STATE QUIZ APP

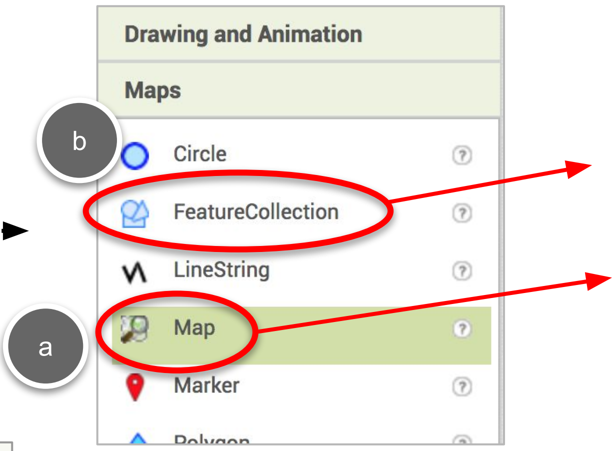
START HERE

This tutorial is an example of using GeoJSON files to create a FeatureCollection on a Map..

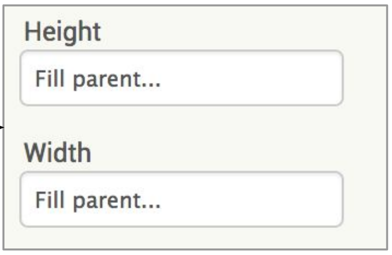
In this tutorial, you will make an app to test people's geography knowledge of US states!

1 Open a new project in App Inventor and name it "**StateQuiz**".

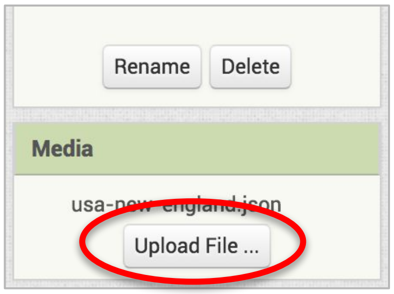
2 Add a new **Map** component in the Designer. Then drag a **FeatureCollection** component onto the Map.



3 Set the *Height* and *Width* for the Map to "Fill Parent".

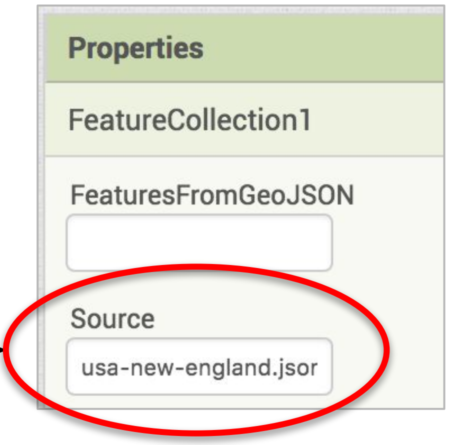

Height
Fill parent...
Width
Fill parent...

4 Download this GeoJSON file to your computer, the upload it as Media for your app.



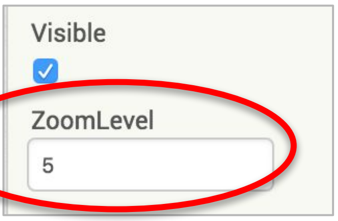
GeoJSON is a format for encoding a variety of geographic data structures.

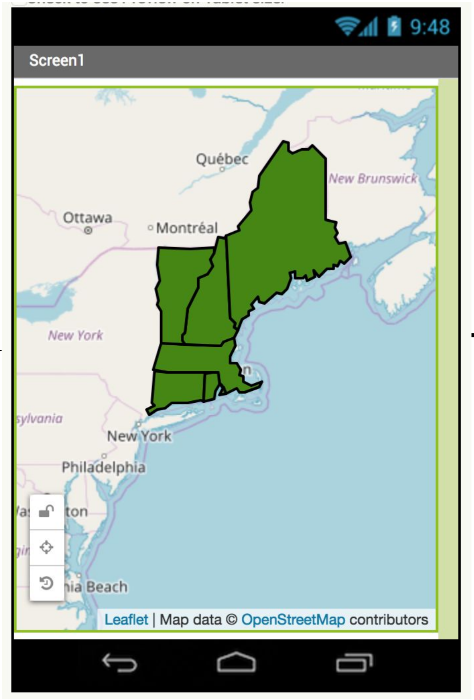
5 Set the *Source* for **FeatureCollection1** to the uploaded usa-new-england.json file.

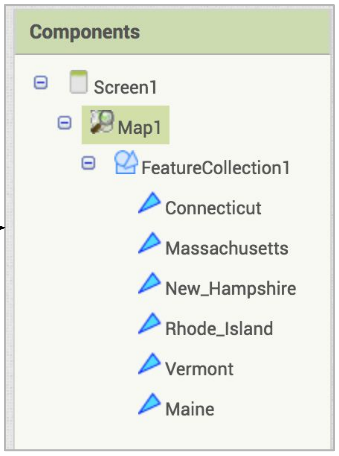

Properties
FeatureCollection1
FeaturesFromGeoJSON
Source
usa-new-england.json

SETTING PROPERTIES

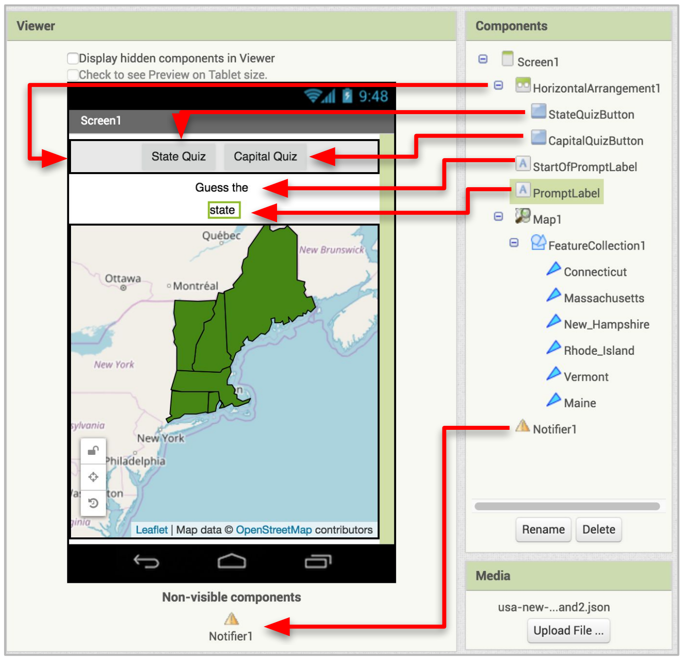
6 Update the *ZoomLevel* for **Map1** to **5**. You may have to move the center of the map, but you should see the outline of the New England states in green. This is based on the GeoJSON file. You will also see the six New England states appear as Features in the FeatureCollection.


Visible
✓
ZoomLevel
5




Components
Screen1
Map1
FeatureCollection1
Connecticut
Massachusetts
New_Hampshire
Rhode_Island
Vermont
Maine

7 Add more UI components to complete the layout, as shown to the left.

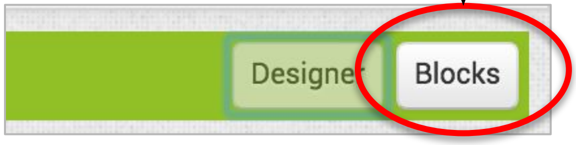


Screen1
HorizontalArrangement1
StateQuizButton
CapitalQuizButton
StartOfPromptLabel
PromptLabel
Map1
FeatureCollection1
Connecticut
Massachusetts
New_Hampshire
Rhode_Island
Vermont
Maine
Notifier1

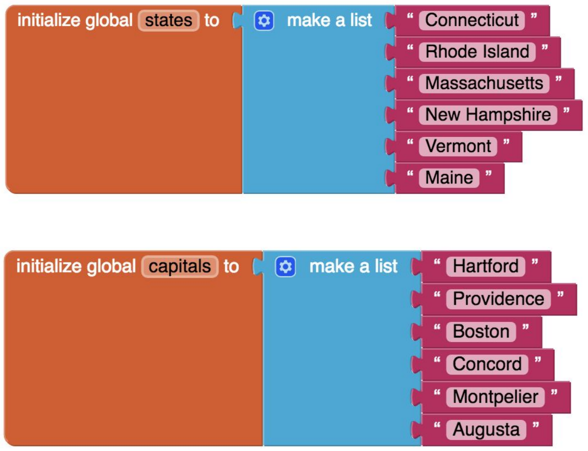
You can search online for readymade GeoJSON files for states or countries.

CODING THE APP

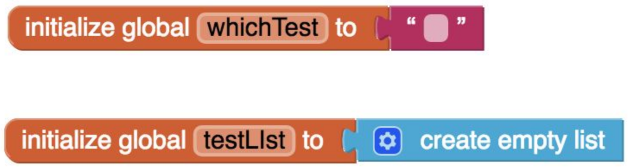
8 Switch to the Blocks Editor.



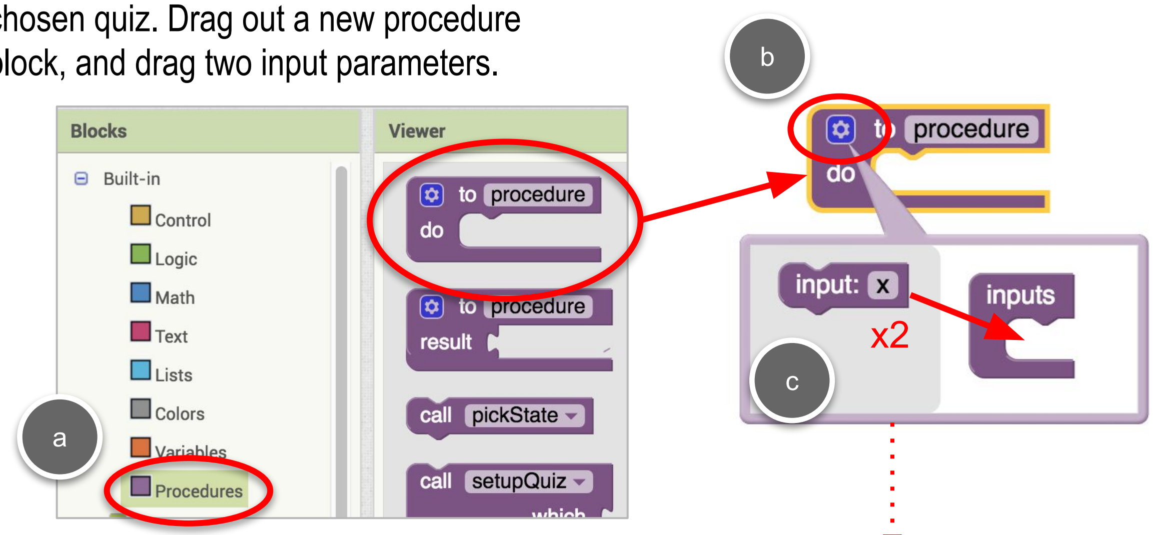
9 Add two new variables, one for states, and one for capitals. Initialize them to lists for the six states and their matching capitals. Make sure they are in the same order!



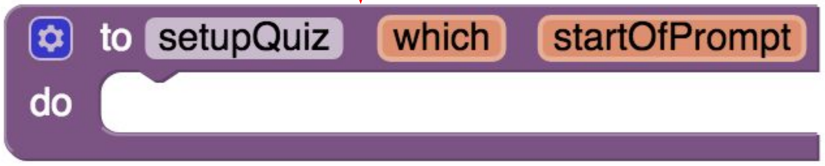
10 Add two more variables to hold information for which quiz (states or capitals) and to make a copy of the appropriate list for the test. We'll need a copy as we're going to remove items as they are answered correctly.


initialize global whichTest to " "
initialize global testLIst to create empty list

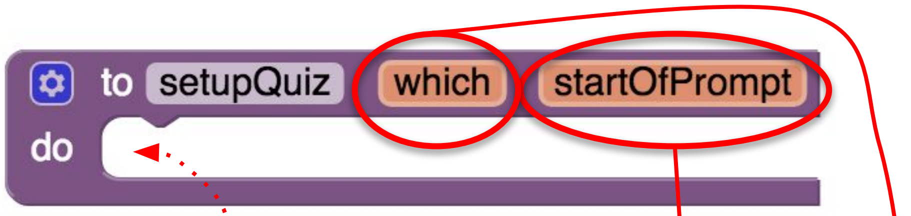
11 Let's make a procedure to setup up the chosen quiz. Drag out a new procedure block, and drag two input parameters.



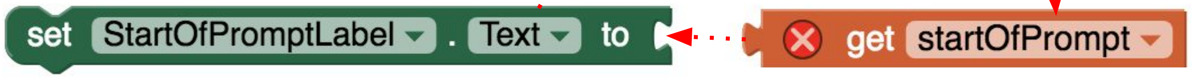
12 Name the procedure "setupQuiz", with parameters "which" and "startPrompt".



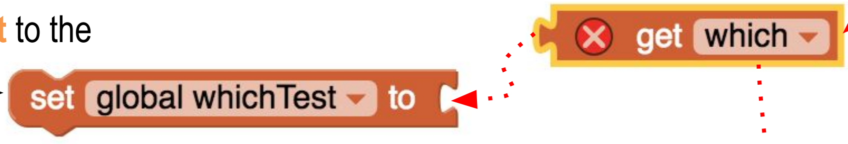
SETUP THE QUIZ



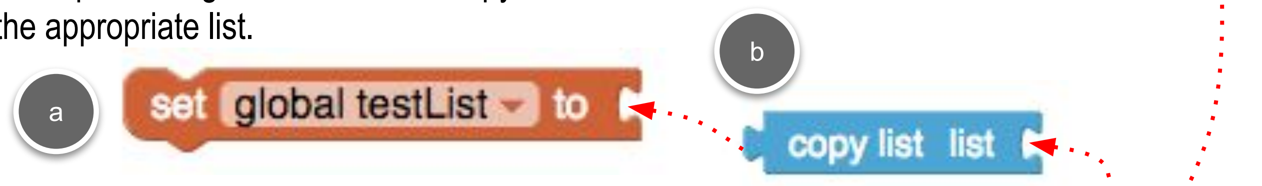
13 Set the **StartOfPromptLabe** parameter startOfPrompt.



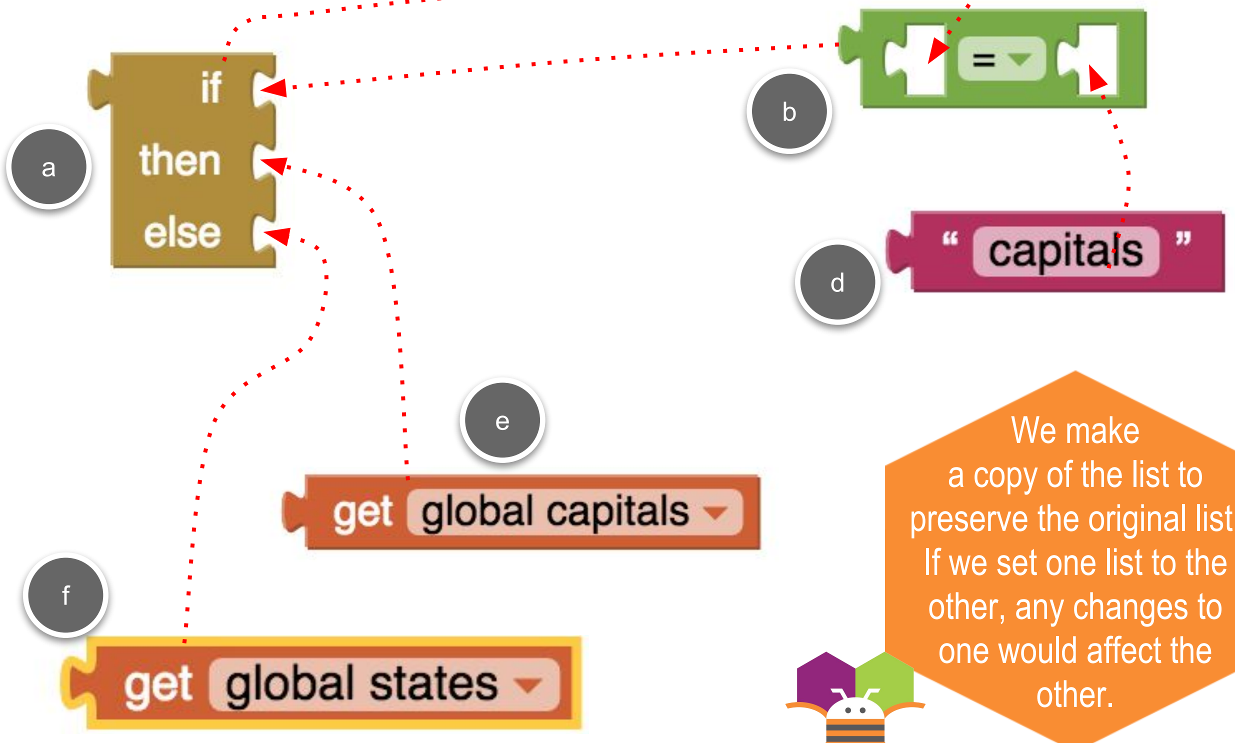
14 Set the global variable whichTest to the input parameter which.



15 Let's set the testList, based on which quiz we're presenting. We will make a copy of the appropriate list.

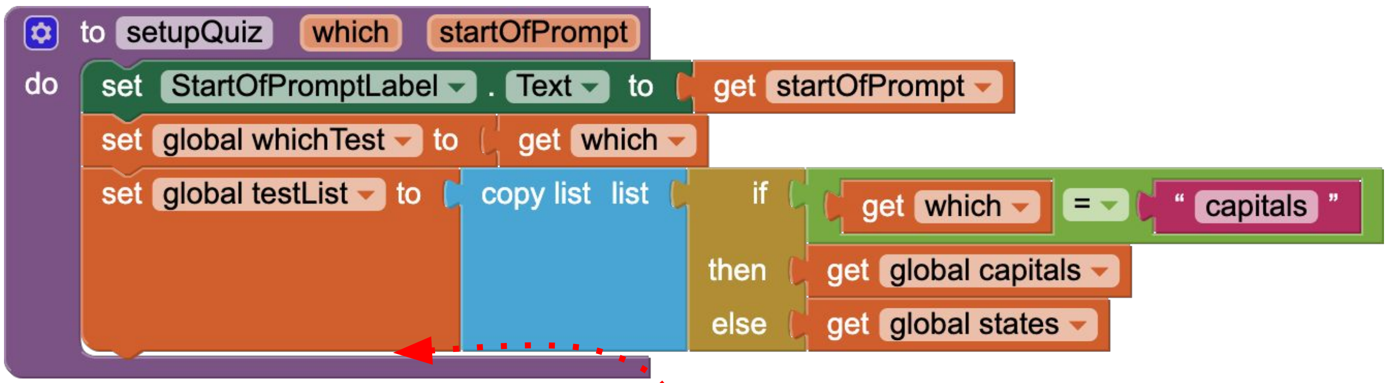


16 Copy the appropriate list, states or capitals, based on the which input. Use an if-then-else block to append the appropriate list.



We make a copy of the list to preserve the original list. If we set one list to the other, any changes to one would affect the other.

CHANGE THE BACKGROUND COLOR OF STATES



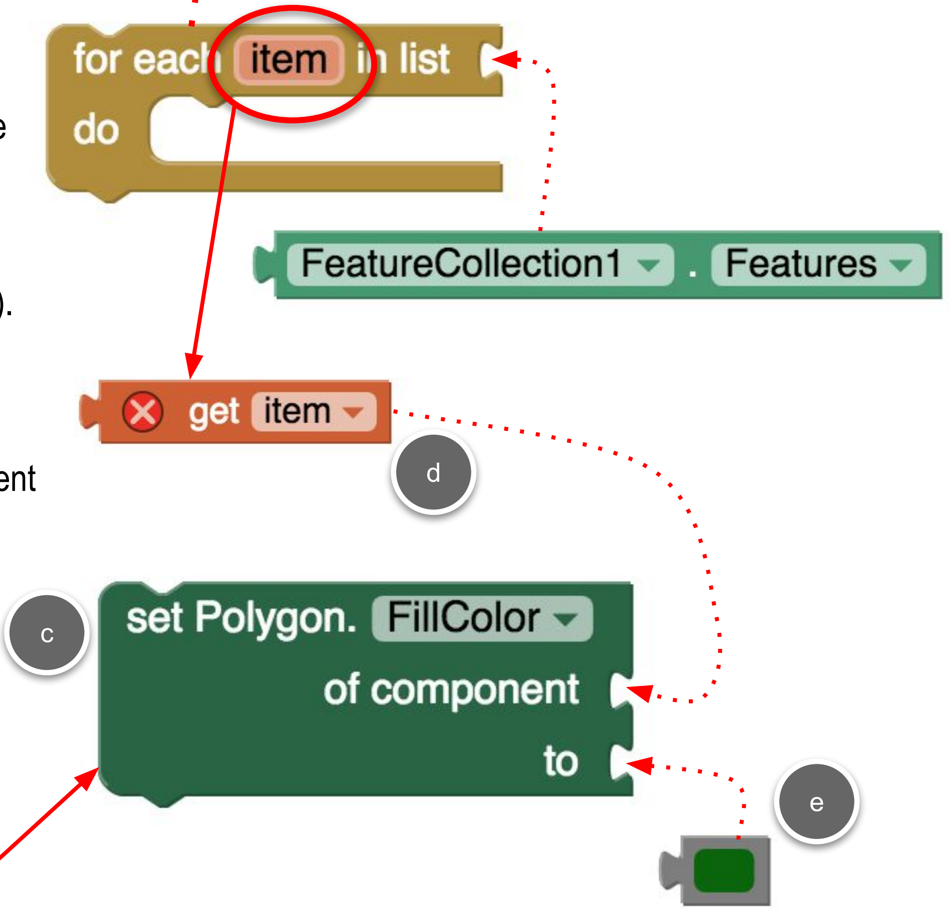
```
to setupQuiz which startOfPrompt
do  set StartOfPromptLabel . Text to get startOfPrompt
    set global whichTest to get which
    set global testList to copy list list  if   get which = " capitals "
                                           then get global capitals
                                           else get global states
```

Let's reset the colors of each state to the dark green (or choose your own favorite color). As users get correct answers, we'll change the background to a different color.

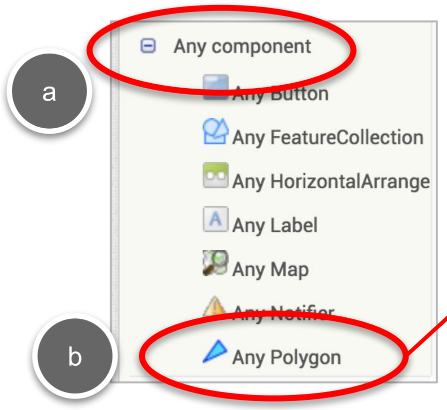
17 Drag out a **for each item in list** block and add it to the procedure

18 The list will be **FeatureCollection1.Features** which is a list of features (states).

19 Use the Any Component drawer to use the Any Polygon component (each state is a polygon).



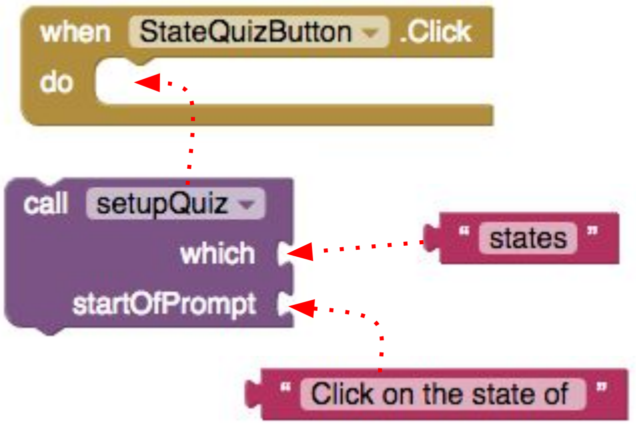
```
for each item in list   FeatureCollection1 . Features
do

get item

set Polygon. FillColor
    of component   get item
    to
```



CALL SETUPQUIZ

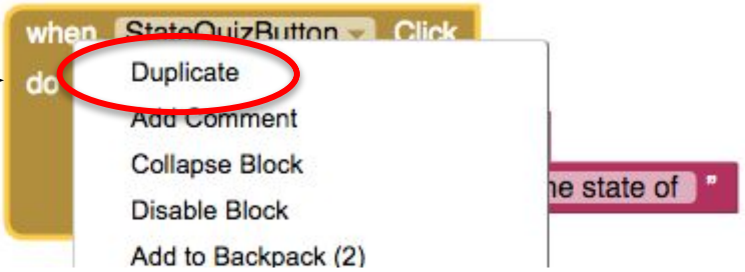
20 Drag out a **StateQuizButton.Click** block and add the **call setupQuiz** block to the event block.

21 which is "states" and startOfPrompt is "Click on the state of ".

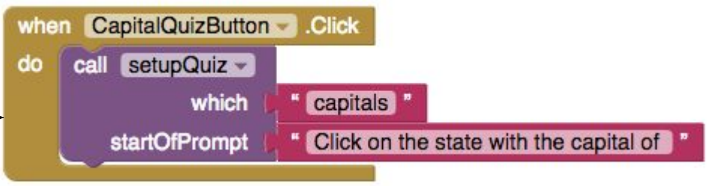


```
when StateQuizButton .Click
do

call setupQuiz
     which        " states "
     startOfPrompt " Click on the state of "
```

22 Duplicate the entire block, and change "StateQuizButton" to "CapitalQuizButton".

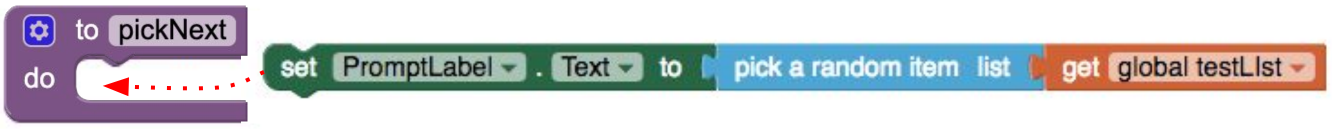


23 Change which and startOfPrompt.



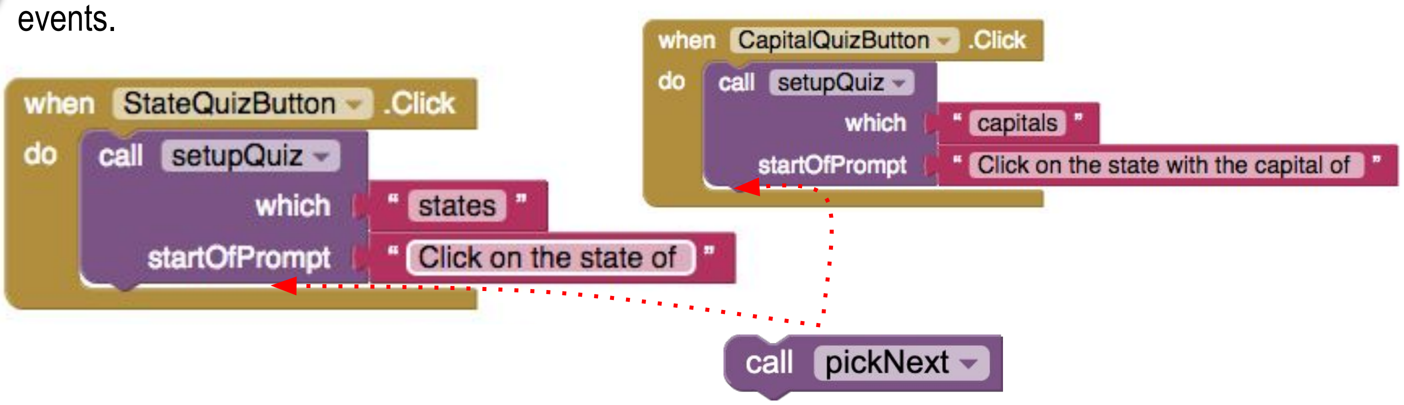
```
when CapitalQuizButton .Click
do  call setupQuiz
         which         " capitals "
         startOfPrompt " Click on the state with the capital of "
```

24 Let's make another procedure, called **pickNext** to randomly pick a state/capital from the testList.



```
to pickNext
do   set PromptLabel . Text to  pick a random item list  get global testList
```

25 Add **call pickNext** to both the quiz button click events.

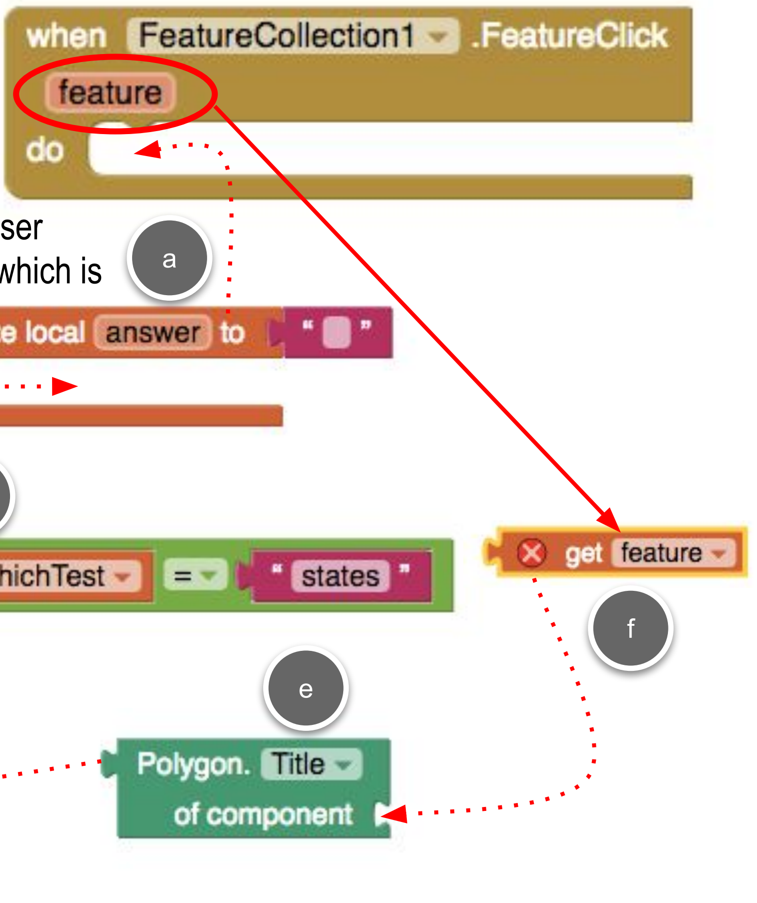


```
when StateQuizButton .Click
do  call setupQuiz
         which         " states "
         startOfPrompt " Click on the state of "

when CapitalQuizButton .Click
do  call setupQuiz
         which         " capitals "
         startOfPrompt " Click on the state with the capital of "

call pickNext
```

CLICKING ON A STATE

The last thing we need to do is to handle when the user clicks on one of the states, to answer the quiz question.

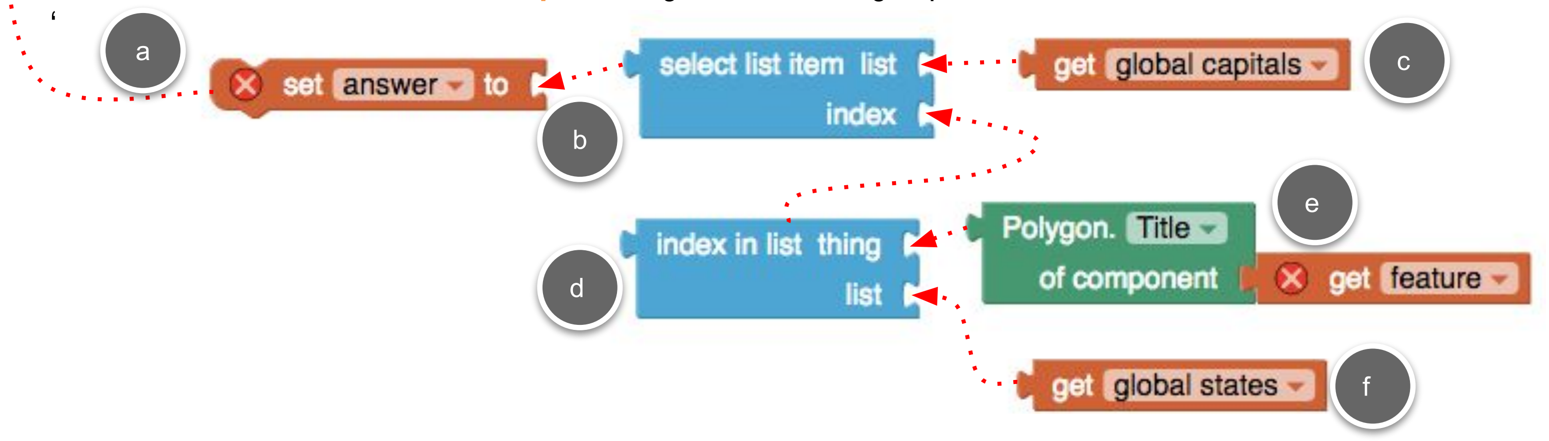
26 Drag out a **FeatureCollection1.FeatureClick** block.


when FeatureCollection1 .FeatureClick
feature
do
a
initialize local answer to " "
in
b
if
then
else
c
get global whichTest = " states "
get feature
f
e
Polygon. Title
of component
set answer to
d

27 Add a local variable, called answer, and if the user is doing the state quiz, set it to the state name, which is the Feature (Polygon) title.

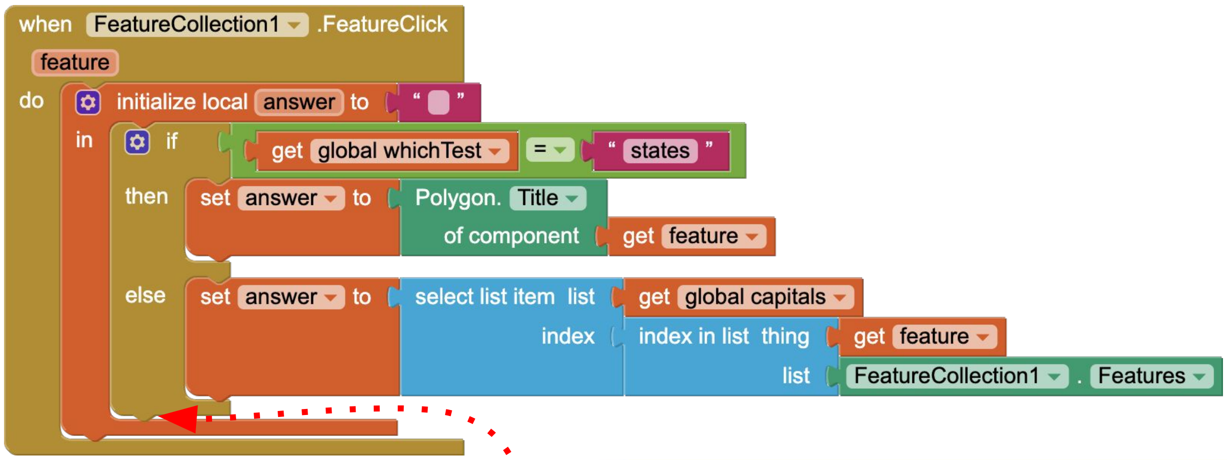
28 If the user is doing the capital quiz, get the index of the state from the states list, then use that to index into capitals to get the matching capital.

‘



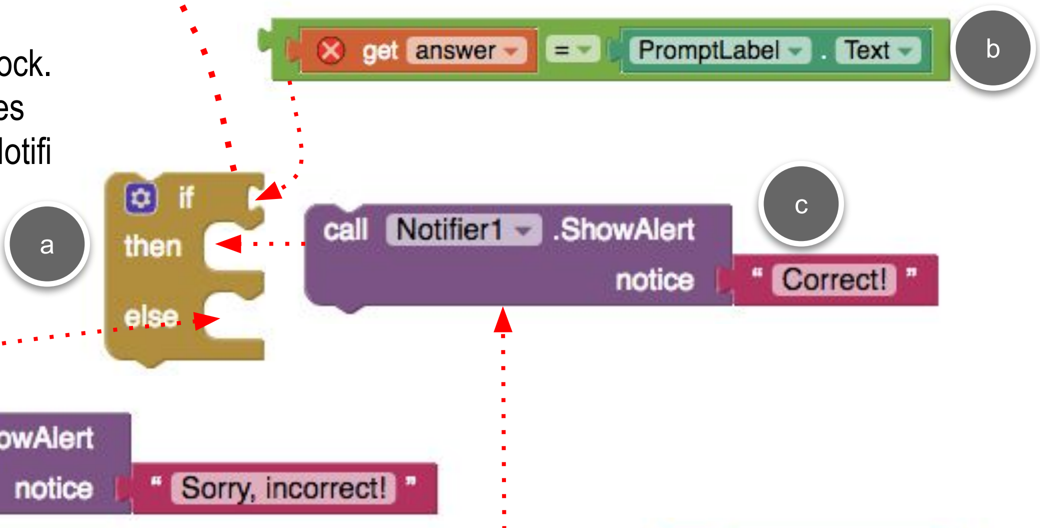
TESTING FOR CORRECT ANSWER

Now test what the user clicks on matches the state or capital.



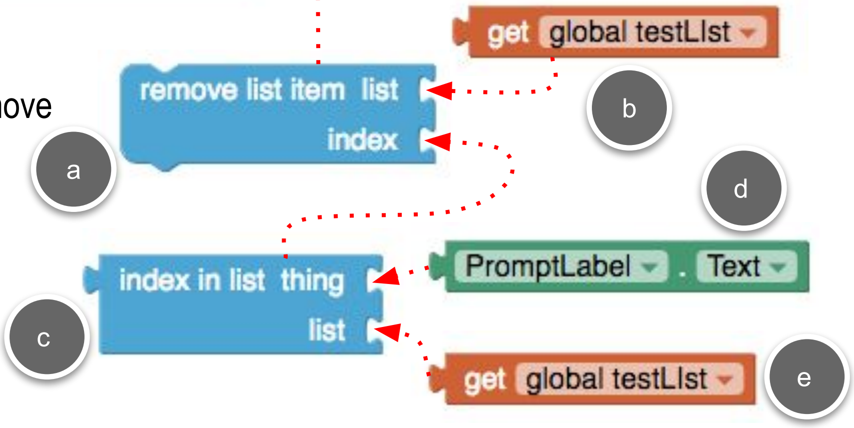
```
when FeatureCollection1 .FeatureClick
  feature
do  initialize local answer to " "
    in  if  get global whichTest = " states "
        then  set answer to  Polygon. Title
                                of component  get feature
        else  set answer to  select list item  list  get global capitals
                                               index  index in list  thing  get feature
                                                                     list   FeatureCollection1 . Features
```

29 Drag out an **if-then-else** block. Check if the answer matches PromptLabel, and set the Notifi message appropriately.



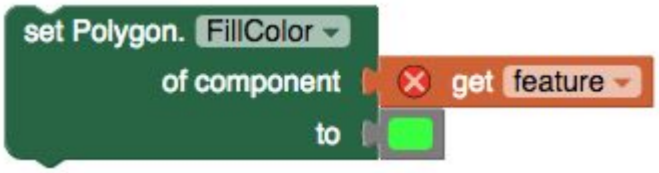
```
b: get answer = PromptLabel . Text
a: if / then / else
c: call Notifier1 .ShowAlert
     notice  " Correct! "
d: call Notifier1 .ShowAlert
     notice  " Sorry, incorrect! "
```

30 If the user is correct, we also want to remove the item from **testList**. Use PromptLabel to find the correct index in testList.



```
b: get global testLIst
a: remove list item  list
                     index
d: PromptLabel . Text
c: index in list  thing
                  list
e: get global testLIst
```

31 And then, optionally, signal a correctly answered state by setting the color of the polygon to a different color.

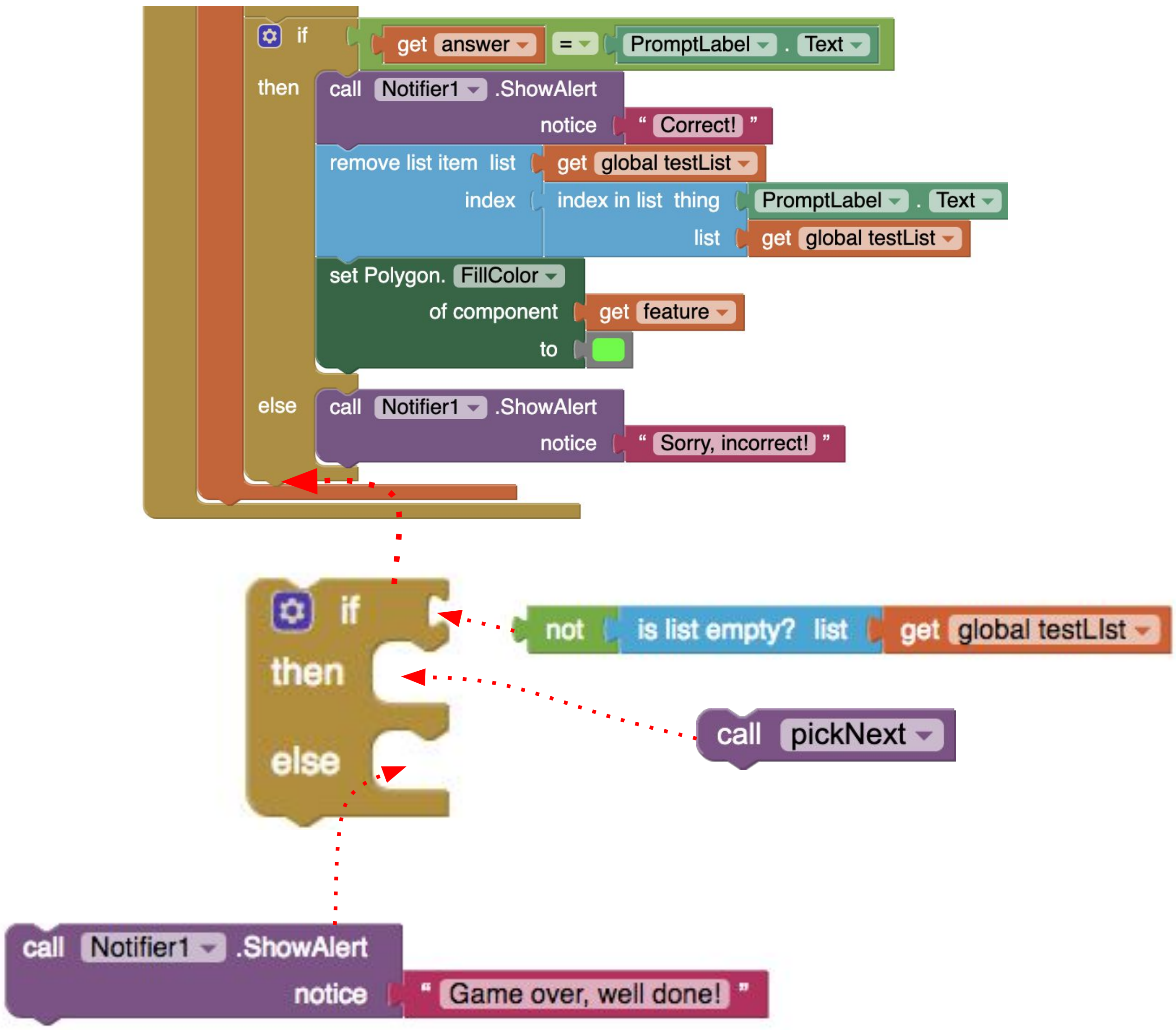


```
set Polygon. FillColor
    of component  get feature
    to  🟩
```

CHECK FOR EMPTY LIST

32 Last thing is another **if-then-else** to check if the testList is empty, which means the quiz is over. If the list is not empty, pick another state for the next question. Otherwise, let the user know the quiz is over.

‘



```
if   get answer = PromptLabel . Text
then call Notifier1 .ShowAlert
                    notice " Correct! "
     remove list item list get global testList
                    index  index in list thing PromptLabel . Text
                                         list  get global testList
     set Polygon. FillColor
          of component get feature
          to
else call Notifier1 .ShowAlert
                    notice " Sorry, incorrect! "

if   not is list empty? list get global testList
then call pickNext
else call Notifier1 .ShowAlert
                    notice " Game over, well done! "
```