ITEC 399
MOBILE APPLICATION DEVELOPMENT

CHAPTER 2 –APP INVENTOR

Subject Teacher: Mobina Beheshti

OBJECTIVES

- Background to App Inventor
- App Inventor Architecture
- Advantages and disadvantages of App Inventor
- App Inventor Basics
- Building Android Apps with App Inventor (Designer, Block Editor and Emulator)

APP INVENTOR

App Inventor

- In 2009 Google deployed the first App Inventor pilots at a few universities in the United States for teaching their students mobile applications development, including those with zero programming experience.
- App Inventor allows development of applications for Android phones using online web servers, web browsers and either a connected phone or emulator (Figure 1). The App Inventor servers store developers work and helps them keep track of their projects.

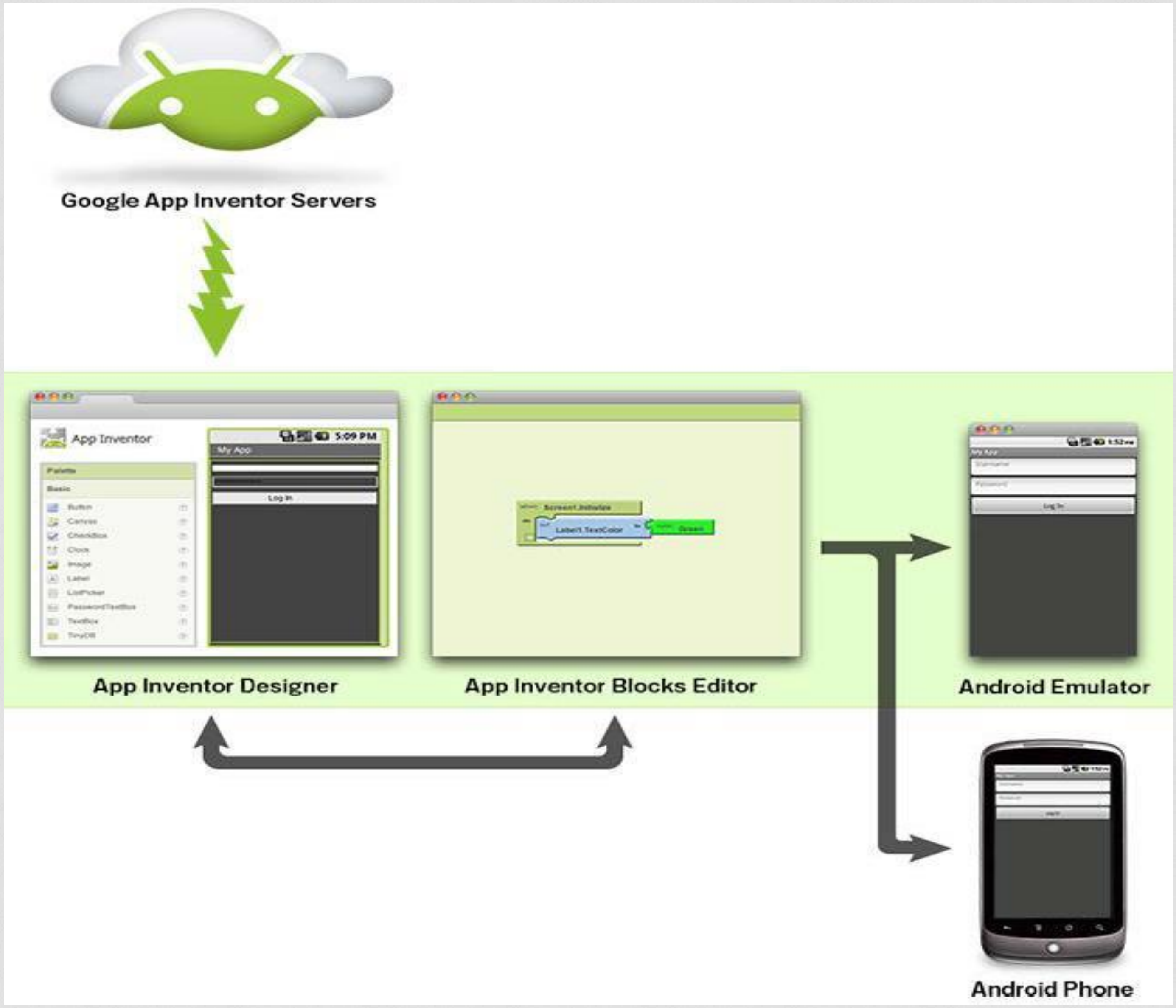


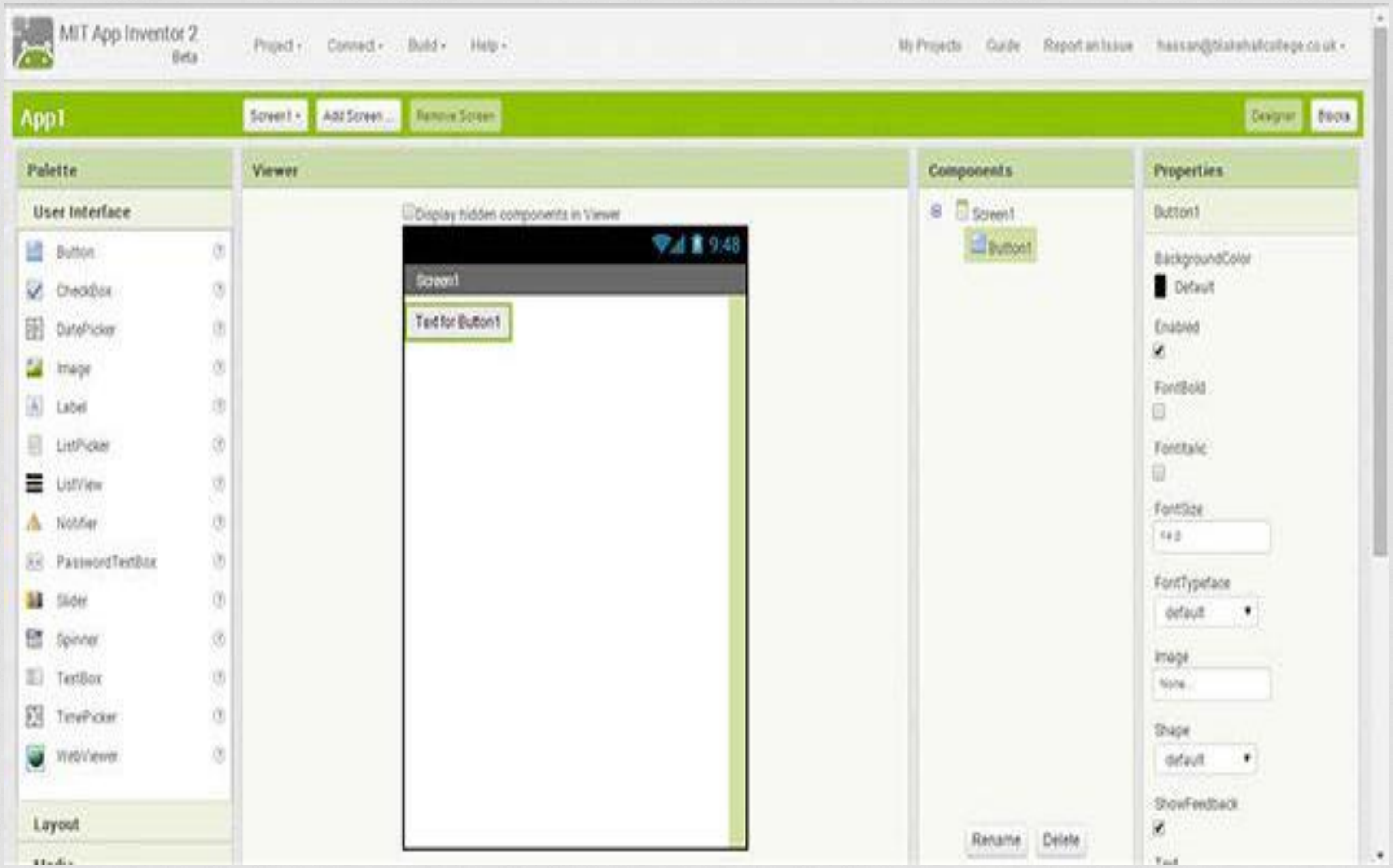
Figure 1: App Inventor Designer, Editor and Emulator

APP INVENTOR

- App Inventor is an open source web-based IDE and is maintained by the Massachusetts Institute of Technology (MIT), which facilitates an online tool to create apps for mobile devices using Android platforms, free of charge.

- App Inventor uses a simple graphical user interface designer (excellent for beginners) that allows users to drag-and-drop visual objects and code block editors to create apps that can run on Android devices.

- The online IDE of App Inventor (Figure 2) may sound odd when compared with the traditional and even mainstream IDEs, as instead of writing code, developers can assemble their apps from prefabricated blocks of code similar to building a map with pieces of puzzle.



Button1

BackgroundColor
Default

Enabled
✓

FontBold
☐

FontItalic
☐

FontSize
14.0

FontTypeface
default

Image
None...

Shape
default

ShowFeedback
✓

APP INVENTOR FEATURES

- App Inventor allows people to familiarize themselves with a mobile applications development environment.
- At the same time they can use it to create genuine, professional apps.
- With App Inventor developers can design how an app looks and then design its behavior.
- The app itself is built from a set of event handlers that make an app behave as a developer wishes.
- Developers build these event handlers by assembling and configuring blocks representing events, functions, conditional branches, repeat loops, web calls, database operations, and more.

APP INVENTOR'S CAPABILITIES

App Inventor's capabilities include:

- Access to most of the phone's functionality: phone calls, SMS texting, sensors for location, orientation, and acceleration, text-to-speech and speech recognition, sound, video.
- The ability to invoke other apps, with the ActivityStarter component.
- Programming *control* just as with a textual language. There are blocks for conditionals (if, ifelse), foreach, and while, and a fairly comprehensive list of math and logic blocks.
- Database access, both on the device and on the web. So you can save data persistently, and with a web database share data amongst phones.
- Access to web information sources (APIs)-- you can bring in data from Facebook, Amazon, etc.

APP INVENTOR HAS THE FOLLOWING LIMITATIONS

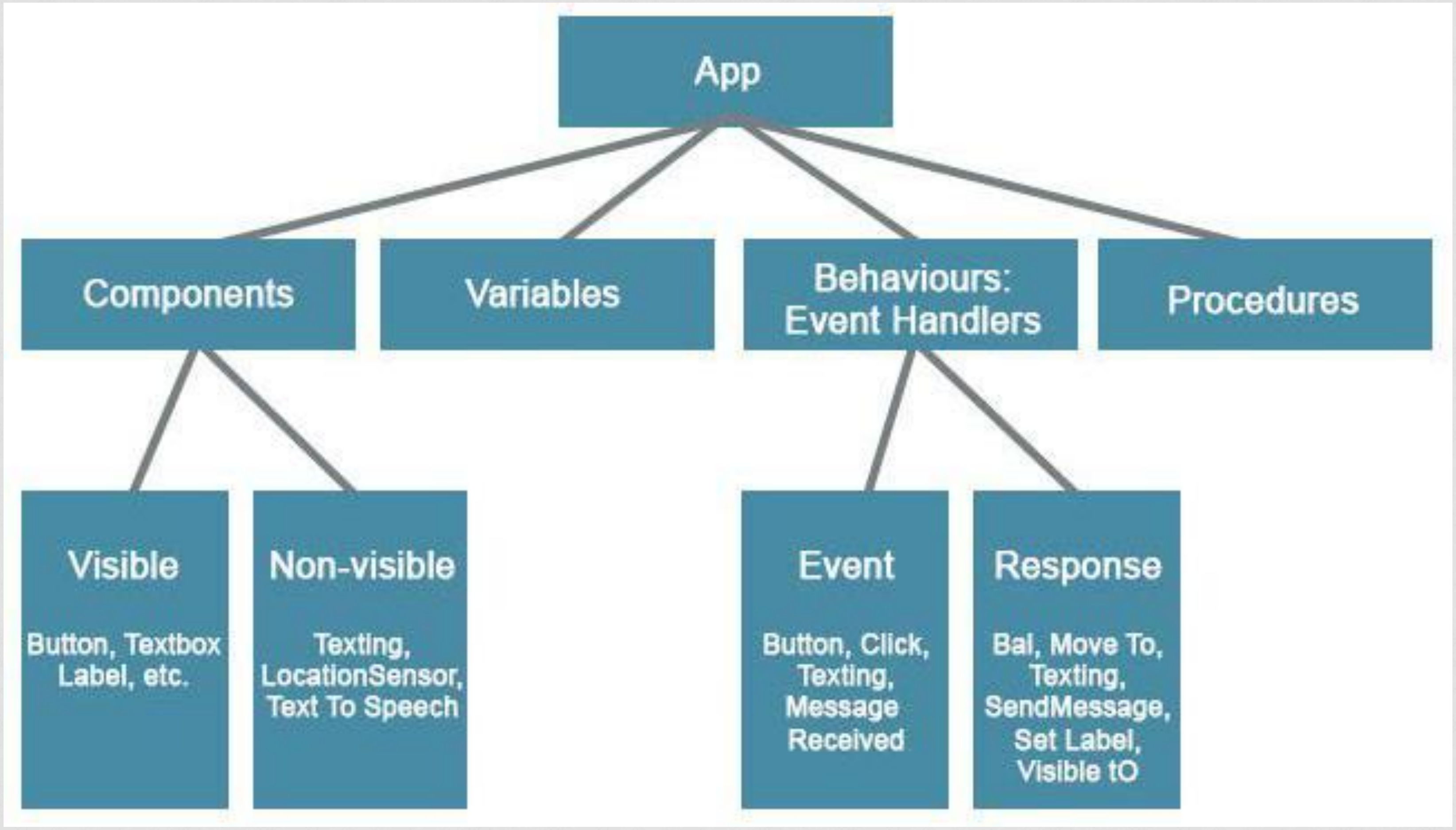
App Inventor has the following limitations in terms of the apps you can build:

- **Limited UIs.** The user interface builder has improved but is still a bit buggy and limited, so you can't build any user interface.
  - For instance, you can't create apps with multiple screens and handling orientation changes has some glitches. These problems are not fundamental to the design of App Inventor and will soon be fixed.
- **Limited Access to the device.** There are not yet components for all the data and functionality of the phone.
  - For instance, you can't save and retrieve files from the file system and you have only limited access to the contact list (e.g., you cannot create groups).
- **Limited Access to Web.** You can only access APIs that follow a particular protocol (App-Inventor-compatible APIs).
- **No polymorphic components.** Function blocks are tied to specific components, so there is no way to call functions on a generic component.
  - For instance, if you create a procedure MoveXY, it has to be tied to a specific image sprite, not a general image sprite.
- **Limited access to the Android Market.** The apps generated by App Inventor lack the required configuration for direct inclusion in the market. However, there is now a workaround for market publication.

APP INVENTOR ARCHITECTURE

App Inventor has its own architecture that developers must fully understand in order to use it effectively when creating mobile apps.

- Figure 3 represents an app's internal structure, which is broken into **two parts**:
  - **components**
  - **behaviors**
- Roughly, these correspond to the two main Windows developers used in App Inventor. Developers use the Component Designer to specify the objects (components) of the app, and use the Blocks Editor to program how the app will respond to the user and external events (the app's behaviour).


App
Components
Variables
Behaviours: Event Handlers
Procedures
Visible
Button, Textbox Label, etc.
Non-visible
Texting, LocationSensor, Text To Speech
Event
Button, Click, Texting, Message Received
Response
Bal, Move To, Texting, SendMessage, Set Label, Visible tO

APP INVENTOR COMPONENTS

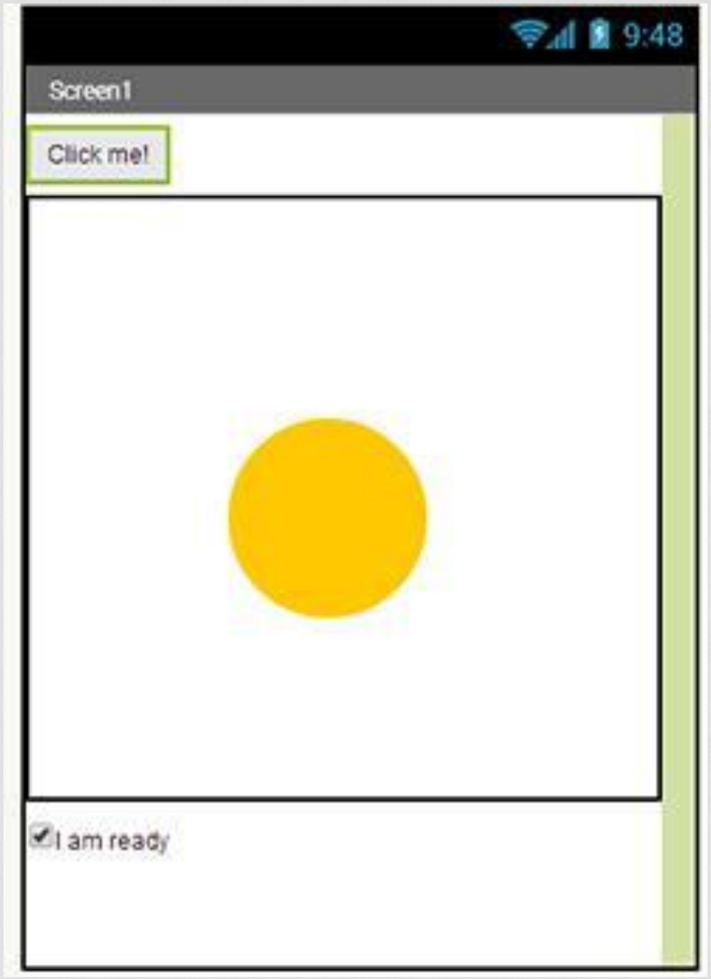
- There are two main types of components in an app:

  visible

  non-visible
- The app's visible components are the ones you can see when the app is launched; things like **buttons**, **text boxes**, and **labels**, as shown in Figure 4.

- The non-visible components are the technology within the device that carries out tasks for your app. Non-visible components are those you can't see, so they're not part of the user interface. Instead, they provide access to the built-in functionality of the device; for example, the **Texting** component **sends** and **processes SMS texts**, the **Location Sensor component** determines the device's location, and the **Text-to-Speech component** articulates the messages.

APP VISIBLE ARCHITECTURE



APP INVENTOR PROPERTIES

- Both visible and non-visible components are defined by a set of **properties**.
- Properties are memory slots for storing information about the component.
- **Visible components**, for instance, have properties like width, height, and alignment, which together define how the component appears. So a button that looks like the Submit button in Figure 5 to the end user is defined in the Component Designer with a set of properties, including those shown in Table 1.
- You can think of **properties** as something like the **cells** you see in a **spreadsheet**. You **modify** them in the Component Designer to define the initial appearance of a component. If you change the **number** in the width slot from 50 to 70, the button will appear wider, both in the Designer and in the app. Note that the end user of the app doesn’t see the 70; the button’s change in width is only visible.

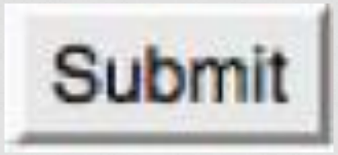


Figure 5: Submit button

Button Properties

Width	Height	Alignment	Text
100	50	centre	submit

APP INVENTOR **BEHAVIOR**

Behavior

An app's components are generally straightforward to understand: a text box is used for entering information; a button is for clicking, and so on.

An app's behavior, defines how the app should respond to events, both user initiated (e.g., a button click) and external (e.g., an SMS text arriving to the phone).

The difficulty of specifying such interactive behavior is why programming is so challenging. Fortunately, App Inventor provides a visual 'blocks' language perfectly suited for specifying behaviors.

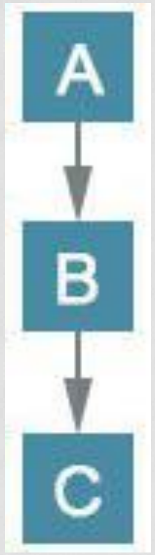

$$\begin{tikzcd}A \arrow[d] \\ B \arrow[d] \\ C\end{tikzcd}$$

Figure 7 above, shows that the computer (chef) should perform sequence of instructions.
A typical app might start a bank transaction (A),
perform some computations and modify a customer’s account (B),
and then print out the new balance on the screen (C).

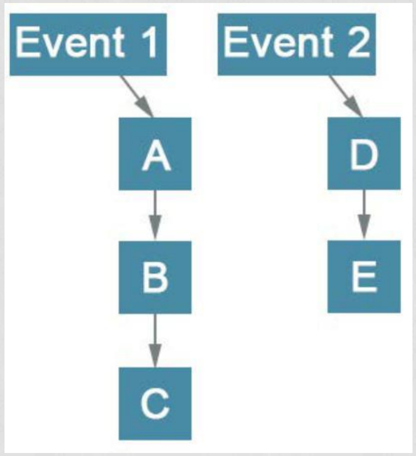
APP AS A SET OF EVENT HANDLERS

Most apps today, whether they're for mobile phones, the Web, or desktop computers, don't fit the recipe paradigm anymore.

They don't perform a series of instructions in a predetermined order; instead they most commonly react to events initiated by the app's end user.

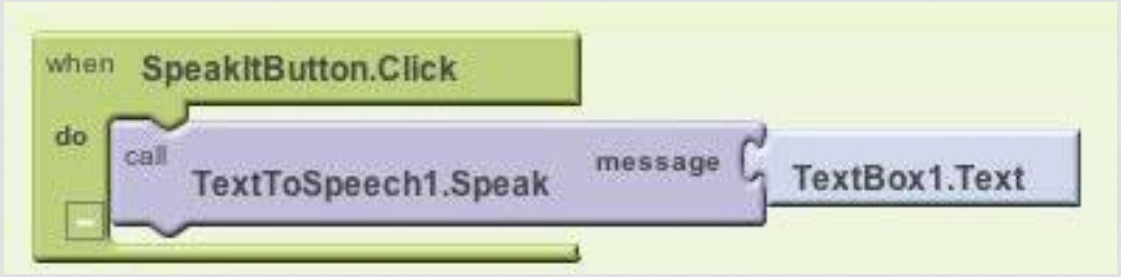
APP AS A SET OF EVENT HANDLERS

- For example, if the user **clicks** a **button**, the app responds by performing an operation (e.g., sending a text message).
- For **touchscreen** phones and devices, the act of dragging your finger across the screen is another event.
- The app might respond to that event by drawing a line from the point of your original touch to the point where you lifted your finger.
- These types of apps are better conceptualized as a set of components that respond to events. The apps do include 'recipes' -- sequences of instructions - but each recipe is only performed in response to an event, as shown in Figure. So, as events occur, the app reacts by calling a **sequence** of **functions**.
- **Functions:** are things you can do to or with a component — operations like sending an SMS text, or property-changing operations such as changing the text in a label of the user interface.
- **Call a function:** it means to invoke it, to make it happen. We call an event and the set of functions performed in response to it, an event handler.



SAMPLE APPLICATION

- consider an app, **SpeakIt**, that responds to button clicks by speaking the text the user has entered, aloud. This application could be programmed with a single event handler as shown in Figure below.



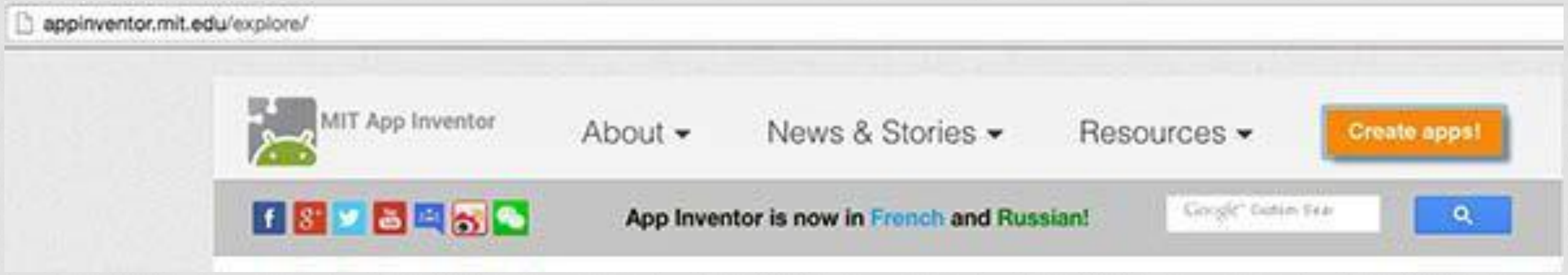
```
when SpeakItButton.Click
do  call TextToSpeech1.Speak  message  TextBox1.Text
```

when the user clicks the button named SpeakItButton , the TextToSpeech component should speak the words the user has entered in the text box named TextBox1. The response is the call to the function TextToSpeech1.Speak. The event is SpeakItButton.

LOG IN TO APP INVENTOR

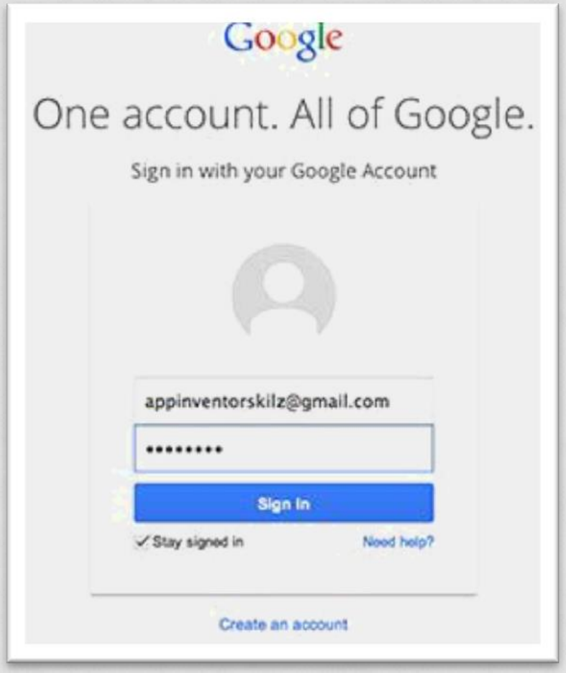
- To get started, go to App Inventor on the web. Click the orange 'Create apps' button from the App Inventor website bellow.

  http://appinventor.mit.edu/


appinventor.mit.edu/explore/
MIT App Inventor
About
News & Stories
Resources
Create apps!
App Inventor is now in French and Russian!

LOG IN TO APP INVENTOR

- Log in to App Inventor with a gmail (or google) username and password.


Google

One account. All of Google.

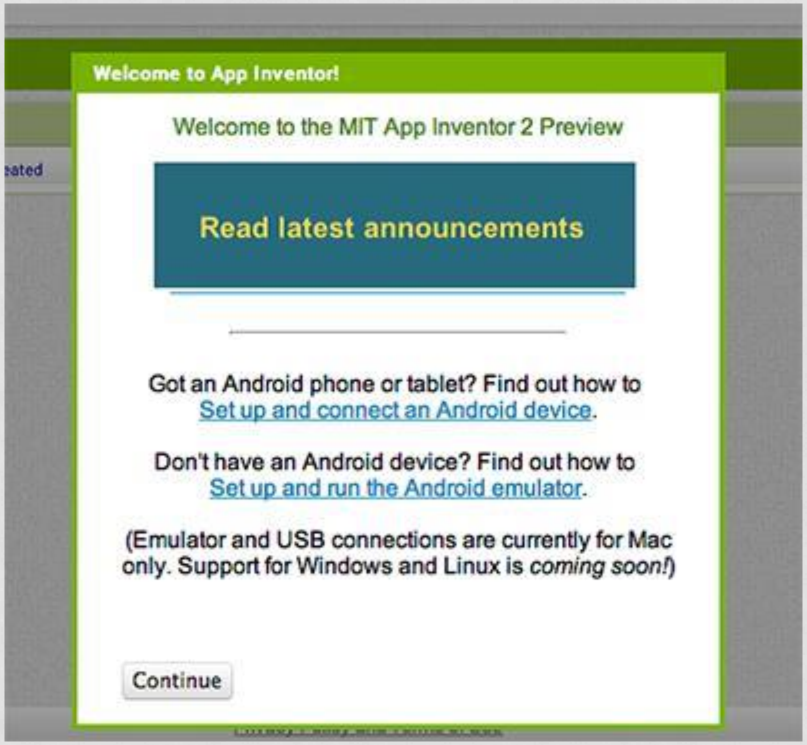
Sign in with your Google Account

appinventorskilz@gmail.com

•••••••••

LOG IN TO APP INVENTOR

- Click 'Continue' to dismiss the splash screen.



Welcome to App Inventor!

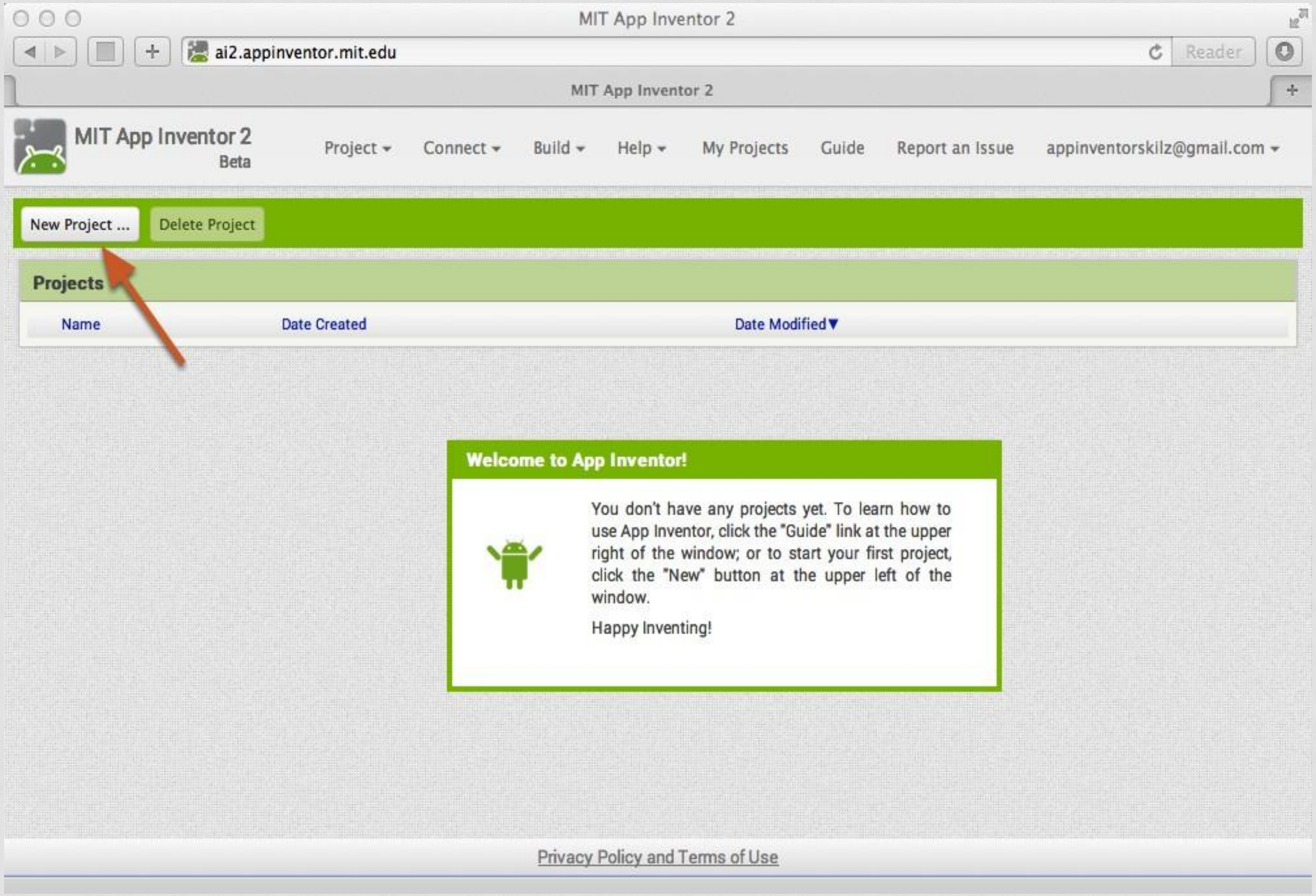
Welcome to the MIT App Inventor 2 Preview

Got an Android phone or tablet? Find out how to Set up and connect an Android device.

Don't have an Android device? Find out how to Set up and run the Android emulator.

(Emulator and USB connections are currently for Mac only. Support for Windows and Linux is *coming soon!*)

START A NEW PROJECT



Welcome to App Inventor!

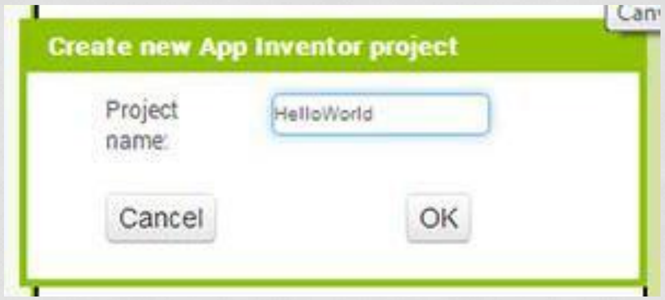
You don't have any projects yet. To learn how to use App Inventor, click the "Guide" link at the upper right of the window; or to start your first project, click the "New" button at the upper left of the window.

Happy Inventing!

Privacy Policy and Terms of Use

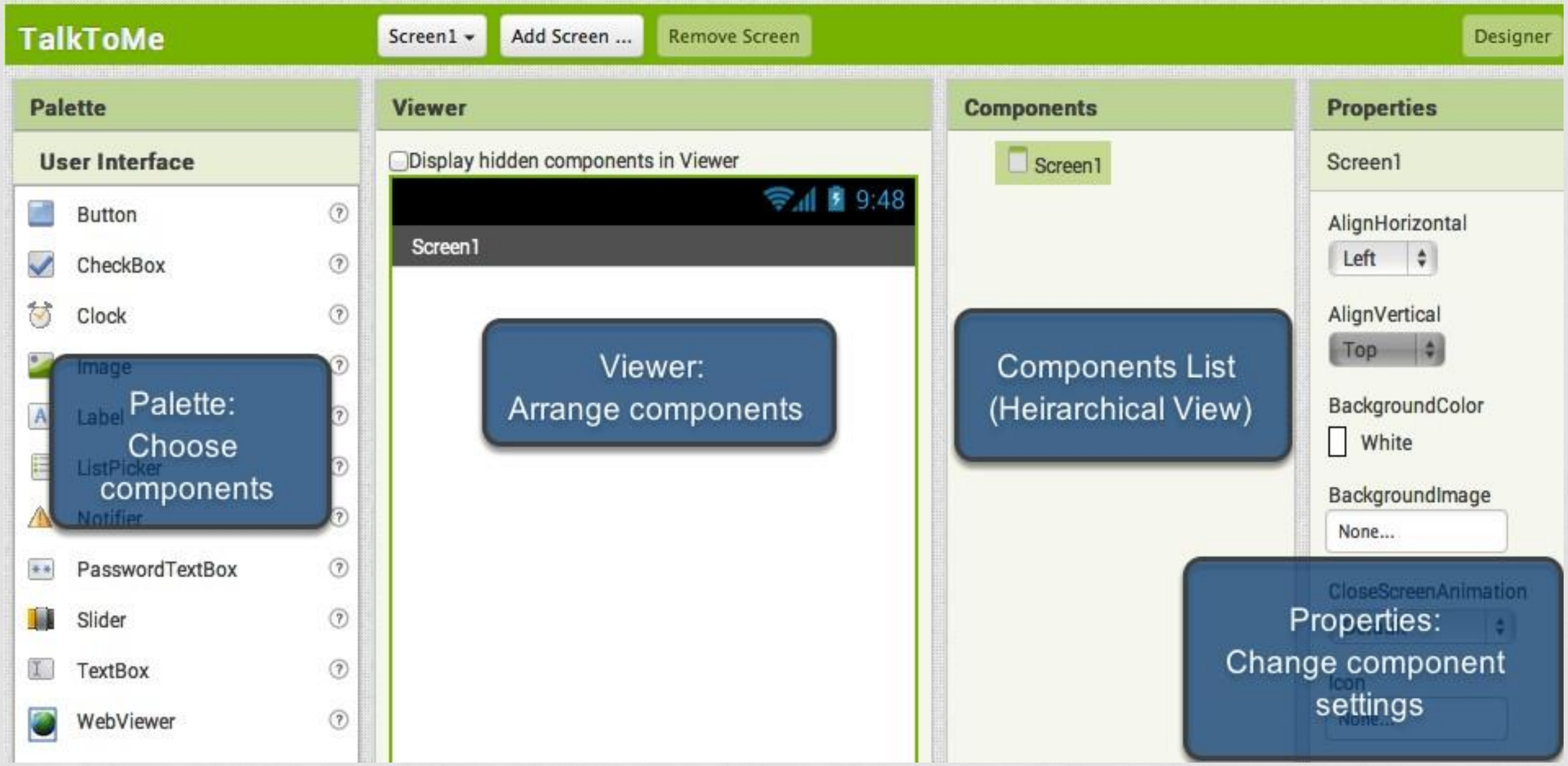
START A NEW PROJECT

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



START A NEW PROJECT

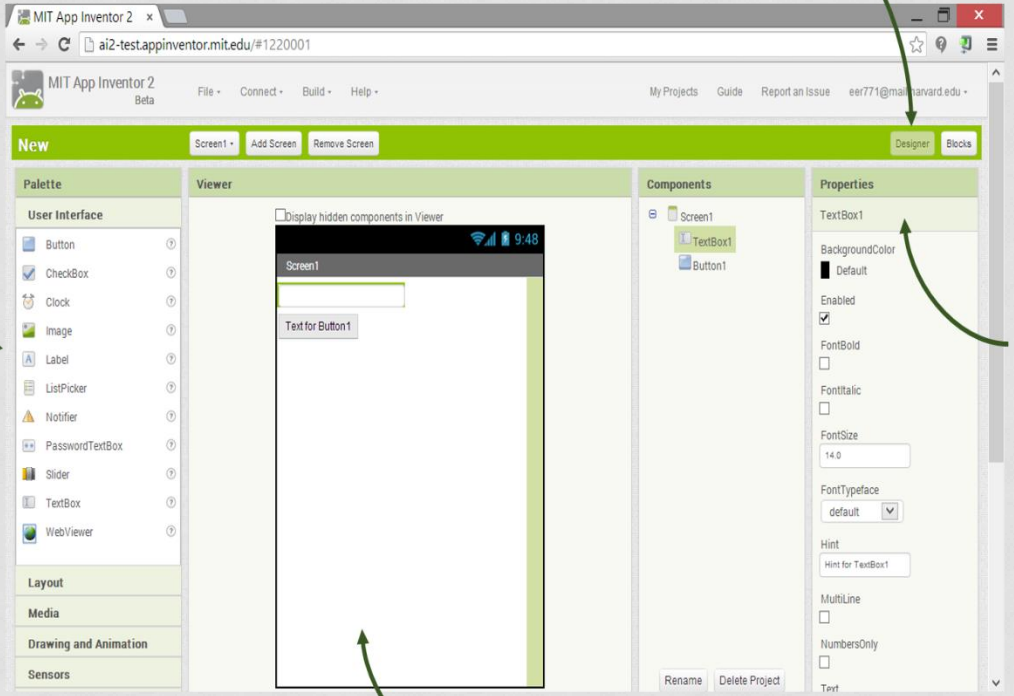
The Design Window, or simply "Designer" is where you lay out the look and feel of your app, and specify what functionalities it should have. You choose things for the user interface things like Buttons, Images, and Text boxes, and functionalities like Text-to-Speech, Sensors, and GPS.



APP INVENTOR DESIGNER

Palette: Find your components and drag them to the Viewer to add them to your app.

Designer Button: Click from any tab to go to the Designer tab.



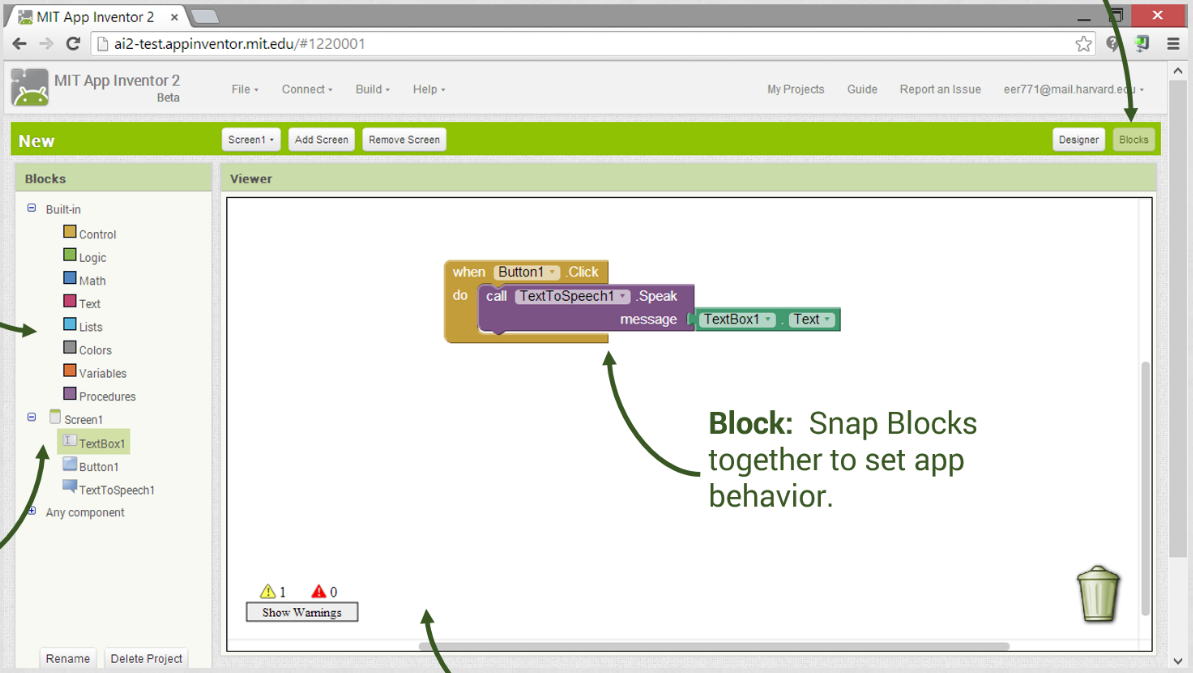
Properties: Select a Component in the Components List to change its properties (color, size, behavior) here.

Viewer: Drag components from the Palette to the Viewer to see what your app will look like.

APP INVENTOR BLOCKS EDITOR

Built-In Drawers: Find Blocks for general behaviors you may want to add to your app and drag them to the Blocks Viewer.

Blocks Button: Click from any tab to go to the Blocks tab.

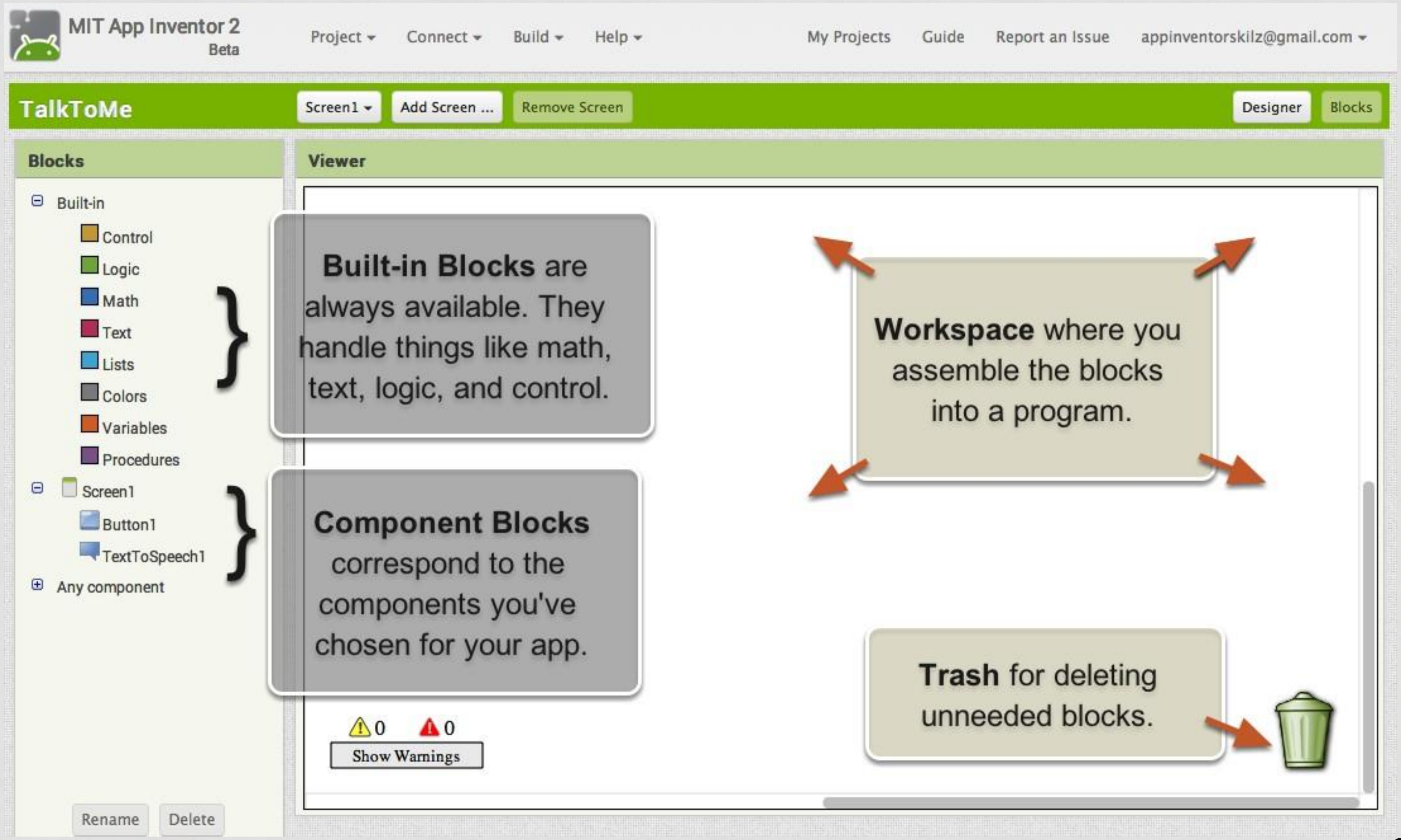


Component-Specific Drawers: Find Blocks for behaviors for specific Components and drag them to the Blocks Viewer.

Block: Snap Blocks together to set app behavior.

Viewer: Drag Blocks from the Drawers to the Blocks Viewer to build relationships and behavior.

APP INVENTOR BLOCKS EDITOR



Built-in Blocks are always available. They handle things like math, text, logic, and control.

Component Blocks correspond to the components you've chosen for your app.

Workspace where you assemble the blocks into a program.

Trash for deleting unneeded blocks.

INSTALLATION PROCESS OF APP INVENTOR

1. Click on the link bellow to download the app:

   http://appinventor.mit.edu/explore/ai2/setup-emulator.html

2. Locate the file **MIT_Appinventor_Tools_2.3.0 (~80 MB)** in your Downloads file or your Desktop. The location of the download on your computer depends on how your browser is configured.
3. Open the file.
4. Click through the steps of the installer. Do not change the installation location but record the installation directory, because you might need it to check drivers later. The directory will differ depending on your version of Windows and whether or not you are logged in as an administrator.
5. You may be asked if you want to allow a program from an **unknown publisher** to make changes to this computer. **Click yes**.

Copy protected with Online-PDF-No-Copy.com