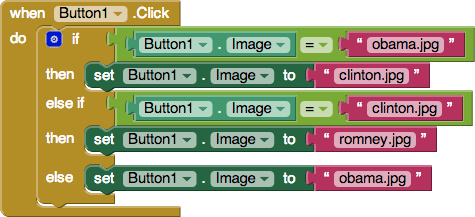
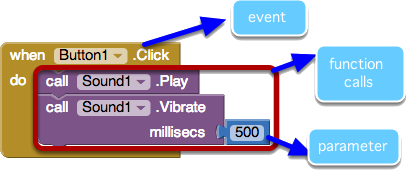
Quiz 1:

**Name and define three ways in which the Android platform is more open than iOS (iPhone). Which of these is key to why App Inventor was developed for Android?**

Android is open source. Programmers can access the program source code and change it, making a new app. It is the Android OS itself which is open source, not (necessarily) Android apps.

**Sketch the blocks for an app:**

**In the following blocks, circle and label:**

**Define the term parameter in the context of an App Inventor app:**

The parameters of a function provide the information the function needs to perform its job, e.g., how long should the phone vibrate, or how big a circle should be drawn.

Quiz 2

**What is a variable?**

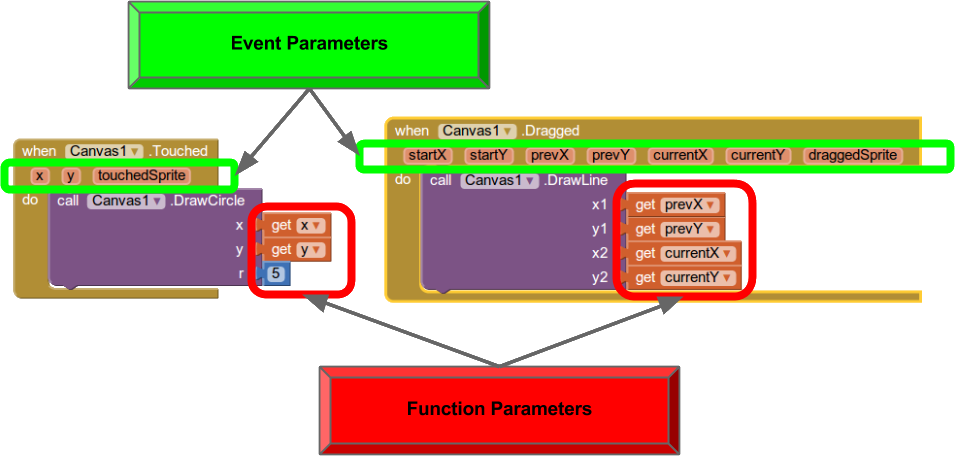
A variable is a named memory cell where information can be stored.

**In the PaintPot app we needed a variable to keep track of the circle size, however, we did not use one for keeping track of the line size. Why?**

We did not make a variable to represent the line size because Canvas has a property called LineWidth. We used this property to keep track of the size of the line. We had to make a seperate variable for circle size because Canvase does not contain a property for representing the circle Size.

**In App Inventor, how are a variable and a property similar? How are they different?**

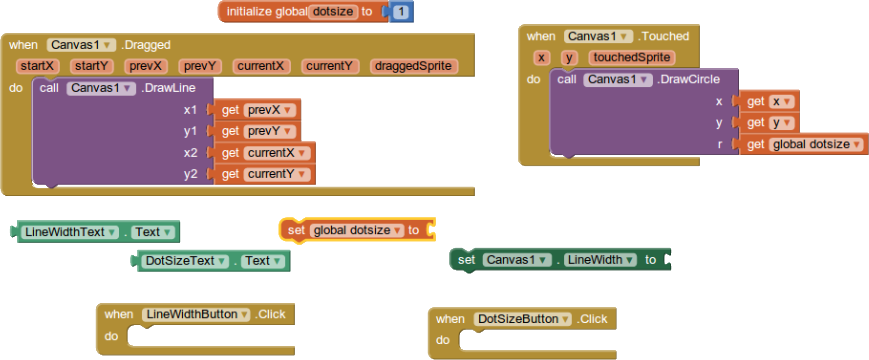
Component properties and variables are both "named memory cells". They are different in that a property is part of a component, whereas a variable is a "free-agent" memory cell not associated with a particular component. So properties are named in the form < componentName >.< propertyName >, e.g., Canvas.Width, while variables are just a single name, e.g., dotsize.



**Circle and label the event parameters and function parameters in the following blocks:**

**Assume we make a small modification to Canvas1's drawLine function. We decide to change x1 to be set to startX and y1 to be set to starty. How would this change the behavior of the app?**

All of the lines drawn during a dragged event will now start at the same initial clicked location and end wherever the user is currently clicking. In other words, you will have a bunch of lines coming from one point and ending in different places.

**Define the behavior for this app by drawing the completed DotSizeButton.Click, and LineWidthButton.Click event handlers. The needed blocks are shown below.**