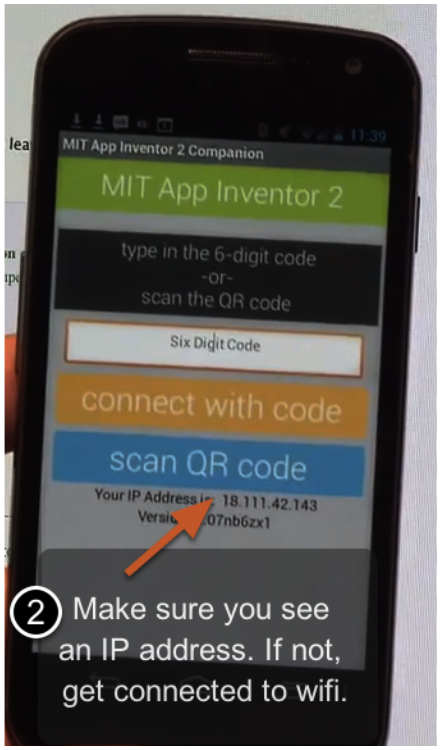
Our project needs a button. ***Click and hold*** on the word "Button" in the palette. ***Drag*** your mouse over to the Viewer. ***Drop*** the button and a new button will appear on the Viewer.

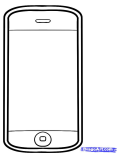


Start the AICompanion on your device

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.

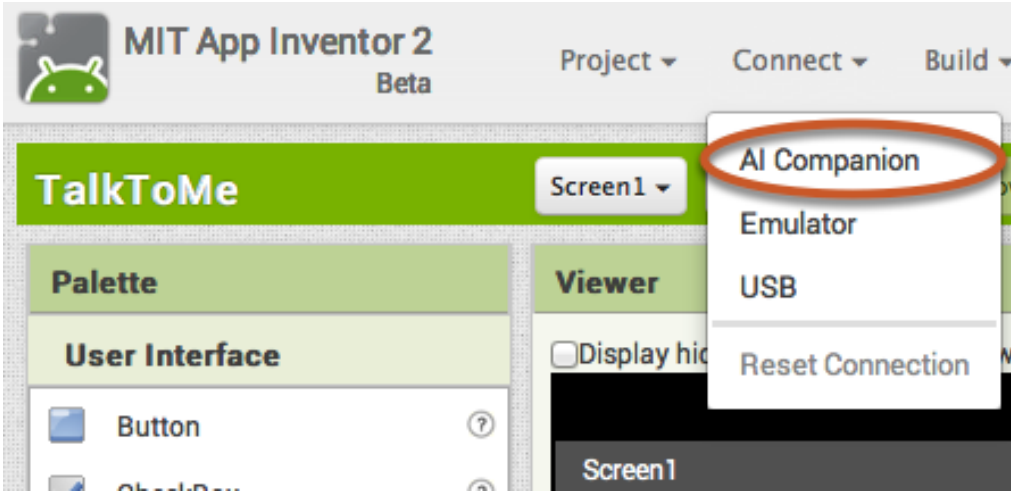






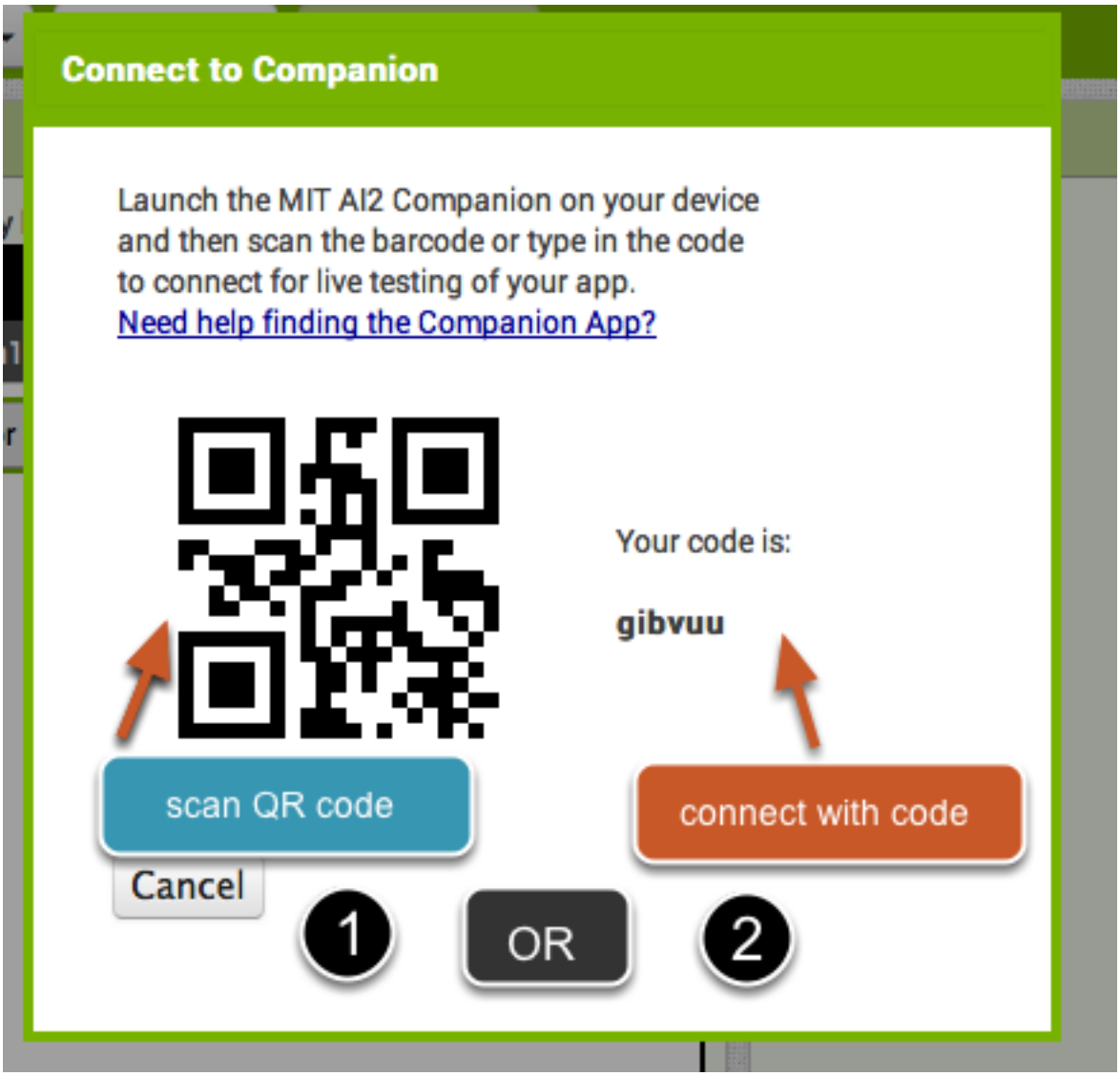
Connect App Inventor to your phone for live testing

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.**



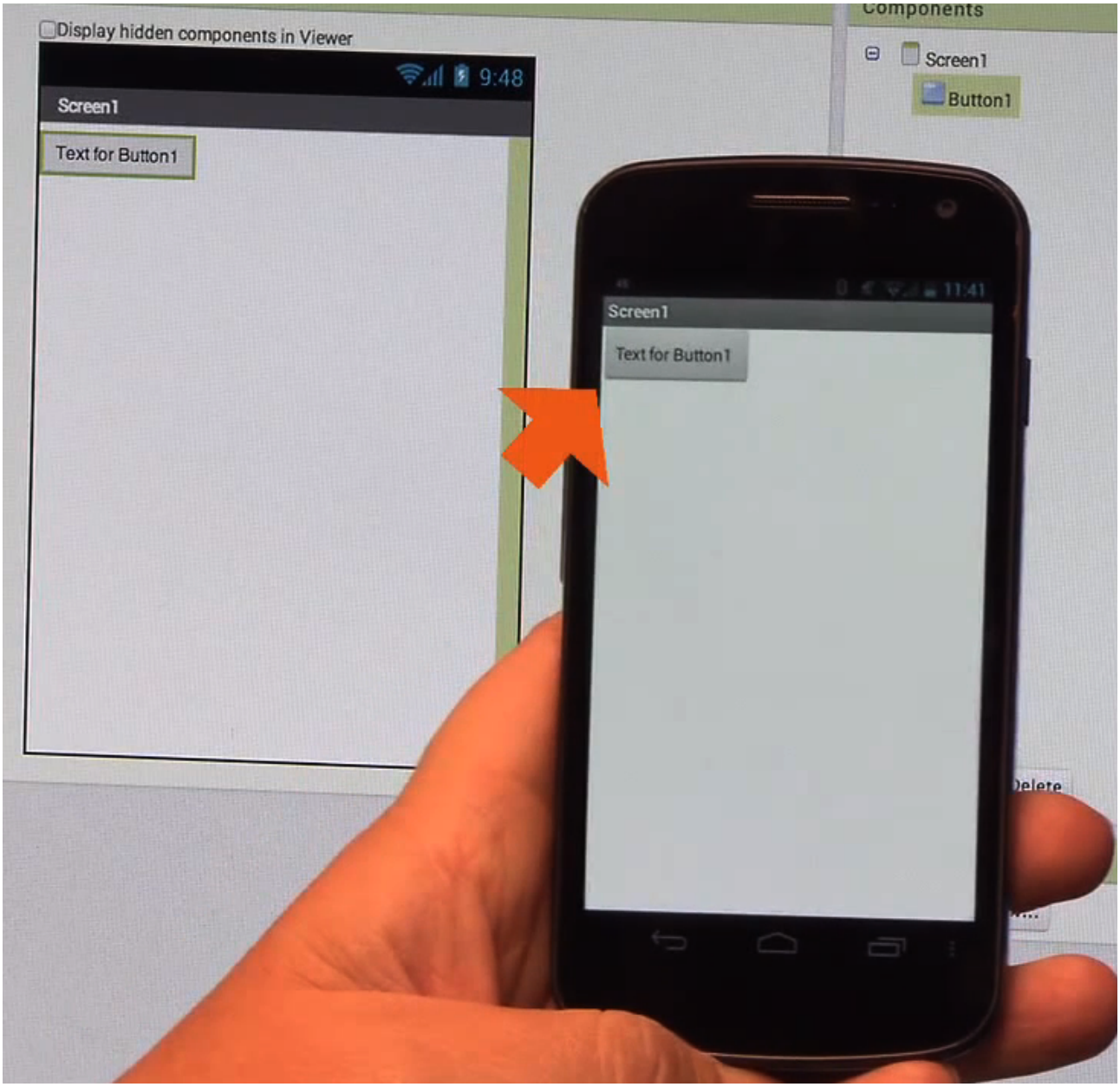
Get the Connection Code from App Inventor and scan or type it into your Companion App

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).



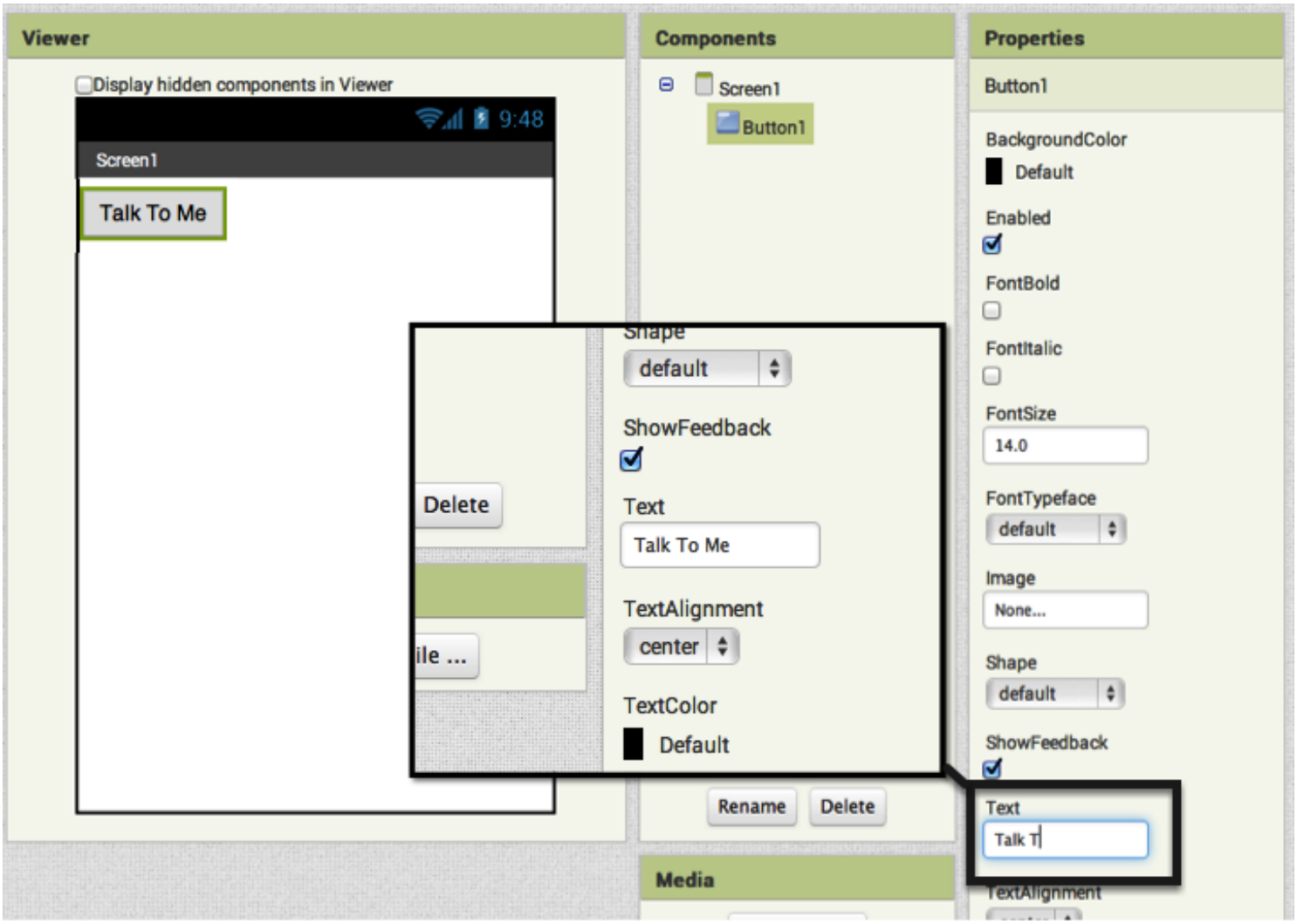
See your app on the connected device

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

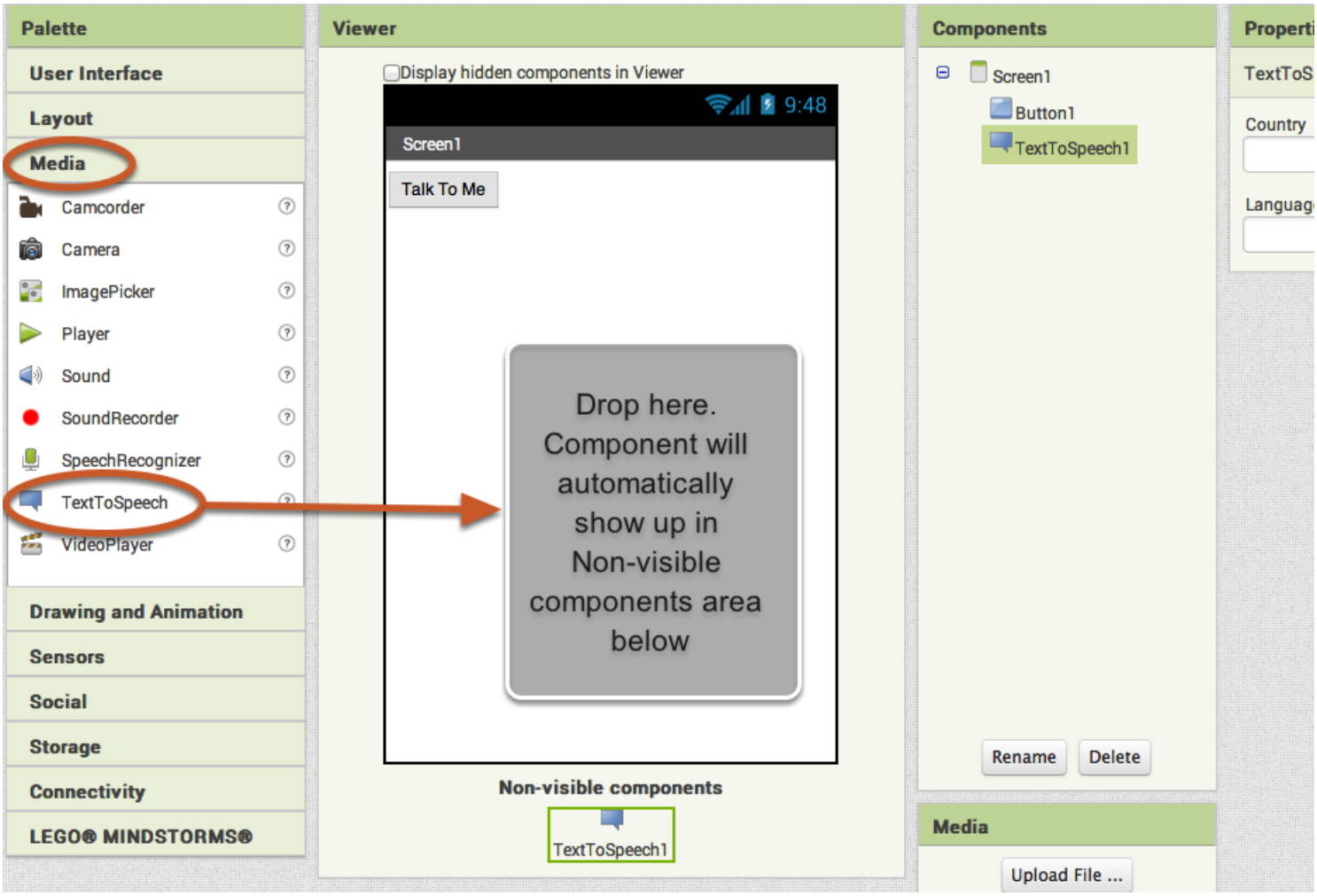
On the properties pane, change the text for the Button. Select the text "Text for Button 1", delete it and type in "Talk To Me". Notice that the text on your app's button changes right away.



technovation

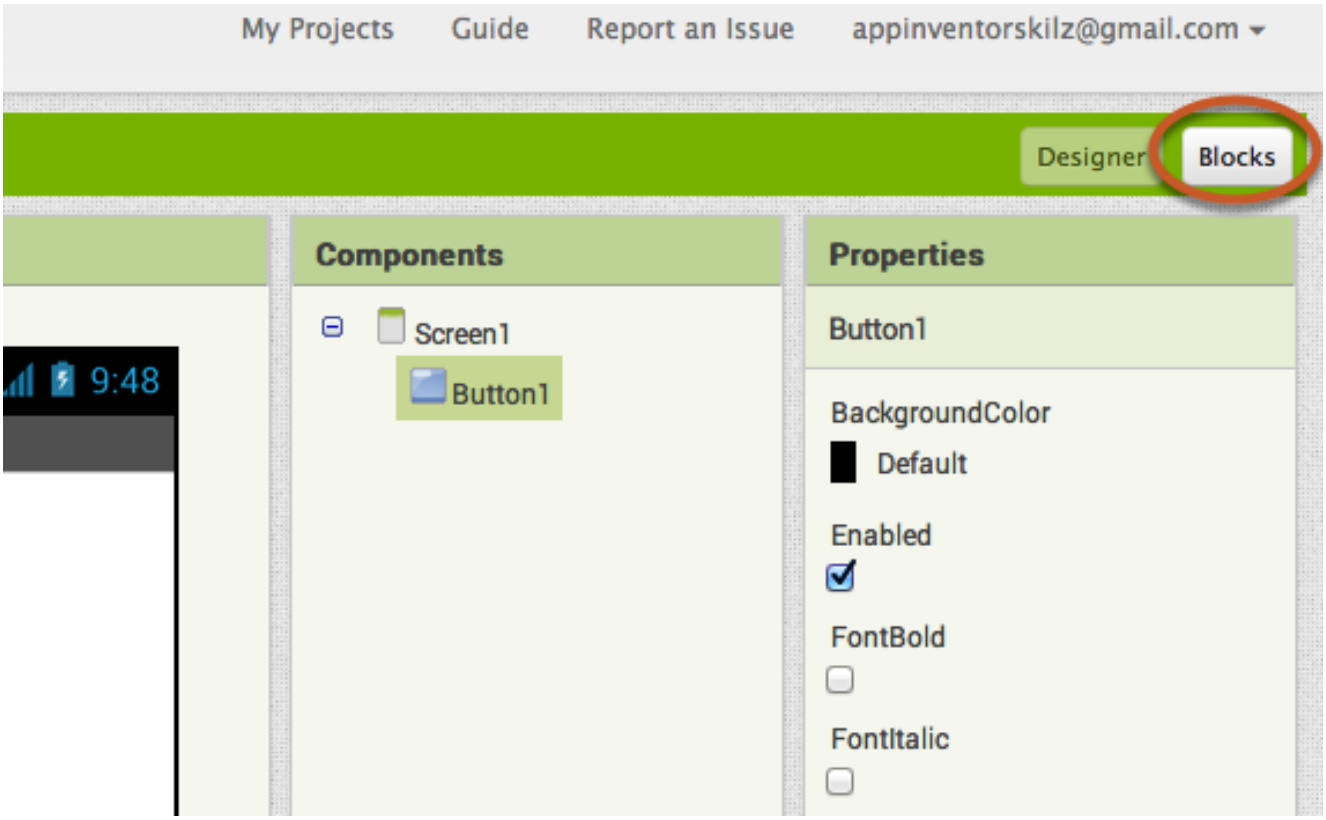
Add a Text-to-Speech component to your app

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.



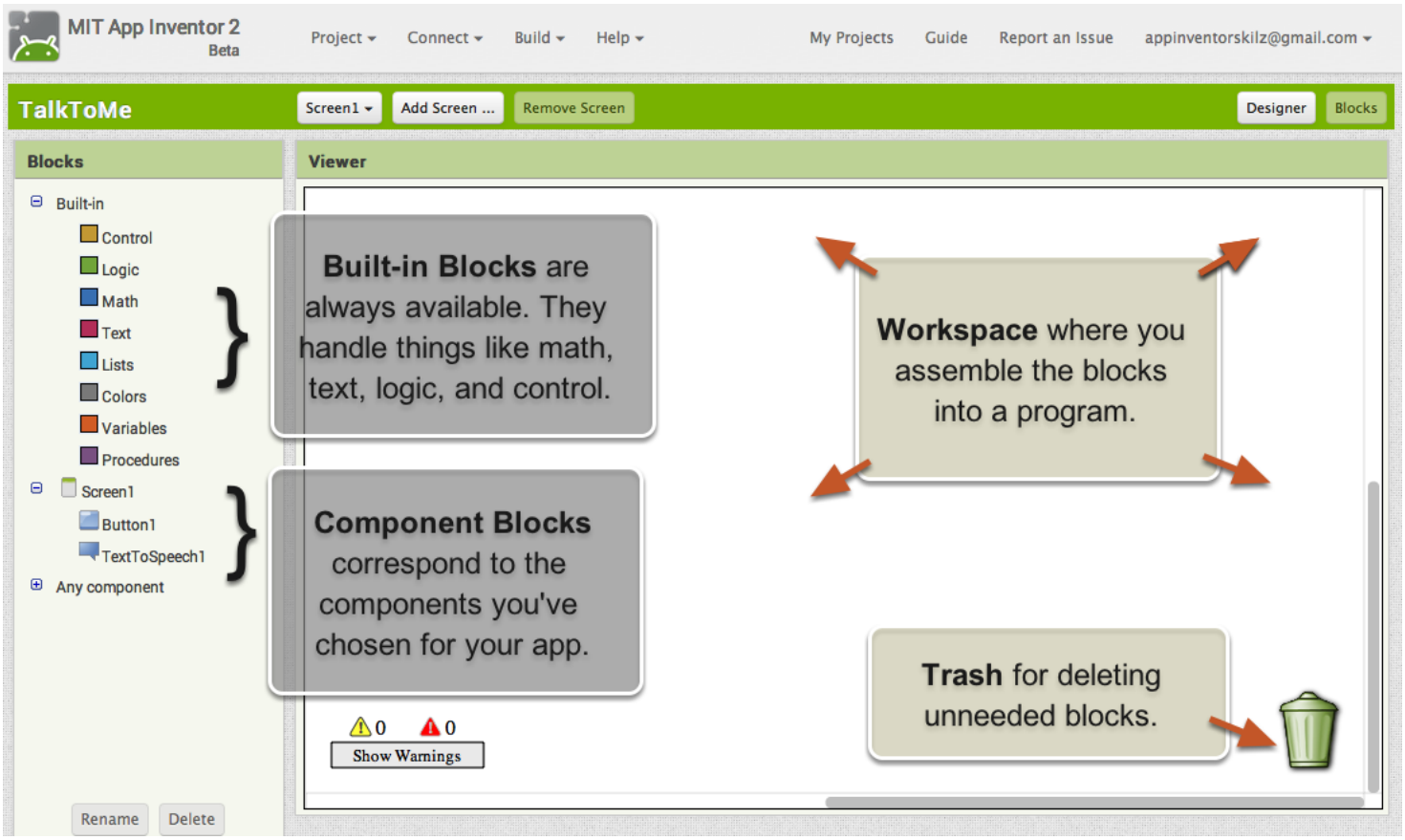
Switch over to the Blocks Editor

It's time to tell your app what to do! Click "Blocks" to move over to the Blocks Editor. So far, we've only decided how the app should look. The Blocks Editor will allow us to tell the app how it should behave. Think of the Designer and Blocks buttons like tabs -- you use them to move back and forth between the two areas of App Inventor.



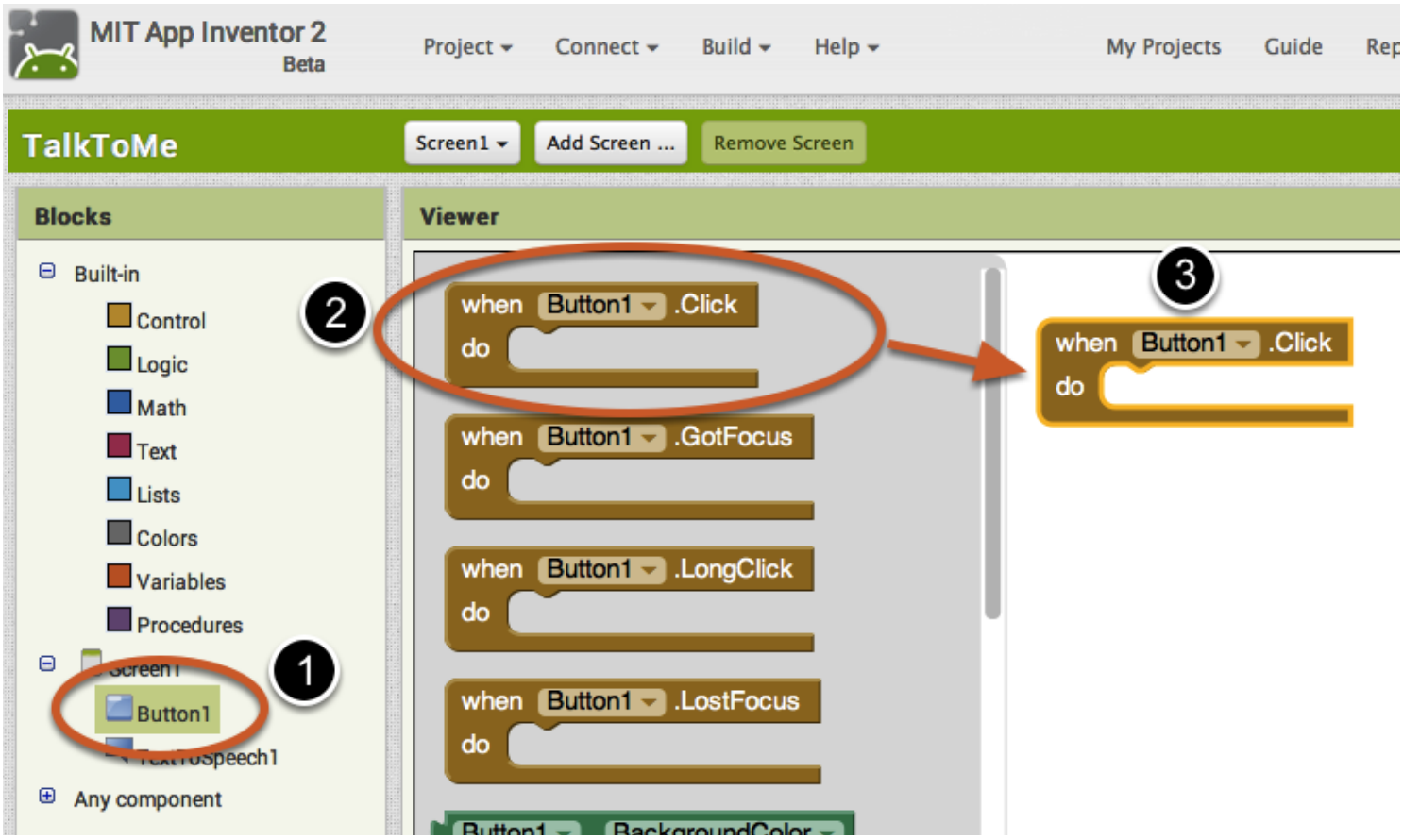
The Blocks Editor

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.*



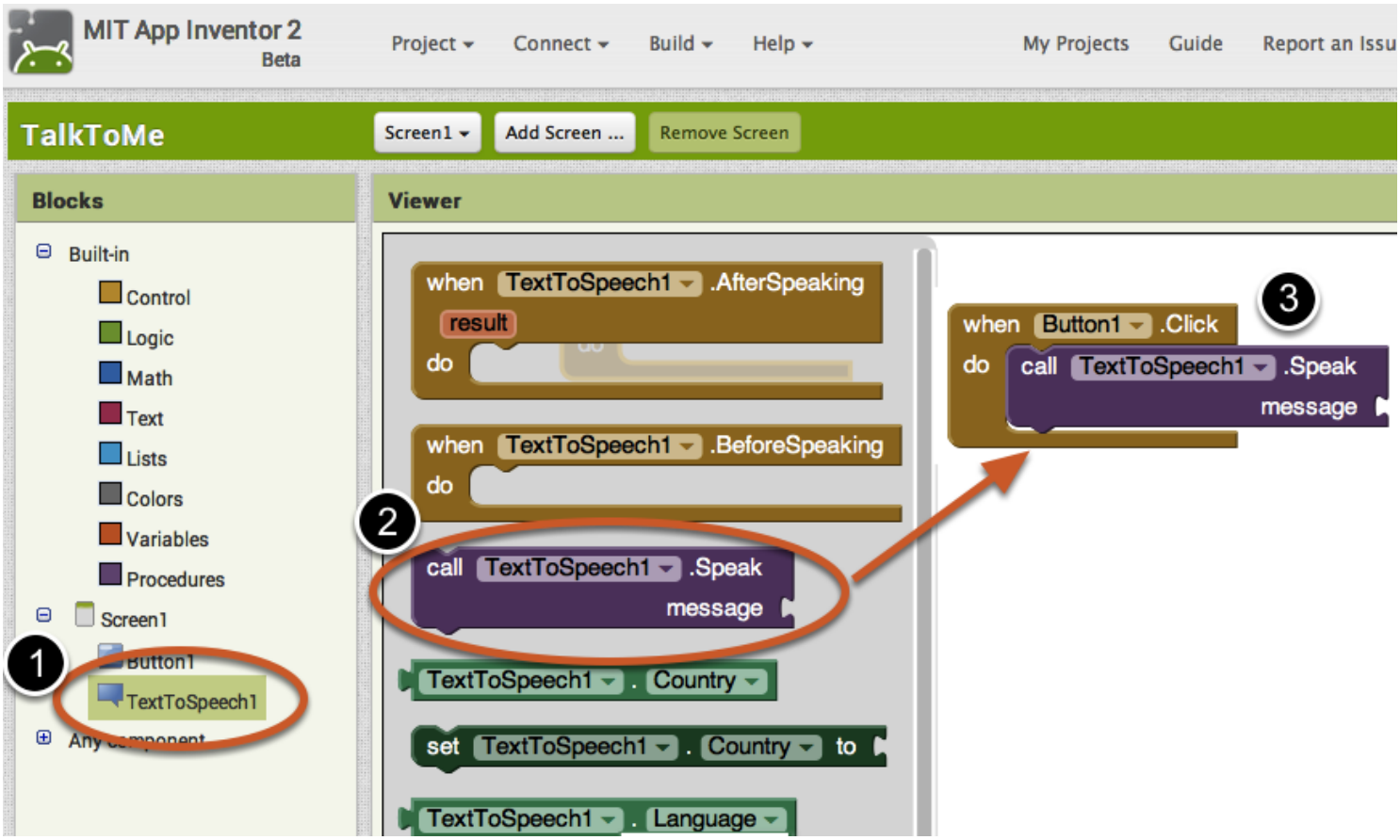
Make a button click event

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". All event handlers are a tan/brown color in AI2. Event handlers are triggered when an event is initiated by the user (for example, clicking a button).



Program the TextToSpeech action

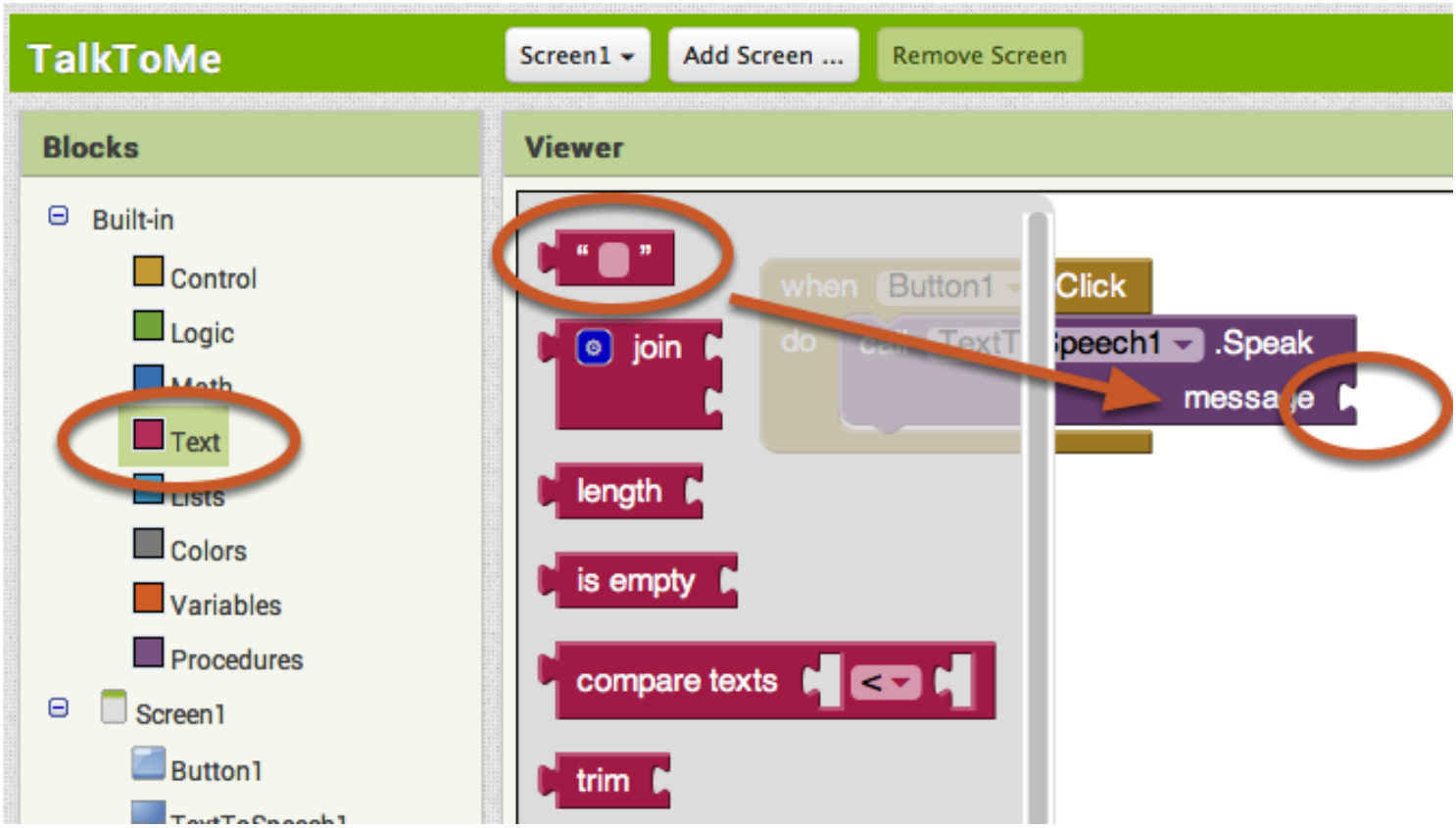
Click on the TextToSpeech drawer. Drag the ***call TextToSpeech1.Speak*** block over to the workspace and drop it there. This purple block calls a “procedure” in App Inventor. This procedure will make the phone speak. Because it is inside the Button.Click, it will run when the button on your app is clicked. Procedures must be called on an event (for example, the button click event).



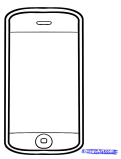
technovation

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".



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!

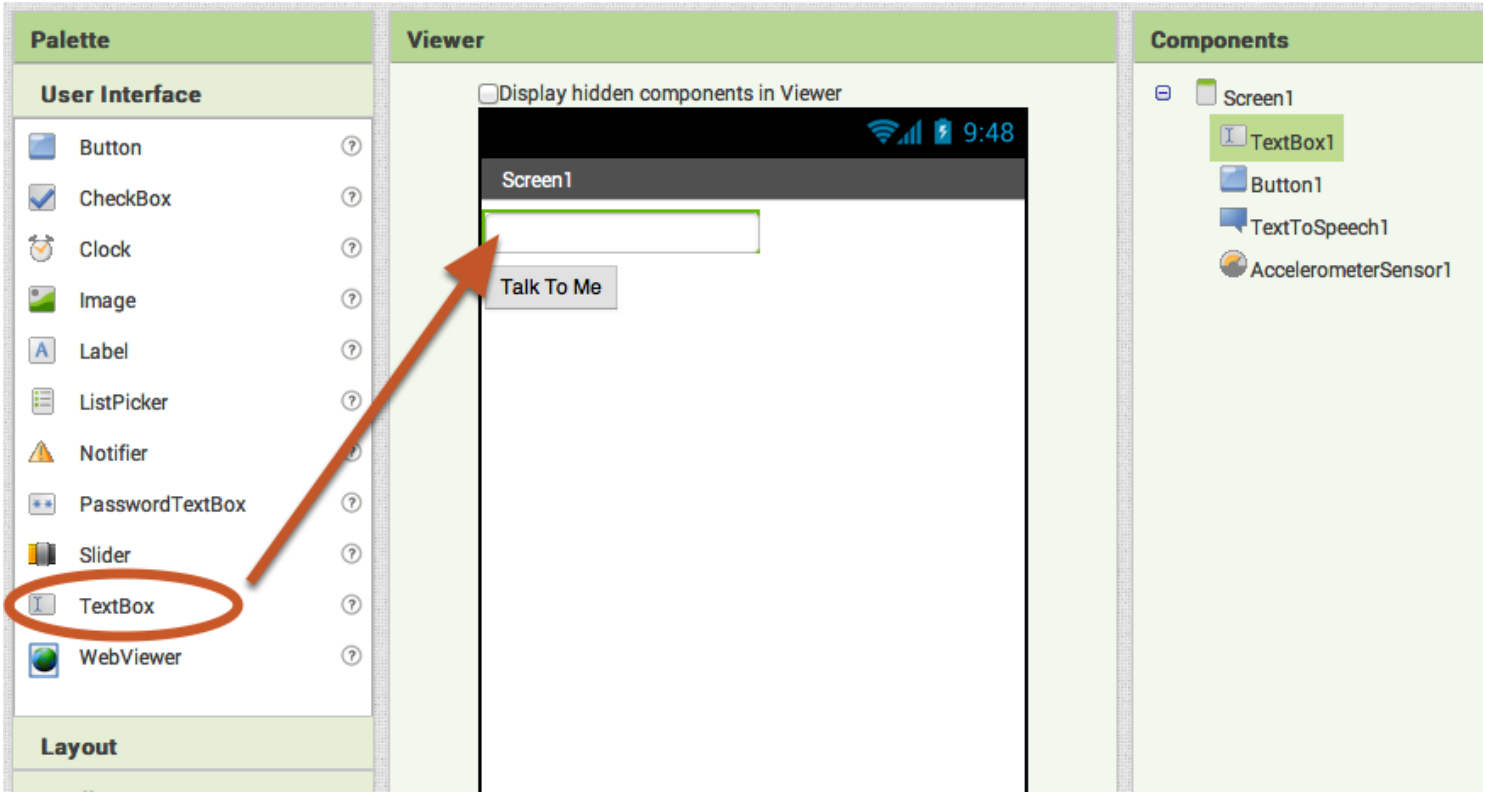
Go to your connected device and click the button. Make sure your volume is up! You should hear the phone speak the phrase out loud.

Go back to the Designer Tab

Now let's make the phone say whatever you want! Let's go back to the designer tab to do so. Click "Designer" in the upper right.

Add a Text Box to your user interface.

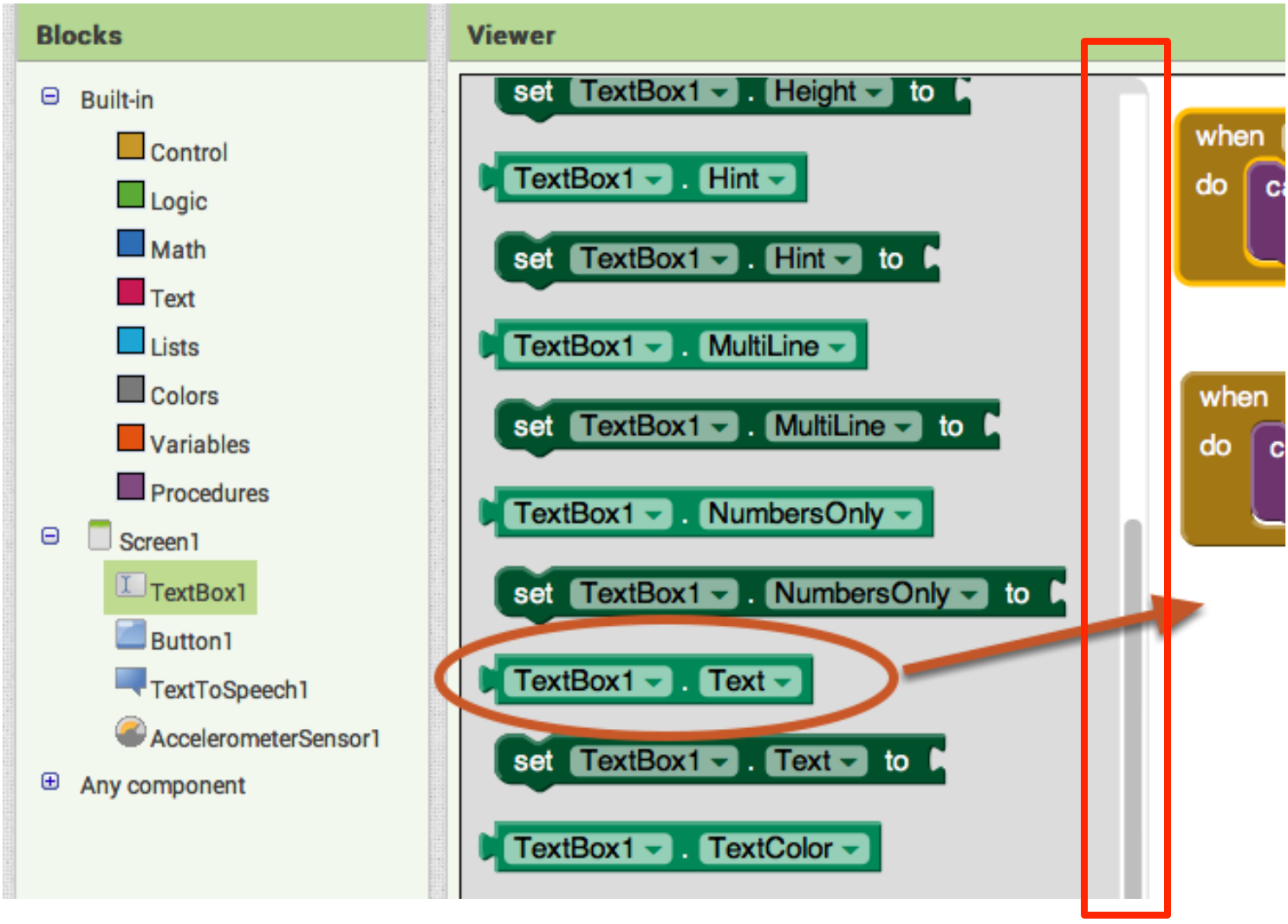
From the User Interface drawer, drag out a TextBox and put it above the Button that is already on the screen.



Get the text that is typed into the TextBox.

Get the text property of the TextBox1. The light green blocks in the TextBox1 drawer are the "getters" and the dark green blocks are "setters" for the TextBox1 component. This is true for all components in App Inventor. These blocks are called “getters” and “setters” because you can “get” or “set” the value that’s stored for that part of the component using these blocks.

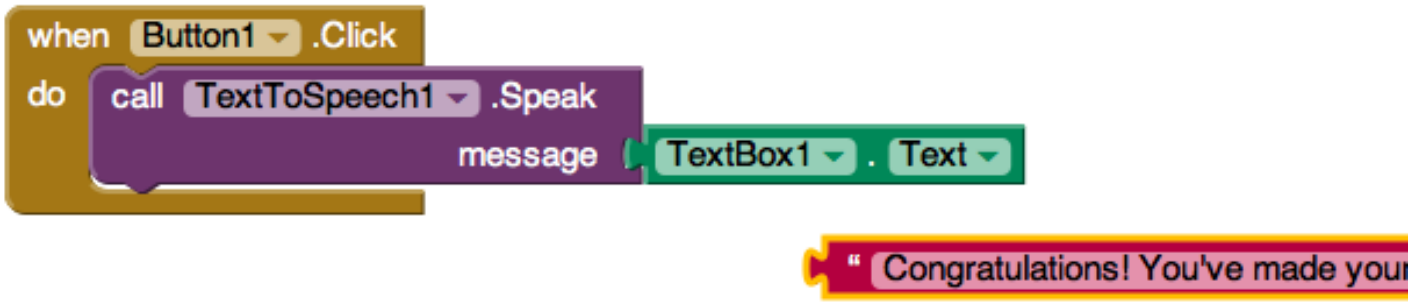
You want your app to speak out loud whatever is currently in the TextBox1.Text block (i.e. whatever is typed into the text box). Drag out the ***TextBox1.Text*** getter block.



technovation

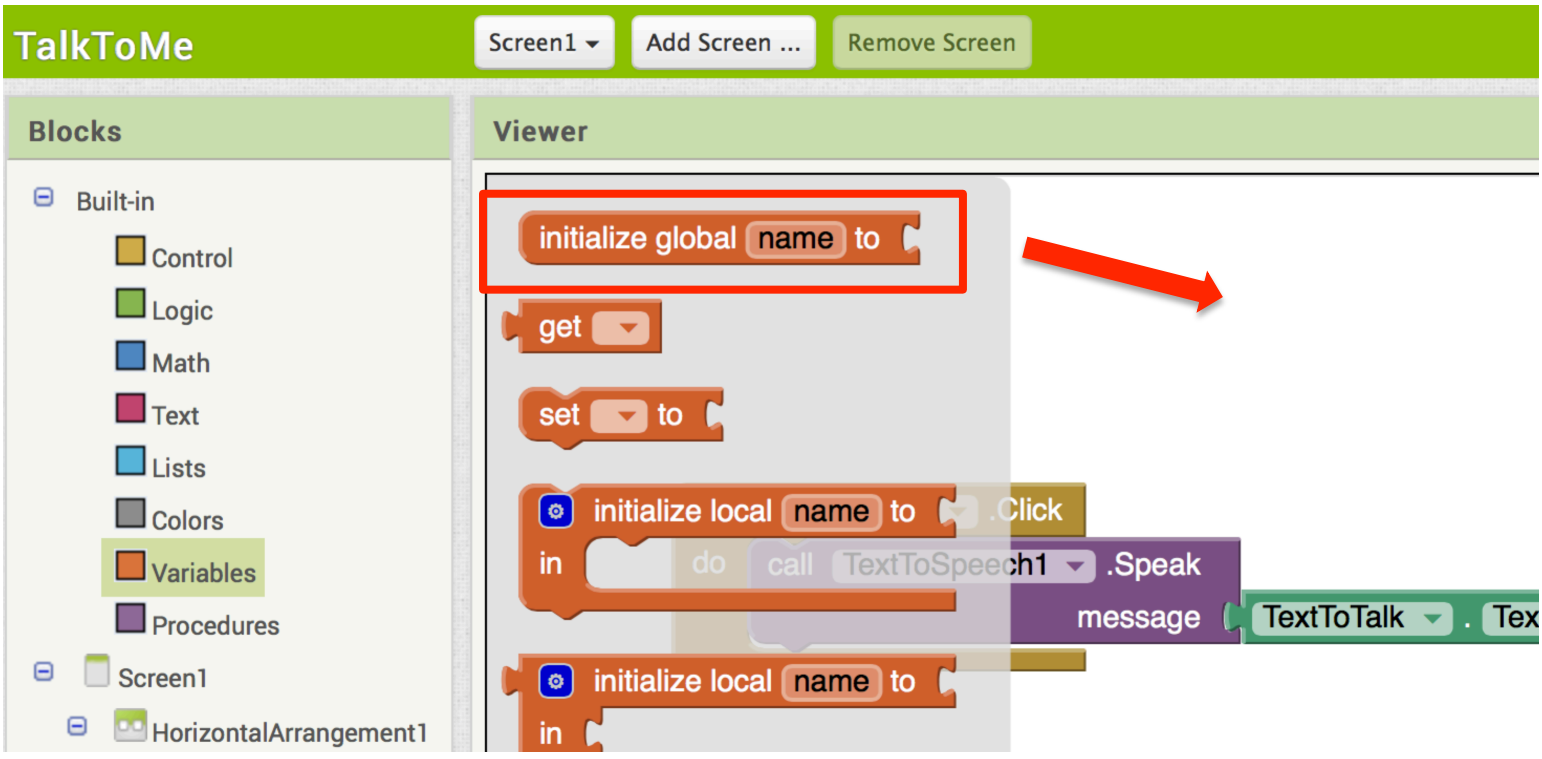
Set the Button Click event to speak the text that is in the Text Box.

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.



Let's save the Text as a Variable.

The Text that the app will speak is now variable, or changing with usage of the app. Therefore, we can save the Text in a variable. We can name the variable by clicking on the "name" part of the block after dragging it out. Variables should always be named in a meaningful way so that if you were to come back to this app in 3 months, it will be easy to remember what this variable is supposed to be tracking. Let's name this variable "textToSpeak".



technovation

Every variable has to have a value to begin with. Since this variable will be storing text, we'll start it off with blank text.

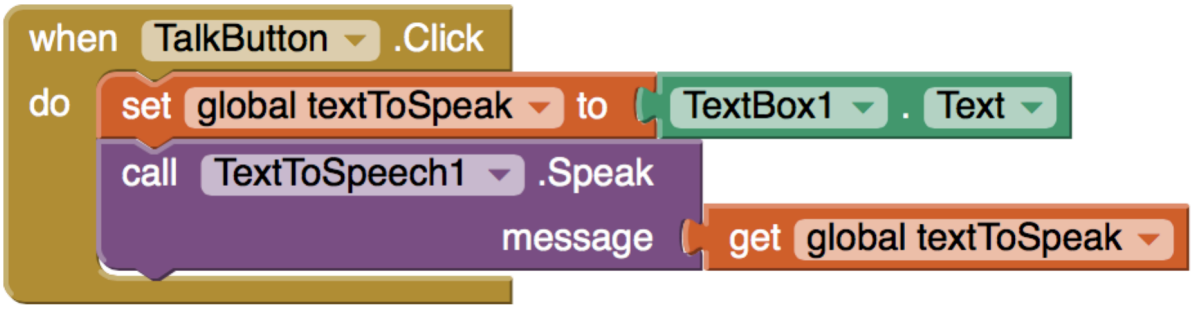
So how do we use the variable? The original assignment for the message was as follows:

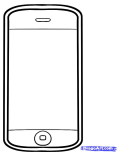
Message → TextBox1.Text

Since we'll be using the variable textToSpeak for the message now, we need to replace TextBox1.Text with the variable and assign the TextBox1.Text value to the variable. Every time the button is clicked, the variable value is updated, and the correct message is passed to TextToSpeech.

textToSpeak → TextBox1.Text

Message → textToSpeak





Test your app!

The functionality should remain the same. What happens if you switch the order of the "set" and "call" blocks? Try it out and see! Describe what happens and why.

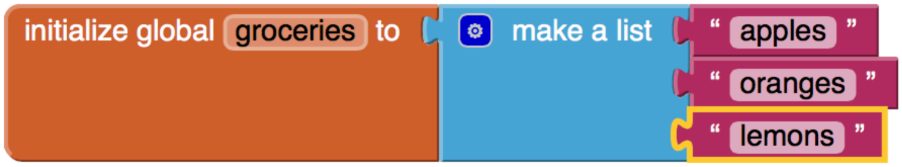
Save all the phrases you enter.

How can you save all the phrases that we're telling the app to speak? We need a way to "store" the text that we type into the TextBox. Here's how you can do that with a list.

Lists

Lists are exactly what they sound like they should be. They keep track of certain items. For example, you may track the grocery items you need to buy in a list. Let's call this list "**groceries"**.

1. apples
2. oranges
3. lemons



Each of these items is "indexed," meaning you can ask to get item number 1 from this list and you will get "apples." If you ask for item number 2, you'll get "oranges," and so on.

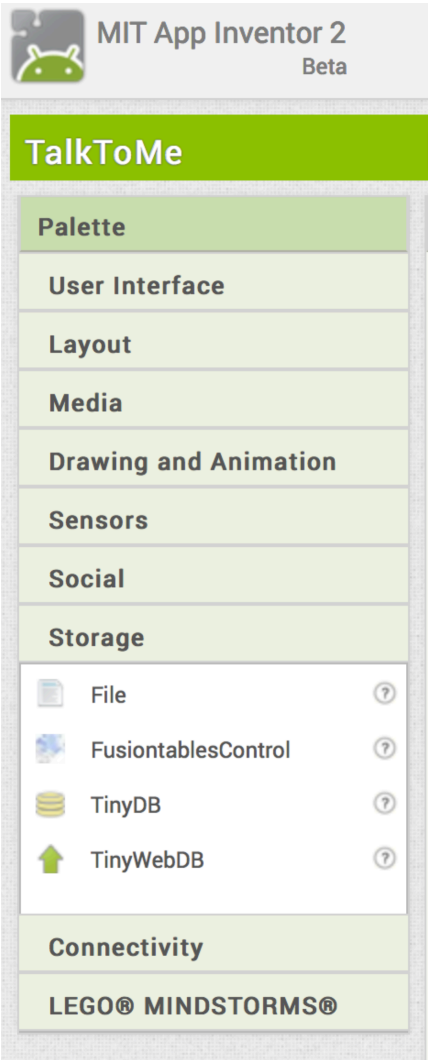
In App Inventor, every time you add an item to the list, you are "appending" to the list, or putting another item on the end of the list. For example, if I wanted to add "bananas" to the list, the final list would look like this:

1. apples
2. oranges
3. lemons
4. bananas

Adding items to the list does so without discrimination. This means that if an item already exists, you can add a duplicate. If I wanted to add "oranges" to the list, I could do so and end up with the following:

1. apples
2. oranges
3. lemons
4. bananas
5. oranges

As a result, asking for item number 2 will return "oranges," but asking for item number 5 will also return "oranges." In order to avoid this, you can first check if this item "is in list?" This block will return true if the item already exists in this list, and false if it doesn't.



HINT: If you want more information about a component, you can click on the question mark next to the component in the palette.

Okay! Let's save all the phrases in your app. Implement the necessary blocks to save all your phrases.

Test your app! Has the functionality changed at all? Can you tell if the values are being saved?

_____

_____

_____

_____

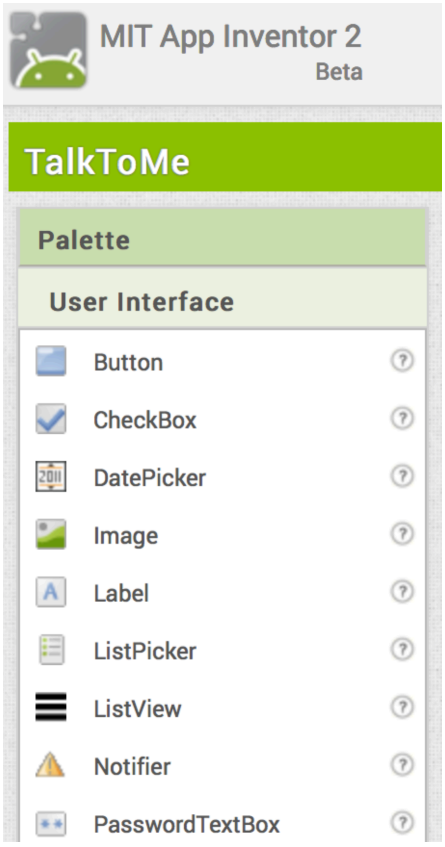
technovation

Let's make sure our saving actually works by adding a viewing feature.

There are a couple of different ways that we can view what's in the list: Labels and ListViews. Let's explore the ListView option.

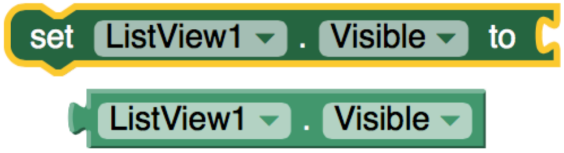
The ListView needs to be added from the designer from the "User Interface" palette since this allows the user to view something.

The ListView component has an "Elements" property. This is how the ListView component will store a list (as described on page 19) of items that it will be displaying. Continuing with the grocery list example, if we **set** ListView1.Elements to "groceries," the list defined there will be displayed by this component.



How will your TalkToMe blocks for ListView differ from the ones above?

The ListView component also has a property called "Visible." Setting this property to be "false" allows the list to be hidden from the user (not visible on the screen). The ListView component will be Visible (set to true) by default if you don't modify it.



You've already added a list to your blocks to save all your phrases. How would you use that with the ListView component so you can see that all your phrases are being saved correctly?

technovation

Let’s add a selective save feature to our app.

There might be certain phrases that you want the phone to say multiple times, but it can be a hassle to have to re-type the phrase every time you want to use it. We’ve already added a way to save all of the phrases that have been entered, but what if you only want to save some of them? **How would you design this save feature?**

Here are some components that may be useful:

Button	ListView	ListPicker	TinyDB

Here are some Block groups that may be useful:

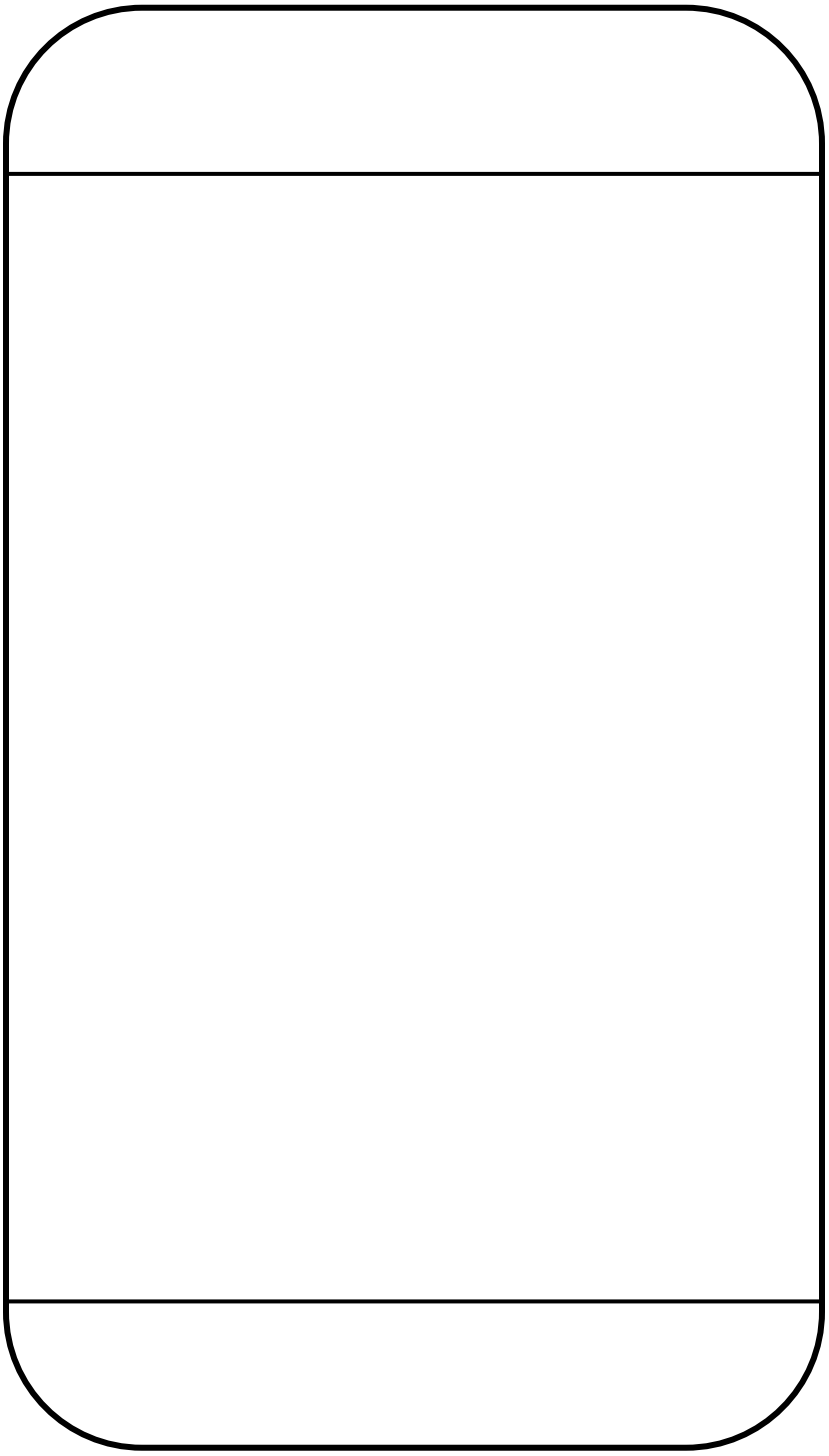
Lists	Procedures	Logic (true or false)

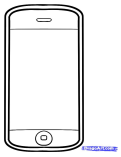
Start by listing any additional components and blocks you may use.

_____

_____

Next, sketch what your screen will look like with the additional component(s).





Now code and test it! How are you verifying that you are only saving the phrases you want?

_____

_____

_____

What happens if you try to save the same phrase now? How can you prevent this from happening? (hint: try looking around in the "Lists" blocks and the "Control" blocks)

_____

_____

_____

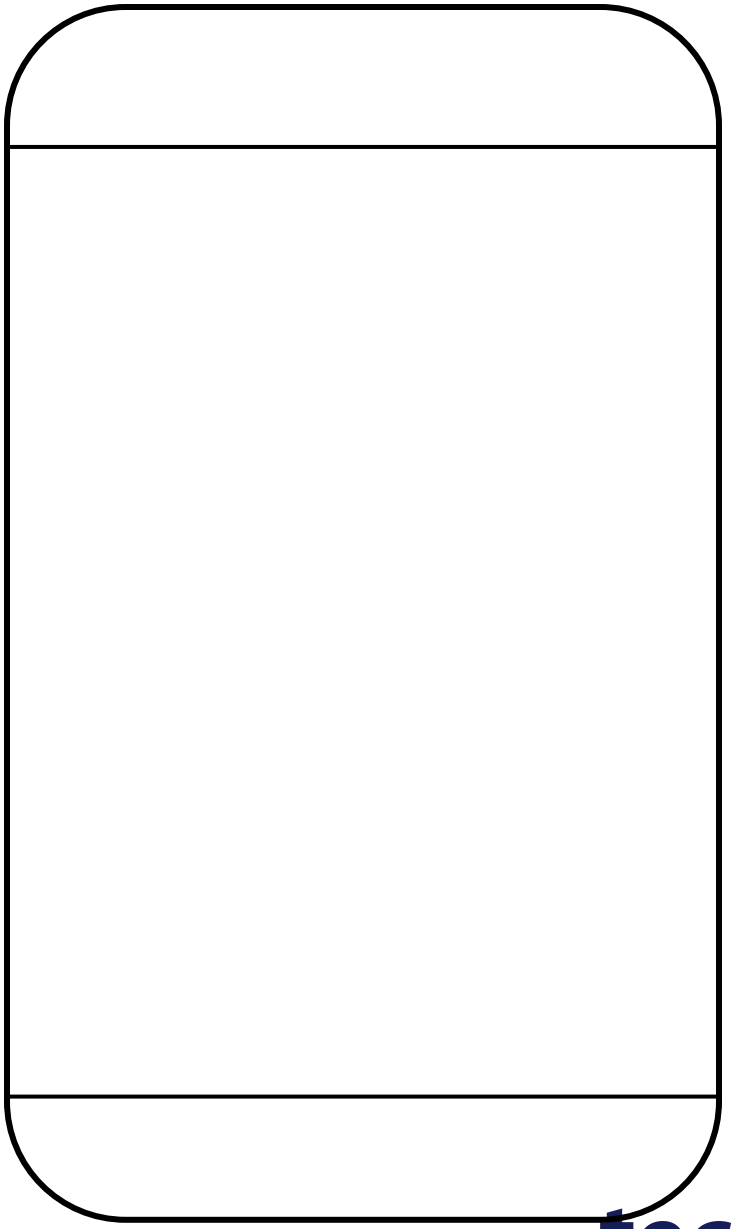
How useful is this app to you right now? How can you use some other components in App Inventor?

_____

_____

Let's try adding the YandexTranslator component. How would your app change if you added this? (This is up to you; you're the designer!)

Prototype (sketch the screen interface) here first, and remember you can get more information about the component by clicking the question mark next to it in the palette.



Takeaways from this tutorial.

By the end of this tutorial, you should feel comfortable:

- Exploring the App Inventor design and blocks tabs to build what you need.
- Testing your features: It's great to build out a feature, but even more important to make sure they work correctly. Always think about how you can double-check that you're doing what you think you're doing.
- Incremental progress: Features are often built one at a time. This way, if something in your project breaks, you have a better idea of what can be the problem.